IBM SPSS Modeler 17.1 Algorithms Guide

Note: Before using this information and the product it supports, read the general information under “Notices” on p. 403.

This edition applies to IBM SPSS Modeler 17 and to all subsequent releases and modifications until otherwise indicated in new editions.

Adobe product screenshot(s) reprinted with permission from Adobe Systems Incorporated.

Microsoft product screenshot(s) reprinted with permission from Microsoft Corporation.

Preface

IBM® SPSS® Modeler is the IBM Corp. enterprise-strength data mining workbench. SPSS Modeler helps organizations to improve customer and citizen relationships through an in-depth understanding of data. Organizations use the insight gained from SPSS Modeler to retain profitable customers, identify cross-selling opportunities, attract new customers, detect fraud, reduce risk, and improve government service delivery.

SPSS Modeler's visual interface invites users to apply their specific business expertise, which leads to more powerful predictive models and shortens time-to-solution. SPSS Modeler offers many modeling techniques, such as prediction, classification, segmentation, and association detection algorithms. Once models are created, IBM® SPSS® Modeler Solution Publisher enables their delivery enterprise-wide to decision makers or to a database.

About IBM Business Analytics

IBM Business Analytics software delivers complete, consistent and accurate information that decision-makers trust to improve business performance. A comprehensive portfolio of business intelligence, predictive analytics, financial performance and strategy management, and analytic applications provides clear, immediate and actionable insights into current performance and the ability to predict future outcomes. Combined with rich industry solutions, proven practices and professional services, organizations of every size can drive the highest productivity, confidently automate decisions and deliver better results.

As part of this portfolio, IBM SPSS Predictive Analytics software helps organizations predict future events and proactively act upon that insight to drive better business outcomes. Commercial, government and academic customers worldwide rely on IBM SPSS technology as a competitive advantage in attracting, retaining and growing customers, while reducing fraud and mitigating risk. By incorporating IBM SPSS software into their daily operations, organizations become predictive enterprises – able to direct and automate decisions to meet business goals and achieve measurable competitive advantage. For further information or to reach a representative visit *http://www.ibm.com/spss*.

Technical support

Technical support is available to maintenance customers. Customers may contact Technical Support for assistance in using IBM Corp. products or for installation help for one of the supported hardware environments. To reach Technical Support, see the IBM Corp. web site at *http://www.ibm.com/support*. Be prepared to identify yourself, your organization, and your support agreement when requesting assistance.

Contents

Adjusted Propensities Algorithms 1

Model-Dependent Method 1
General Purpose Method 1

Anomaly Detection Algorithm 3

Overview 3
Primary Calculations. 3
Notation 3
Algorithm Steps 4
Blank Handling 7
Generated Model/Scoring 7
Predicted Values 7
Blank Handling 7

Apriori Algorithms 9

Overview 9
Deriving Rules 9
Frequent Itemsets 9
Generating Rules. 10
Blank Handling 11
Effect of Options 11
Generated Model/Scoring 12
Predicted Values 12
Confidence 12
Blank Handling 12

Automated Data Preparation Algorithms 13

Notation 13
Date/Time Handling 14
Univariate Statistics Collection 15
Basic Variable Screening 17
Checkpoint 1: Exit? 17

Measurement Level Recasting 17
Outlier Identification and Handling 18
Missing Value Handling 19
Continuous Predictor Transformations 20
    Z-score Transformation 20
    Min-Max Transformation 20
Target Handling 21
Bivariate Statistics Collection 22
Categorical Variable Handling 25
    Reordering Categories 25
    Identify Highly Associated Categorical Features 26
    Supervised Merge 26
    P-value Calculations 27
    Unsupervised Merge 30
Continuous Predictor Handling 31
    Supervised Binning 32
    Feature Selection and Construction 32
    Principal Component Analysis 33
    Correlation and Partial Correlation 34
Discretization of Continuous Predictors 35
Predictive Power 35
References 36

Bayesian Networks Algorithms 37

Bayesian Networks Algorithm Overview 37
Primary Calculations 37
    Notation 37
    Handling of Continuous Predictors 38
    Feature Selection via Breadth-First Search 38
    Tree Augmented Naïve Bayes Method 40
    Markov Blanket Algorithms 43
    Blank Handling 47
Model Nugget/Scoring 47

Binary Classifier Comparison Metrics 49

C5.0 Algorithms 51

Scoring. 51

Carma Algorithms 53

Overview 53

Deriving Rules 53

Frequent Itemsets 53

Generating Rules. 54

Blank Handling 55

Effect of Options 55

Generated Model/Scoring 56

Predicted Values 56

Confidence 56

Blank Handling 57

C&RT Algorithms 59

Overview of C&RT. 59

Primary Calculations. 59

Frequency and Case Weight Fields 59

Model Parameters. 60

Blank Handling 61

Effect of Options 62

Secondary Calculations 68

Risk Estimates 68

Gain Summary. 69

Generated Model/Scoring 69

Predicted Values 70

Confidence 71

Blank Handling 71

CHAID Algorithms 73

Overview of CHAID . . . 73
Primary Calculations. . . . 73
Frequency and Case Weight Fields . . . 73
Binning of Scale-Level Predictors . . . 74
Model Parameters. . . . 75
Blank Handling . . . 81
Effect of Options . . . 81
Secondary Calculations . . . 82
Risk Estimates . . . 82
Gain Summary . . . 83
Generated Model/Scoring . . . 84
Predicted Values . . . 84
Confidence . . . 85
Blank Handling . . . 85

Cluster Evaluation Algorithms 87

Notation . . . 87
Goodness Measures . . . 87
Data Preparation . . . 88
Basic Statistics . . . 88
Silhouette Coefficient . . . 89
Sum of Squares Error (SSE) . . . 89
Sum of Squares Between (SSB) . . . 89
Predictor Importance . . . 89
References . . . 91

COXREG Algorithms 93

Cox Regression Algorithms. . . . 93
Estimation . . . 93
Estimation of Beta . . . 94
Estimation of the Baseline Function . . . 96
Selection Statistics for Stepwise Methods . . . 97
Score Statistic . . . 98
Wald Statistic . . . 98

LR (Likelihood Ratio) Statistic 98
Conditional Statistic 98
Statistics 99
Initial Model Information 99
Model Information 99
Information for Variables in the Equation 100
Information for the Variables Not in the Equation 101
Survival Table 101
Plots 101
Survival Plot 102
Hazard Plot 102
LML Plot 102
Blank Handling 102
Scoring 102
Blank Handling 102
References 102

Decision List Algorithms *105*

Algorithm Overview 105
Terminology of Decision List Algorithm 105
Main Calculations. 106
Notation 106
Primary Algorithm 106
Decision Rule Algorithm. 107
Decision Rule Split Algorithm. 108
Secondary Measures 111
Blank Handling 111
Generated Model/Scoring 111
Blank Handling 111

DISCRIMINANT Algorithms *113*

Notation 113
Basic Statistics 113
Mean 113
Variances 113
Within-Groups Sums of Squares and Cross-Product Matrix (W) 114
Total Sums of Squares and Cross-Product Matrix (T) 114

- Within-Groups Covariance Matrix 114
- Individual Group Covariance Matrices 114
- Within-Groups Correlation Matrix (R) 114
- Total Covariance Matrix 114
- Univariate F and $\Lambda$for Variable I 114

Rules of Variable Selection 114

- Method = Direct 115
- Stepwise Variable Selection 115
- Ineligibility for Inclusion 115

Computations During Variable Selection 116

- Tolerance 116
- F-to-Remove 116
- F-to-Enter 116
- Wilks’ Lambda for Testing the Equality of Group Means 116
- The Approximate F Test for Lambda (the “overall F”), also known as Rao’s R (Tatsuoka, 1971) 117
- Rao’s V (Lawley-Hotelling Trace) (Rao, 1952; Morrison, 1976) 117
- The Squared Mahalanobis Distance (Morrison, 1976) between groups a and b 117
- The F Value for Testing the Equality of Means of Groups a and b (Smallest F ratio) 117
- The Sum of Unexplained Variations (Dixon, 1973) 117

Classification Functions 117

Canonical Discriminant Functions 118

- Percentage of Between-Groups Variance Accounted for 118
- Canonical Correlation 118
- Wilks’ Lambda 119
- The Standardized Canonical Discriminant Coefficient Matrix D 119
- The Correlations Between the Canonical Discriminant Functions and the Discriminating Variables 119
- The Unstandardized Coefficients 119
- Tests For Equality Of Variance 120

Blank Handling 121

Generated model/scoring 121

- Cross-Validation (Leave-one-out classification) 122
- Blank Handling (discriminant analysis algorithms scoring) 123

References 123

Ensembles Algorithms 125

Bagging and Boosting Algorithms 125

- Notation 125
- Bootstrap Aggregation 126
- Bagging Model Measures 127
- Adaptive Boosting 128

Stagewise Additive Modeling using Multiclass Exponential loss 129
Boosting Model Measures 130
References 130
Very large datasets (pass, stream, merge) algorithms 130
Pass 131
Stream 132
Merge 132
Adaptive Predictor Selection 132
Automatic Category Balancing 133
Model Measures 134
Scoring 136
Ensembling model scores algorithms 136
Notation 136
Scoring 136

Factor Analysis/PCA Algorithms *139*

Overview 139
Primary Calculations. 139
Factor Extraction 139
Factor Rotation 145
Factor Score Coefficients 151
Blank Handling 151
Secondary Calculations 152
Field Statistics and Other Calculations. 152
Generated Model/Scoring 152
Factor Scores 152
Blank Handling 152

Feature Selection Algorithm *153*

Introduction 153
Primary Calculations. 153
Screening 153
Ranking Predictors 154
Selecting Predictors 160
Generated Model 160

GENLIN Algorithms 163

Generalized Linear Models 163
- Notation 163
- Model 163
- Estimation 169
- Model Testing 177
- Blank handling 183
- Scoring 183

References 184

Generalized linear mixed models algorithms 187

Notation 187

Model 188
- Fixed effects transformation 191

Estimation 192
- Linear mixed pseudo model 192
- Iterative process 194
- Wald confidence intervals for covariance parameter estimates 195
- Statistics for estimates of fixed and random effects 196

Testing 199
- Goodness of fit 199
- Tests of fixed effects 199
- Estimated marginal means 200
- Method for computing degrees of freedom 203

Scoring 204

Nominal multinomial distribution 206
- Notation 206
- Model 207
- Estimation 208
- Post-estimation statistics 209
- Testing 211
- Scoring 212

Ordinal multinomial distribution 213
- Notation 213
- Model 214
- Estimation 216
- Post-estimation statistics 217

Testing 218
Scoring 219
References 220

Imputation of Missing Values 223

Imputing Fixed Values. 223
Imputing Random Values 224
Imputing Values Derived from an Expression. 225
Imputing Values Derived from an Algorithm 225

K-Means Algorithm 227

Overview 227
Primary Calculations. 227
Field Encoding 227
Model Parameters. 229
Blank Handling 230
Effect of Options 230
Model Summary Statistics 231
Generated Model/Scoring 231
Predicted Cluster Membership 231
Distances 231
Blank Handling 232

KNN Algorithms 233

Notation 233
Preprocessing 233
Training 234
Distance Metric 234
Crossvalidation for Selection of k 235
Feature Selection 235
Combined k and Feature Selection 236
Blank Handling 236
Output Statistics 237

Scoring 238
Blank Handling 238
References 239

Kohonen Algorithms 241

Overview 241
Primary Calculations. 241
Field Encoding 241
Model Parameters. 242
Blank Handling 244
Effect of Options 244
Generated Model/Scoring 245
Cluster Membership 245
Blank Handling 245

Linear modeling algorithms 247

Notation 247
Model 248
Least squares estimation 248
Model selection 249
Forward stepwise 250
Best subsets 253
Model evaluation 255
Coefficients and statistical inference 256
Scoring 258
Diagnostics 258
Predictor importance 259
References 259

Linear Regression Algorithms 261

Overview 261
Primary Calculations. 261
Notation 261
Model Parameters. 261

Automatic Field Selection 263
Blank Handling 265
Secondary Calculations 265
Model Summary Statistics 265
Field Statistics and Other Calculations 265
Generated Model/Scoring 265
Predicted Values 265
Blank Handling 266

Logistic Regression Algorithms *267*

Logistic Regression Models 267
Multinomial Logistic Regression 267
Primary Calculations 267
Secondary Calculations 272
Stepwise Variable Selection 274
Generated Model/Scoring 279
Binomial Logistic Regression 279
Notation 279
Model 280
Maximum Likelihood Estimates (MLE) 280
Stepwise Variable Selection 281
Statistics 284
Generated Model/Scoring 289

Neural Networks Algorithms *291*

Multilayer Perceptron 291
Notation 291
Architecture 291
Training 294
Radial Basis Function 298
Notation 298
Architecture 299
Training 299
Missing Values 301
Output Statistics 301
Confidence 302
References 302

OPTIMAL BINNING Algorithms 305

Notation 305
Simple MDLP 305
Class Entropy 305
Class Information Entropy 306
Information Gain 306
MDLP Acceptance Criterion 306
Algorithm: BinaryDiscretization 307
Algorithm: MDLPCut 307
Algorithm: SimpleMDLP 308
Hybrid MDLP 308
Algorithm: EqualFrequency 308
Algorithm: HybridMDLP 308
Model Entropy 309
Merging Sparsely Populated Bins 309
Blank Handling 309
References 310

Predictor Importance Algorithms 311

Notation 311
Variance Based Method 311
References 314

QUEST Algorithms 315

Overview of QUEST. 315
Primary Calculations. 315
Frequency Weight Fields 315
Model Parameters. 316
Blank Handling 319
Effect of Options 321
Secondary Calculations 324
Risk Estimates 324
Gain Summary. 325
Generated Model/Scoring 325
Predicted Values 325

Confidence 326
Blank Handling 326

Self-Learning Response Model Algorithms 327

Primary Calculations. 327
Naive Bayes Algorithms. 327
Notation 327
Naive Bayes Model 327
Secondary Calculations 328
Model Assessment 328
Blank Handling 329
Updating the Model 329
Generated Model/Scoring 329
Predicted Values and Confidences. 329
Variable Assessment 330

Sequence Algorithm 333

Overview of Sequence Algorithm 333
Primary Calculations. 333
Itemsets, Transactions, and Sequences 333
Sequential Patterns 335
Adjacency Lattice 336
Mining for Frequent Sequences 337
Generating Sequential Patterns 339
Blank Handling 340
Secondary Calculations 340
Confidence 340
Generated Model/Scoring 341
Predicted Values 341
Confidence 342
Blank Handling 342

Simulation algorithms 343

Simulation algorithms 343
Notation 343

Distribution fitting 343
Goodness of fit measures 352
Anderson-Darling statistic with frequency weights 360
References 360
Simulation algorithms: run simulation 361
Generating correlated data 361
Sensitivity measures 363
References 365

Support Vector Machine (SVM) Algorithms 367

Introduction to Support Vector Machine Algorithms 367
SVM Algorithm Notation 367
SVM Types 367
C-Support Vector Classification (C-SVC) 368
ε-Support Vector Regression (ε-SVR) 368
Primary Calculations. 369
Solving Quadratic Problems. 369
Variable Scale 370
Model Building Algorithm 370
Model Nugget/Scoring 377
Blank Handling 377

Time Series Algorithms 379

Notation 379
Models 379
Exponential Smoothing Models 379
ARIMA and Transfer Function Models 382
Outlier Detection in Time Series Analysis 387
Notation 388
Definitions of Outliers 388
Estimating the Effects of an Outlier 390
Detection of Outliers 390
Goodness-of-Fit Statistics. 391
Mean Squared Error 392
Mean Absolute Percent Error 392
Maximum Absolute Percent Error 392
Mean Absolute Error 392

Maximum Absolute Error 392
Normalized Bayesian Information Criterion 392
R-Squared 392
Stationary R-Squared 392
Expert Modeling 393
Univariate Series 393
Multivariate Series 394
Blank Handling 396
Generated Model/Scoring 396
Blank Handling 396
References 396

TwoStep Cluster Algorithms 397

Overview 397
Model Parameters 397
Pre-cluster 397
Cluster 398
Distance Measure 398
Number of Clusters (auto-clustering) 399
Blank Handling 400
Effect of Options 400
Outlier Handling 400
Generated Model/Scoring 401
Predicted Values 401
Blank Handling 401

Appendix

A Notices 403

Bibliography 407

Index 413

Adjusted Propensities Algorithms

Adjusted propensity scores are calculated as part of the process of building the model, and will not be available otherwise. Once the model is built, it is then scored using data from the test or validation partition, and a new model to deliver adjusted propensity scores is constructed by analyzing the original model's performance on that partition. Depending on the type of model, one of two methods may be used to calculate the adjusted propensity scores.

Model-Dependent Method

For rule set and tree models, the following method is used:

1. Score the model on the test or validation partition.
2. **Tree models.** Calculate the frequency of each category at each tree node using the test/validation partition, reflecting the distribution of the target value in the records scored to that node.

   Rule set models. Calculate the support and confidence of each rule using the test/validation partition, reflecting the model performance on the test/validation partition.

   This results in a new rule set or tree model which is stored with the original model. Each time the original model is applied to new data, the new model can subsequently be applied to the raw propensity scores to generate the adjusted scores.

General Purpose Method

For other models, the following method is used:

1. Score the model on the test or validation partition to compute predicted values and predicted raw propensities.
2. Remove all records which have a missing value for the predicted or observed value.
3. Calculate the observed propensities as 1 for true observed values and 0 otherwise.
4. Bin records according to predicted raw propensity using 100 equal-count tiles.
5. Compute the mean predicted raw propensity and mean observed propensity for each bin.
6. Build a neural network with mean observed propensity as the target and predicted raw propensity as a predictor. For the neural network settings:

```
Use a random seed, value 0
Use the "quick" training method
Stop after 250 cycles
Do not use prevent overtaining option
Use expert mode
Quick Method Expert Options:
  Use one hidden layer with 3 neurons and persistence set to 200
Learning Rates Expert Options:
  Alpha 0.9
```

Initial Eta 0.3
High Eta 0.1
Eta decay 50
Low Eta 0.01

The result is a neural network model that attempts to map raw propensity to a more accurate estimate which takes into account the original model's performance on the testing or validation partition. To calculate adjusted propensities at score time, this neural network is applied to the raw propensities obtained from scoring the original model.

Anomaly Detection Algorithm

Overview

The Anomaly Detection procedure searches for unusual cases based on deviations from the norms of their cluster groups. The procedure is designed to quickly detect unusual cases for data-auditing purposes in the exploratory data analysis step, prior to any inferential data analysis. This algorithm is designed for generic anomaly detection; that is, the definition of an anomalous case is not specific to any particular application, such as detection of unusual payment patterns in the healthcare industry or detection of money laundering in the finance industry, in which the definition of an anomaly can be well-defined.

Primary Calculations

Notation

The following notation is used throughout this chapter unless otherwise stated:

ID — The identity variable of each case in the data file.

n — The number of cases in the training data X_{train} .

X_{ok}, k = 1, …, K — The set of input variables in the training data.

M_k, k ∈ {1, …, K} — If X_{ok} is a continuous variable, M_k represents the grand mean, or average of the variable across the entire training data.

SD_k, k ∈ {1, …, K} — If X_{ok} is a continuous variable, SD_k represents the grand standard deviation, or standard deviation of the variable across the entire training data.

X_{K+1} — A continuous variable created in the analysis. It represents the percentage of variables (k = 1, …, K) that have missing values in each case.

X_k, k = 1, …, K — The set of processed input variables after the missing value handling is applied. For more information, see the topic "Modeling Stage " on p. 4.

H, or the boundaries of H: [H_{min}, H_{max}] — H is the pre-specified number of cluster groups to create. Alternatively, the bounds [H_{min}, H_{max}] can be used to specify the minimum and maximum numbers of cluster groups.

n_h, h = 1, …, H — The number of cases in cluster h, h = 1, …, H, based on the training data.

p_h, h = 1, …, H — The proportion of cases in cluster h, h = 1, …, H, based on the training data. For each h, $p_h = n_h/n$.

M_{hk}, k = 1, …, K+1, h = 1, …, H — If X_k is a continuous variable, M_{hk} represents the cluster mean, or average of the variable in cluster h based on the training data. If X_k is a categorical variable, it represents the cluster mode, or most popular categorical value of the variable in cluster h based on the training data.

SD_{hk}, k ∈ {1, …, K+1}, h = 1, …, H — If X_k is a continuous variable, SD_{hk} represents the cluster standard deviation, or standard deviation of the variable in cluster h based on the training data.

{n_{hkj}}, k ∈ {1, …, K}, h = 1, …, H, j = 1, …, J_k — The frequency set {n_{hkj}} is defined only when X_k is a categorical variable. If X_k has J_k categories, then n_{hkj} is the number of cases in cluster h that fall into category j.

m — An adjustment weight used to balance the influence between continuous and categorical variables. It is a positive value with a default of 6.

VDI_k, k = 1, …, K+1 — The variable deviation index of a case is a measure of the deviation of variable value X_k from its cluster norm.

GDI — The group deviation index GDI of a case is the log-likelihood distance d(h, s), which is the sum of all of the variable deviation indices $\{VDI_k, k = 1, \ldots, K+1\}$.

anomaly index — The anomaly index of a case is the ratio of the GDI to that of the average GDI for the cluster group to which the case belongs.

variable contribution measure — The variable contribution measure of variable X_k for a case is the ratio of the VDI_k to the case's corresponding GDI.

$pct_{anomaly}$ or $n_{anomaly}$ — A pre-specified value $pct_{anomaly}$ determines the percentage of cases to be considered as anomalies. Alternatively, a pre-specified positive integer value $n_{anomaly}$ determines the number of cases to be considered as anomalies.

$cutpoint_{anomaly}$ — A pre-specified cutpoint; cases with anomaly index values greater than $cutpoint_{anomaly}$ are considered anomalous.

$k_{anomaly}$ — A pre-specified integer threshold $1 \le k_{anomaly} \le K+1$ determines the number of variables considered as the reasons that the case is identified as an anomaly.

Algorithm Steps

This algorithm is divided into three stages:

Modeling. Cases are placed into cluster groups based on their similarities on a set of input variables. The clustering model used to determine the cluster group of a case and the sufficient statistics used to calculate the norms of the cluster groups are stored.

Scoring. The model is applied to each case to identify its cluster group and some indices are created for each case to measure the unusualness of the case with respect to its cluster group. All cases are sorted by the values of the anomaly indices. The top portion of the case list is identified as the set of anomalies.

Reasoning. For each anomalous case, the variables are sorted by its corresponding variable deviation indices. The top variables, their values, and the corresponding norm values are presented as the reasons why a case is identified as an anomaly.

Modeling Stage

This stage performs the following tasks:

1. **Training Set Formation.** Starting with the specified variables and cases, remove any case with extremely large values (greater than 1.0E+150) on any continuous variable. If missing value handling is not in effect, also remove cases with a missing value on any variable. Remove variables with all constant nonmissing values or all missing values. The remaining cases and variables are used to create the anomaly detection model. Statistics output to pivot table by the procedure are based on this training set, but variables saved to the dataset are computed for all cases.

2. **Missing Value Handling (Optional).** For each input variable X_{ok}, k = 1, …, K, if X_{ok} is a continuous variable, use all valid values of that variable to compute the grand mean M_k and grand standard deviation SD_k. Replace the missing values of the variable by its grand mean. If X_{ok} is a categorical variable, combine all missing values into a "missing value" category. This category is treated as a valid category. Denote the processed form of $\{X_{ok}\}$ by $\{X_k\}$.

3. **Creation of Missing Value Pct Variable (Optional).** A new continuous variable, X_{K+1}, is created that represents the percentage of variables (both continuous and categorical) with missing values in each case.
4. **Cluster Group Identification.** The processed input variables $\{X_k, k = 1, \ldots, K+1\}$ are used to create a clustering model. The two-step clustering algorithm is used with noise handling turned on (see the TwoStep Cluster algorithm document for more information).
5. **Sufficient Statistics Storage.** The cluster model and the sufficient statistics for the variables by cluster are stored for the Scoring stage:
   - The grand mean M_k and standard deviation SD_k of each continuous variable are stored, $k \in \{1, \ldots, K+1\}$.
   - For each cluster h = 1, …, H, store the size n_h. If X_k is a continuous variable, store the cluster mean M_{hk} and standard deviation SD_{hk} of the variable based on the cases in cluster h. If X_k is a categorical variable, store the frequency n_{hkj} of each category j of the variable based on the cases in cluster h. Also store the modal category M_{hk}. These sufficient statistics will be used in calculating the log-likelihood distance d(h, s) between a cluster h and a given case s.

Scoring Stage

This stage performs the following tasks on scoring (testing or training) data:

1. **New Valid Category Screening.** The scoring data should contain the input variables $\{X_{ok}, k = 1, \ldots, K\}$ in the training data. Moreover, the format of the variables in the scoring data should be the same as those in the training data file during the Modeling Stage.

   Cases in the scoring data are screened out if they contain a categorical variable with a valid category that does not appear in the training data. For example, if *Region* is a categorical variable with categories IL, MA and CA in the training data, a case in the scoring data that has a valid category FL for *Region* will be excluded from the analysis.
2. **Missing Value Handling (Optional).** For each input variable X_{ok}, if X_{ok} is a continuous variable, use all valid values of that variable to compute the grand mean M_k and grand standard deviation SD_k. Replace the missing values of the variable by its grand mean. If X_{ok} is a categorical variable, combine all missing values and put together a missing value category. This category is treated as a valid category.
3. **Creation of Missing Value Pct Variable (Optional depending on Modeling Stage).** If X_{K+1} is created in the Modeling Stage, it is also computed for the scoring data.
4. **Assign Each Case to its Closest Non-Noise Cluster.** The clustering model from the Modeling Stage is applied to the processed variables of the scoring data file to create a cluster ID for each case. Cases belonging to the noise cluster are reassigned to their closest non-noise cluster. See the TwoStep Cluster algorithm document for more information on the noise cluster.
5. **Calculate Variable Deviation Indices.** Given a case s, the closest cluster h is found. The variable deviation index VDI_k of variable X_k is defined as the contribution $d_k(h, s)$ of the variable to its log-likelihood distance d(h, s). The corresponding norm value is M_{hk}, which is the cluster sample mean of X_k if X_k is continuous, or the cluster mode of X_k if X_k is categorical.

6. **Calculate Group Deviation Index.** The group deviation index GDI of a case is the log-likelihood distance d(h, s), which is the sum of all the variable deviation indices $\{VDI_k, k = 1, \ldots, K+1\}$.

7. **Calculate Anomaly Index and Variable Contribution Measures.** Two additional indices are calculated that are easier to interpret than the group deviation index and the variable deviation index.

   The anomaly index of a case is an alternative to the GDI, which is computed as the ratio of the case's GDI to the average GDI of the cluster to which the case belongs. Increasing values of this index correspond to greater deviations from the average and indicate better anomaly candidates.

   A variable's variable contribution measure of a case is an alternative to the VDI, which is computed as the ratio of the variable's VDI to the case's GDI. This is the proportional contribution of the variable to the deviation of the case. The larger the value of this measure, the greater the variable's contribution to the deviation.

Odd Situations

Zero Divided by Zero

The situation in which the GDI of a case is zero and the average GDI of the cluster that the case belongs to is also zero is possible if the cluster is a singleton or is made up of identical cases and the case in question is the same as the identical cases. Whether this case is considered as an anomaly or not depends on whether the number of identical cases that make up the cluster is large or small. For example, suppose that there is a total of 10 cases in the training and two clusters are resulted in which one cluster is a singleton; that is, made up of one case, and the other has nine cases. In this situation, the case in the singleton cluster should be considered as an anomaly as it does not belong to the larger cluster. One way to calculate the anomaly index in this situation is to set it as the ratio of average cluster size to the size of the cluster *h*, which is:

$$\frac{n/H}{n_h}$$

Following the 10 cases example, the anomaly index for the case belonging to the singleton cluster would be (10/2)/1 = 5, which should be large enough for the algorithm to catch it as an anomaly. In this situation, the variable contribution measure is set to 1/(K+1), where (K+1) is the number of processed variables in the analysis.

Nonzero Divided by Zero

The situation in which the GDI of a case is nonzero but the average GDI of the cluster that the case belongs to is 0 is possible if the corresponding cluster is a singleton or is made up of identical cases and the case in question is not the same as the identical cases. Suppose that case *i* belongs to cluster *h*, which has a zero average GDI; that is, $\text{average(GDI)}_h = 0$, but the GDI between case *i* and cluster *h* is nonzero; that is, GDI(i, h) ≠ 0. One choice for the anomaly index calculation of case *i* could be to set the denominator as the weighted average GDI over all other clusters if this value is not 0; else set the calculation as the ratio of average cluster size to the size of cluster *h*. That is,

$$\begin{cases} \frac{GDI(i,h)}{\frac{1}{(n-n_h)}\Sigma_{s=1,\neq h}^{H} n_s \cdot average(GDI)_s} & \text{if } \frac{1}{(n-n_h)} \sum_{s=1,\neq h}^{H} n_s \cdot average(GDI)_s \neq 0 \\ \frac{n/H}{n_h} & \text{otherwise} \end{cases}$$

This situation triggers a warning that the case is assigned to a cluster that is made up of identical cases.

Reasoning Stage

Every case now has a group deviation index and anomaly index and a set of variable deviation indices and variable contribution measures. The purpose of this stage is to rank the likely anomalous cases and provide the reasons to suspect them of being anomalous.

1. **Identify the Most Anomalous Cases.** Sort the cases in descending order on the values of the anomaly index. The top $\text{pct}_{\text{anomaly}}$ % (or alternatively, the top $\text{n}_{\text{anomaly}}$) gives the anomaly list, subject to the restriction that cases with an anomaly index less than or equal to $\text{cutpoint}_{\text{anomaly}}$ are not considered anomalous.

2. **Provide Reasons for Considering a Case Anomalous.** For each anomalous case, sort the variables by their corresponding VDI_{k} values in descending order. The top $\text{k}_{\text{anomaly}}$ variable names, its value (of the corresponding original variable X_{ok}), and the norm values are displayed as reasoning.

Blank Handling

Blanks and missing values are handled in model building as described in “Algorithm Steps ” on p. 4, based on user settings.

Generated Model/Scoring

The Anomaly Detection generated model can be used to detect anomalous records in new data based on patterns found in the original training data. For each record scored, an anomaly score is generated and a flag indicating anomaly status and/or the anomaly score are appended as new fields

Predicted Values

For each record, the anomaly score is calculated as described in “Scoring Stage ” on p. 5, based on the cluster model created when the model was built. If anomaly flags were requested, they are determined as described in “Reasoning Stage ” on p. 7.

Blank Handling

In the generated model, blanks are handled according to the setting used in building the model. For more information, see the topic “Scoring Stage ” on p. 5.

Apriori Algorithms

Overview

Apriori is an algorithm for extracting association rules from data. It constrains the search space for rules by discovering frequent itemsets and only examining rules that are made up of frequent itemsets (Agrawal and Srikant, 1994).

Apriori deals with items and itemsets that make up transactions. **Items** are flag-type conditions that indicate the presence or absence of a particular thing in a specific transaction. An **itemset** is a group of items which may or may not tend to co-occur within transactions.

IBM® SPSS® Modeler uses Christian Borgelt's Apriori implementation. Full details on this implementation can be obtained at *http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/apriori/apriori.html*.

Deriving Rules

Apriori proceeds in two stages. First it identifies frequent itemsets in the data, and then it generates rules from the table of frequent itemsets.

Frequent Itemsets

The first step in Apriori is to identify frequent itemsets. A frequent itemset is defined as an itemset with support greater than or equal to the user-specified minimum support threshold s_{min}. The support of an itemset is the number of records in which the itemset is found divided by the total number of records.

The algorithm begins by scanning the data and identifying the single-item itemsets (i.e. individual items, or itemsets of length 1) that satisfy this criterion. Any single items that do not satisfy the criterion are not be considered further, because adding an infrequent item to an itemset will always result in an infrequent itemset.

Apriori then generates larger itemsets recursively using the following steps:

- Generate a candidate set of itemsets of length *k* (containing *k* items) by combining existing itemsets of length $(k-1)$:

  For every possible pair of frequent itemsets p and q with length $(k-1)$, compare the first $(k-2)$ items (in lexicographic order); if they are the same, and the last item in q is (lexicographically) greater than the last item in p, add the last item in q to the end of p to create a new candidate itemset with length *k*.

- Prune the candidate set by checking every $(k-1)$ length subset of each candidate itemset; all subsets must be frequent itemsets, or the candidate itemset is infrequent and is removed from further consideration.

- Calculate the support of each itemset in the candidate set, as

$$support = \frac{N_i}{N}$$

where N_i is the number of records that match the itemset and *N* is the number of records in the training data. (Note that this definition of itemset support is different from the definition used for rule support.)

- Itemsets with support ≥ s_{min} are added to the list of frequent itemsets.
- If any frequent itemsets of length *k* were found, and *k* is less than the user-specified maximum rule size k_{max}, repeat the process to find frequent itemsets of length $(k+1)$.

Generating Rules

When all frequent itemsets have been identified, the algorithm extracts rules from the frequent itemsets. For each frequent itemset L with length *k* > 1, the following procedure is applied:

- Calculate all subsets A of length $(k-1)$ of the itemset such that all the fields in A are input fields and all the other fields in the itemset (those that are *not* in A) are output fields. Call the latter subset $\tilde{A}$. (In the first iteration this is just one field, but in later iterations it can be multiple fields.)
- For each subset A, calculate the evaluation measure (rule confidence by default) for the rule $A \Rightarrow \tilde{A}$ as described below.
- If the evaluation measure is greater than the user-specified threshold, add the rule to the rule table, and, if the length *k'* of A is greater than 1, test all possible subsets of A with length $(k'-1)$

Evaluation Measures

Apriori offers several evaluation measures for determining which rules to retain. The different measures will emphasize different aspects of the rules, as detailed in the *IBM® SPSS® Modeler User's Guide*. Values are calculated based on the prior confidence and the posterior confidence, defined as

$$C_{prior} = \frac{c}{N}$$

and

$$C_{posterior} = \frac{r}{a}$$

where c is the support of the consequent, a is the support of the antecedent, r is the support of the conjunction of the antecedent and the consequent, and N is the number of records in the training data.

Rule Confidence. The default evaluation measure for rules is simply the posterior confidence of the rule,

$$e = C_{posterior}$$

Confidence Difference (Absolute Confidence Difference to Prior). This measure is based on the simple difference of the posterior and prior confidence values,

$$e = |C_{posterior} - C_{prior}|$$

Confidence Ratio (Difference of Confidence Quotient to 1). This measure is based on the ratio of posterior confidence to prior confidence,

$$e = 1 - min\left(\frac{C_{posterior}}{C_{prior}}, \frac{C_{prior}}{C_{posterior}}\right)$$

Information Difference (Information Difference to Prior). This measure is based on the information gain criterion, similar to that used in building C5.0 trees. The calculation is

$$e = \frac{r \cdot \log\left(\frac{r}{a \cdot c}\right) + (a-r)\log\left(\frac{a-r}{a \cdot \overline{c}}\right) + (c-r)\log\left(\frac{c-r}{\overline{a} \cdot c}\right) + (1-a-c+r)\log\left(\frac{1-a-c+r}{\overline{a} \cdot \overline{c}}\right)}{\log(2)}$$

where *r* is the rule support, *a* is the antecedent support, *c* is the consequent support, $\overline{a} = 1 - a$ is the complement of antecedent support, and $\overline{c} = 1 - c$ is the complement of consequent support.

Normalized Chi-square (Normalized Chi-squared Measure). This measure is based on the chi-squared statistical test for independence of categorical data, and is calculated as

$$e = \frac{(a \cdot c - r)^2}{a \cdot \overline{a} \cdot c \cdot \overline{c}}$$

Blank Handling

Blanks are ignored by the Apriori algorithm. The algorithm will handle records containing blanks for input fields, but such a record will not be considered to match any rule containing one or more of the fields for which it has blank values.

Effect of Options

Minimum rule support/confidence. These values place constraints on which rules may be entered into the table. Only rules whose support and confidence values exceed the specified values can be entered into the rule table.

Maximum number of antecedents. This determines the maximum number of antecedents that will be examined for any rule. When the number of conditions in the antecedent part of the rule equals the specified value, the rule will not be specialized further.

Only true values for flags. If this option is selected, rules with values of *false* will not be considered for either input or output fields.

Optimize Speed/Memory. This option controls the trade-off between speed of processing and memory usage. Selecting Speed will cause Apriori to use condition values directly in the frequent itemset table, and to load the transactions into memory, if possible. Selecting Memory will cause Apriori to use pointers into a value table in the frequent itemset table. Using pointers in the frequent itemset table reduces the amount of memory required by the algorithm for large problems, but it also involves some additional work to reference and dereference the pointers during model building. The Memory option also causes Apriori to process transactions from the file rather than loading them into memory.

Generated Model/Scoring

The Apriori algorithm generates an unrefined rule node. To create a model for scoring new data, the unrefined rule node must be refined to generate a ruleset node. Details of scoring for generated ruleset nodes are given below.

Predicted Values

Predicted values are based on the rules in the ruleset. When a new record is scored, it is compared to the rules in the ruleset. How the prediction is generated depends on the user's setting for Ruleset Evaluation in the stream options.

- **Voting.** This method attempts to combine the predictions of all of the rules that apply to the record. For each record, all rules are examined and each rule that applies to the record is used to generate a prediction. The sum of confidence figures for each predicted value is computed, and the value with the greatest confidence sum is chosen as the final prediction.
- **First hit.** This method simply tests the rules in order, and the first rule that applies to the record is the one used to generate the prediction.

There is a **default rule**, which specifies an output value to be used as the prediction for records that don't trigger any other rules from the ruleset. For rulesets derived from decision trees, the value for the default rule is the modal (most prevalent) output value in the overall training data. For association rulesets, the default value is specified by the user when the ruleset is generated from the unrefined rule node.

Confidence

Confidence calculations also depend on the user's Ruleset Evaluation stream options setting.

- **Voting.** The confidence for the final prediction is the sum of the confidence values for rules triggered by the current record that give the winning prediction divided by the number of rules that fired for that record.
- **First hit.** The confidence is the confidence value for the first rule in the ruleset triggered by the current record.

If the default rule is the only rule that fires for the record, it's confidence is set to 0.5.

Blank Handling

Blanks are ignored by the algorithm. The algorithm will handle records containing blanks for input fields, but such a record will not be considered to match any rule containing one or more of the fields for which it has blank values.

Automated Data Preparation Algorithms

The goal of automated data preparation is to prepare a dataset so as to generally improve the training speed, predictive power, and robustness of models fit to the prepared data.

These algorithms do not assume which models will be trained post-data preparation. At the end of automated data preparation, we output the predictive power of each recommended predictor, which is computed from a linear regression or naïve Bayes model, depending upon whether the target is continuous or categorical.

Notation

The following notation is used throughout this chapter unless otherwise stated:

X A continuous or categorical variable

x_i Value of the variable X for case i.

f_i Frequency weight for case i. Non-integer positive values are rounded to the nearest integer. If there is no frequency weight variable, then all $f_i = 1$. If the frequency weight of a case is zero, negative or missing, then this case will be ignored.

w_i Analysis weight for case i. If there is no analysis weight variable, then all $w_i = 1$. If the analysis weight of a case is zero, negative or missing, then this case will be ignored.

n Number of cases in the dataset

N_X $\sum_{i=1}^{n} f_i I\left(x_i \text{ is not missing}\right)$, where I (expression) is the indicator function taking value 1 when the expression is true, 0 otherwise.

W_X $\sum_{i=1}^{n} f_i w_i I\left(x_i \text{ is not missing}\right)$

N_{XY} $$\sum_{i=1}^{n} f_i I\left(x_i \text{ and } y_i \text{ are not missing}\right)$$

W_{XY} $$\sum_{i=1}^{n} f_i w_i I\left(x_i \text{ and } y_i \text{ are not missing}\right)$$

$\overline{x}$ The mean of variable X, $\frac{1}{W_X}\sum_{i=1}^{n} f_i w_i x_i I\left(x_i \quad \text{is not missing}\right)$

M_X^r $$\sum_{i=1}^{n} f_i w_i (x_i - \overline{x})^r$$

$\overline{x}_y$ $$\frac{1}{W_{XY}}\sum_{i=1}^{n} f_i w_i x_i I\left(x_i \text{ and } y_i \text{ are not missing}\right)$$

M_{XY} $$\sum_{i=1}^{n} f_i w_i \left(x_i - \overline{x}_y\right)\left(y_i - \overline{y}_x\right)$$

A note on missing values

Listwise deletion is used in the following sections:

- "Univariate Statistics Collection " on p. 15

- "Basic Variable Screening " on p. 17
- "Measurement Level Recasting " on p. 17
- "Missing Value Handling " on p. 19
- "Outlier Identification and Handling " on p. 18
- "Continuous Predictor Transformations " on p. 20
- "Target Handling " on p. 21
- "Reordering Categories " on p. 25
- "Unsupervised Merge " on p. 30

Pairwise deletion is used in the following sections:

- "Bivariate Statistics Collection " on p. 22
- "Supervised Merge " on p. 26
- "Supervised Binning " on p. 32
- "Feature Selection and Construction " on p. 32
- "Predictive Power " on p. 35

A note on frequency weight and analysis weight

The frequency weight variable is treated as a case replication weight. For example if a case has a frequency weight of 2, then this case will count as 2 cases.

The analysis weight would adjust the variance of cases. For example if a case x_i of a variable X has an analysis weight w_i, then we assume that $x_i \sim N\left(\mu, \frac{\sigma^2}{w_i}\right)$.

Frequency weights and analysis weights are used in automated preparation of other variables, but are themselves left unchanged in the dataset.

Date/Time Handling

Date Handling

If there is a date variable, we extract the date elements (year, month and day) as ordinal variables. If requested, we also calculate the number of elapsed days/months/years since the user-specified reference date (default is the current date). Unless specified by the user, the "best" unit of duration is chosen as follows:

1. If the minimum number of elapsed days is less than 31, then we use days as the best unit.
2. If the minimum number of elapsed days is less than 366 but larger than or equal to 31, we use months as the best unit. The number of months between two dates is calculated based on average number of days in a month (30.4375): *months* = *days* / 30.4375.
3. If the minimum number of elapsed days is larger than or equal to 366, we use years as the best unit. The number of years between two dates is calculated based on average number of days in a year (365.25): *years* = *days* / 365.25.

Once the date elements are extracted and the duration is obtained, then the original date variable will be excluded from the rest of the analysis.

Time Handling

If there is a time variable, we extract the time elements (second, minute and hour) as ordinal variables. If requested, we also calculate the number of elapsed seconds/minutes/hours since the user-specified reference time (default is the current time). Unless specified by the user, the "best" unit of duration is chosen as follows:

1. If the minimum number of elapsed seconds is less than 60, then we use seconds as the best unit.
2. If the minimum number of elapsed seconds is larger than or equal to 60 but less than 3600, we use minutes as the best unit.
3. If the minimum number of elapsed seconds is larger than or equal to 3600, we use hours as the best unit.

Once the elements of time are extracted and time duration is obtained, then original time predictor will be excluded.

Univariate Statistics Collection

Continuous Variables

For each continuous variable, we calculate the following statistics:

- Number of missing values: $N_X^{missing} = \sum_{i=1}^{n} f_i I\left(x_i \text{ is missing}\right)$
- Number of valid values: N_X
- Minimum value: $\min_i x_i$
- Maximum value: $\max_i x_i$
- Mean, standard deviation, skewness. (see below)
- The number of distinct values *I*.
- The number of cases for each distinct value s_i: $c_i = \sum_{j=1}^{n} f_j I\left(x_j = s_i\right)$
- Median: If the distinct values of *X* are sorted in ascending order, $s_1 < s_2 < \cdots < s_I$, then the median can be computed by $Median\left(X\right) = \min\left\{s_i : \frac{cc_i}{N_X} \geq 0.5\right\}$, where $cc_i = \sum_{j=1}^{i} c_i$.

Note: If the number of distinct values is larger than a threshold (default is 5), we stop updating the number of distinct values and the number of cases for each distinct value. Also we do not calculate the median.

Categorical Numeric Variables

For each categorical numeric variable, we calculate the following statistics:

- Number of missing values: $N_X^{missing} = \sum_{i=1}^{n} f_i I\left(x_i \text{ is missing}\right)$

- Number of valid values: N_X
- Minimum value: $\min_i x_i$ (only for ordinal variables)
- Maximum value: $\max_i x_i$ (only for ordinal variables)
- The number of categories.
- The counts of each category.
- Mean, Standard deviation, Skewness (only for ordinal variables). (see below)
- Mode (only for nominal variables). If several values share the greatest frequency of occurrence, then the mode with the smallest value is used.
- Median (only for ordinal variables): If the distinct values of X are sorted in ascending order, $s_1 < s_2 < \cdots < s_I$, then the median can be computed by $Median\left(X\right) = \min\left\{s_i : \frac{cc_i}{N_X} \geq 0.5\right\}$, where $cc_i = \sum_{j=1}^{i} c_i$.

Notes:

1. If an ordinal predictor has more categories than a specified threshold (default 10), we stop updating the number of categories and the number of cases for each category. Also we do not calculate mode and median.
2. If a nominal predictor has more categories than a specified threshold (default 100), we stop collecting statistics and just store the information that the variable had more than threshold categories.

Categorical String Variables

For each string variable, we calculate the following statistics:

- Number of missing values: $N_X^{missing} = \sum_{i=1}^{n} f_i I\left(x_i \text{ is missing}\right)$
- Number of valid values: N_X
- The number of categories.
- Counts of each category.
- Mode: If several values share the greatest frequency of occurrence, then the mode with the smallest value is used.

Note: If a string predictor has more categories than a specified threshold (default 100), we stop collecting statistics and just store the information that the predictor had more than threshold categories.

Mean, Standard Deviation, Skewness

We calculate mean, standard deviation and skewness by updating moments.

1. Start with $N_X^{(0)} = W_X^{(0)} = \overline{x}^{(0)} = M_X^{2(0)} = M_X^{3(0)} = 0$.

2. For *j*=1,..,*n* compute:
   $N_X^{(j)} = N_X^{(j-1)} + f_j I\left(x_j \text{ is not missing}\right)$

$$W_X^{(j)} = W_X^{(j-1)} + f_j w_i I\left(x_j \text{ is not missing}\right)$$
$$v_j = \frac{f_j w_j}{W_X^{(j)}}\left(x_j - \overline{x}^{(j-1)}\right)$$
$$\overline{x}^{(j)} = \overline{x}^{(j-1)} + v_j$$
$$M_X^{2(j)} = M_X^{2(j-1)} + \frac{W_X^{(j)} W_X^{(j-1)}}{f_j w_j} v_j^2$$
$$M_X^{3(j)} = M_X^{3(i-1)} - 3 v_j M_X^{2(j-1)} + \frac{W_X^{(j)} W_X^{(j-1)}}{(f_j w_j)^2}\left(W_X^{(j)} - 2 f_j w_j\right) v_j^3$$

3. After the last case has been processed, compute:

   Mean: $\overline{x} = \overline{x}^{(n)}$

   Standard deviation: $sd = \sqrt{\frac{M_X^{2(n)}}{N_X - 1}}$

   Skewness: $skew = \frac{\frac{N_X}{(N_X-2)}\frac{1}{(N_X-1)}M_X^{3(n)}}{sd^3}$

If $N_X \leq 2$ or $sd^2 < 10^{-20}$, then skewness is not calculated.

Basic Variable Screening

1. If the percent of missing values is greater than a threshold (default is 50%), then exclude the variable from subsequent analysis.
2. For continuous variables, if the maximum value is equal to minimum value, then exclude the variable from subsequent analysis.
3. For categorical variables, if the mode contains more cases than a specified percentage (default is 95%), then exclude the variable from subsequent analysis.
4. If a string variable has more categories than a specified threshold (default is 100), then exclude the variable from subsequent analysis.

Checkpoint 1: Exit?

This checkpoint determines whether the algorithm should be terminated. If, after the screening step:

1. The target (if specified) has been removed from subsequent analysis, or
2. All predictors have been removed from subsequent analysis,

then terminate the algorithm and generate an error.

Measurement Level Recasting

For each continuous variable, if the number of distinct values is less than a threshold (default is 5), then it is recast as an ordinal variable.

For each numeric ordinal variable, if the number of categories is greater than a threshold (default is 10), then it is recast as a continuous variable.

Note: The continuous-to-ordinal threshold must be less than the ordinal-to-continuous threshold.

Outlier Identification and Handling

In this section, we identify outliers in continuous variables and then set the outlying values to a cutoff or to a missing value. The identification is based on the robust mean and robust standard deviation which are estimated by supposing that the percentage of outliers is no more than 5%.

Identification

1. Compute the mean and standard deviation from the raw data. Split the continuous variable into non-intersecting intervals: $I_i = (\overline{x} + (i-1) \times sd_w, \overline{x} + i \times sd_w], i = -3, -2, \cdots, 2, 3, 4$, where $I_{-3} = (-\infty, \overline{x} - 3sd_w]$, $I_4 = (\overline{x} + 3sd_w, +\infty]$ and $sd_w = sd \times \sqrt{\frac{N_X - 1}{W_X - 1}}$.
2. Calculate univariate statistics in each interval:

   $N_{I_i} = \sum_{j=1}^{n} f_j I(x_j \in I_i)$, $W_{I_i} = \sum_{i=1}^{n} f_i w_i I(x_j \in I_i)$

   $\overline{x}_{I_i} = \frac{\sum_{j=1}^{n} f_j w_j x_j I(x_j \in I_i)}{W_{I_i}}$, $M_{I_i}^2 = \sum_{j=1}^{n} f_j w_j (x_j - \overline{x}_{I_i})^2 I(x_j \in I_i)$
3. Let $l = -3$, $r = 4$, and $p = 0$.
4. Between two tail intervals I_l and I_r, find one interval with the least number of cases.
5. If $N_{I_l} \leq N_{I_r}$, then $p_{current} = \frac{N_{I_l}}{N_X}$. Check if $p + p_{current}$ is less than a threshold $p_{threshold}$ (default is 0.05). If it does, then $p = p + p_{current}$ and $l = l + 1$, go to step 4; otherwise, go to step 6.

   Else $p_{current} = \frac{N_{I_r}}{N_X}$. Check if $p + p_{current}$ is less than a threshold, $p_{threshold}$. If it is, then $p = p + p_{current}$ and $r = r - 1$, go to step 4; otherwise, go to step 6.
6. Compute the robust mean $\overline{x}_{robust}$ and robust standard deviation sd_{robust} within the range $(\overline{x} + (l-1) \times sd, \overline{x} + r \times sd]$. See below for details.
7. If x_i satisfies the conditions:

   $\sqrt{w_i}(x_i - \overline{x}_{robust}) < -cutoff \times sd_{robust}$ or $\sqrt{w_i}(x_i - \overline{x}_{robust}) > cutoff \times sd_{robust}$

   where *cutoff* is positive number (default is 3), then x_i is detected as an outlier.

Handling

Outliers will be handled using one of following methods:

- Trim outliers to cutoff values. If $\sqrt{w_i}(x_i - \overline{x}_{robust}) < -cutoff \times sd_{robust}$ then replace x_i by $\overline{x}_{robust} - cutoff \times sd_{robust}/\sqrt{w_i}$, and if $\sqrt{w_i}(x_i - \overline{x}_{robust}) > cutoff \times sd_{robust}$ then replace x_i by $\overline{x}_{robust} + cutoff \times sd_{robust}/\sqrt{w_i}$.
- Set outliers to missing values.

Update Univariate Statistics

After outlier handling, we perform a data pass to calculate univariate statistics for each continuous variable, including the number of missing values, minimum, maximum, mean, standard deviation, skewness, and number of outliers.

Robust Mean and Standard Deviation

Robust mean and standard deviation within the range $(\overline{x} + (l-1) \times sd, \overline{x} + r \times sd]$ are calculated as follows:

$$\overline{x}_{robust} = \frac{\sum_{i=l}^{r} W_{I_i} \overline{x}_{I_i}}{\sum_{i=l}^{r} W_{I_i}}$$

and

$$sd_{robust} = \sqrt{\frac{M_{robust}^2}{\sum_{i=l}^{r} N_{I_i} - 1}}$$

where $M_{robust}^2 = \sum_{i=l}^{r} A_{I_i}$ and $A_{I_i} = M_{I_i}^2 + W_{I_i}(\overline{x}_{robust} - \overline{x}_{I_i})^2$.

Missing Value Handling

Continuous variables. Missing values are replaced by the mean, and the following statistics are updated:

- Standard deviation: $sd \times \sqrt{\frac{N_X - 1}{N-1}}$, where $N = N_X + N_X^{missing}$.
- Skewness: $skew \times \frac{L_1}{L_2}$, where $L_1 = \left(\frac{N}{N-2}\right)\left(\frac{N_X - 2}{N_X}\right)$ and $L_2 = \sqrt{\frac{N_X - 1}{N-1}}$
- The number of missing values: $N_X^{missing} = 0$
- The number of valid values: $N_X = N$

Ordinal variables. Missing values are replaced by the median, and the following statistics are updated:

- The number of cases in the median category: $c_{median} + N_X^{missing}$, where c_{median} is the original number of cases in the median category.
- The number of missing values: $N_X^{missing} = 0$
- The number of valid values: $N_X = N$

Nominal variables. Missing values are replaced by the mode, and the following statistics are updated:

- The number of cases in the modal category: $c_{mode} + N_X^{missing}$, where c_{mode} is the original number of cases in the modal category.
- The number of missing values: $N_X^{missing} = 0$
- The number of valid values: $N_X = N$

Continuous Predictor Transformations

We transform a continuous predictor so that it has the user-specified mean $\overline{x}_{user}$ (default 0) and standard deviation sd_{user} (default 1) using the z-score transformation, or minimum $\min_{user}$ (default 0) and maximum $\max_{user}$ (default 100) value using the min-max transformation.

Z-score Transformation

Suppose a continuous variable has mean $\overline{x}$ and standard deviation *sd*. The *z*-score transformation is

$$x_i^{'} = \frac{sd_{user}}{sd} \times (x_i - \overline{x}) + \overline{x}_{user}$$

where $x_i^{'}$ is the transformed value of continuous variable *X* for case *i*.

Since we do not take into account the analysis weight in the rescaling formula, the rescaled values $x_i^{'}$ follow a normal distribution $N\left(\overline{x}_{user}, \frac{sd_{user}^2}{w_i}\right)$.

Update univariate statistics

After a z-score transformation, the following univariate statistics are updated:

- Number of missing values: $N_{X'}^{missing} = N_X^{missing}$
- Number of valid values: $N_{X'} = N_X$
- Minimum value: $\min\left(x_i^{'}\right) = \frac{sd_{user}}{sd} \times (\min x_i - \overline{x}) + \overline{x}_{user}$
- Maximum value: $\max\left(x_i^{'}\right) = \frac{sd_{user}}{sd} \times (\max x_i - \overline{x}) + \overline{x}_{user}$
- Mean: $\overline{x}^{'} = \overline{x}_{user}$
- Standard deviation: $sd\left(x^{'}\right) = sd_{user}$
- Skewness: $skew\left(x^{'}\right) = skew\,(x)$

Min-Max Transformation

Suppose a continuous variable has a minimum value $\min x_i$ and a minimum value $\max x_i$. The min-max transformation is

$$x_i^{'} = \frac{\max_{user} - \min_{user}}{\max x_i - \min x_i} \times (x_i - \min x_i) + \min_{user}$$

where $x_i^{'}$ is the transformed value of continuous variable *X* for case *i*.

Update univariate statistics

After a min-max transformation, the following univariate statistics are updated:

- The number of missing values: $N_{X'}^{missing} = N_X^{missing}$

- The number of valid values: $N_{X'} = N_X$
- Minimum value: $\min\left(x_i^{'}\right) = \min_{user}$
- Maximum value: $\max\left(x_i^{'}\right) = \max_{user}$
- Mean: $\overline{x}^{'} = \frac{\max_{user} - \min_{user}}{\max x_i - \min x_i} \times (\overline{x} - \min x_i) + \min_{user}$
- Standard deviation: $sd\left(x^{'}\right) = \frac{\max_{user} - \min_{user}}{\max x_i - \min x_i} \times sd$
- Skwness: $skew\left(x^{'}\right) = skew\,(x)$

Target Handling

Nominal Target

For a nominal target, we rearrange categories from lowest to highest counts. If there is a tie on counts, then ties will be broken by ascending sort or lexical order of the data values.

Continuous Target

The transformation proposed by Box and Cox (1964) transforms a continuous variable into one that is more normally distributed. We apply the Box-Cox transformation followed by the *z* score transformation so that the rescaled target has the user-specified mean and standard deviation.

Box-Cox transformation. This transforms a non-normal variable *Y* to a more normally distributed variable:

$$g_i(\lambda) = g(y_i, \lambda) = \begin{cases} \frac{\left((y_i - c)^{\lambda} - 1\right)}{\lambda} & \lambda \neq 0 \\ \ln(y_i - c) & \lambda = 0 \end{cases}$$

where $y_i, i = 1, 2, \cdots, n$ are observations of variable *Y*, and *c* is a constant such that all values $y_i - c$ are positive. Here, we choose $c = \min(Y) - 1$.

The parameter λ is selected to maximize the log-likelihood function:

$$L(\lambda) = -\frac{N_Y}{2} \ln\left[\frac{N_Y - 1}{N_Y} (sd(g(\lambda)))^2\right] + (\lambda - 1) \sum_{i=1}^{n} f_i \ln(y_i - c)$$

where $(sd(g(\lambda)))^2 = \frac{1}{N_Y - 1} \sum_{i=1}^{n} f_i w_i (g_i(\lambda_j) - \overline{g}(\lambda_j))^2$ and $\overline{g}(\lambda) = \frac{1}{W_Y} \sum_{i=1}^{n} f_i w_i g_i(\lambda)$.

We perform a grid search over a user-specified finite set [*a*,*b*] with increment *s*. By default *a*=−3, *b*=3, and *s*=0.5.

The algorithm can be described as follows:

1. Compute $\lambda_j = a + (j - 1) * s$ where *j* is an integer such that $a \leq \lambda_j \leq b$.

2. For each λ_j, compute the following statistics:

   Mean: $\overline{g}\left(\lambda_j\right) = \frac{1}{W_Y}\sum_{i=1}^{n} f_i w_i g_i(\lambda_j)$

   Standard deviation: $sd\left(g\left(\lambda_j\right)\right) = \sqrt{\frac{1}{N_Y-1}\sum_{i=1}^{n} f_i w_i \left(g_i\left(\lambda_j\right) - \overline{g}\left(\lambda_j\right)\right)^2}$

   Skewness: $skew\left(g\left(\lambda_j\right)\right) = \frac{\frac{N_Y}{(N_Y-2)}\frac{1}{(N_Y-1)}\sum_{i=1}^{n} f_i w_i \left(g_i(\lambda_j) - \overline{g}(\lambda_j)\right)^3}{sd(g(\lambda_j))^3}$

   Sum of logarithm transformation: $\sum_{i=1}^{n} f_i \ln\left(y_i - c\right)$

3. For each λ_j, compute the log-likelihood function $L\left(\lambda_j\right)$. Find the value of *j* with the largest log-likelihood function, breaking ties by selecting the smallest value of λ_j. Also find the corresponding statistics $\overline{g}\left(\lambda^*\right)$, $sd\left(g\left(\lambda^*\right)\right)$ and $skew\left(g\left(\lambda^*\right)\right)$.

4. Transform target to reflect user's mean $\overline{y}_{user}$ (default is 0) and standard deviation sd_{user} (default is 1):

$$y_i^{'} = \frac{sd_{user}}{sd\left(g\left(\lambda^*\right)\right)} \times \left(g_i\left(\lambda^*\right) - \overline{g}\left(\lambda^*\right)\right) + \overline{y}_{user}$$

where $\overline{g}\left(\lambda^*\right) = \frac{1}{W_Y}\sum_{i=1}^{n} f_i w_i g_i(\lambda^*)$ and $sd\left(g\left(\lambda^*\right)\right) = \sqrt{\frac{1}{N_Y-1}\sum_{i=1}^{n} f_i w_i \left(g_i\left(\lambda^*\right) - \overline{g}\left(\lambda^*\right)\right)^2}$.

Update univariate statistics. After Box-Cox and Z-score transformations, the following univariate statistics are updated:

- Minimum value: $\frac{sd_{user}}{sd(g(\lambda^*))} \times \left(g\left(\min\left(y_i\right) - c, \lambda^*\right) - \overline{g}\left(\lambda^*\right)\right) + \overline{y}_{user}$
- Maximum value: $\frac{sd_{user}}{sd(g(\lambda^*))} \times \left(g\left(\max\left(y_i\right) - c, \lambda^*\right) - \overline{g}\left(\lambda^*\right)\right) + \overline{y}_{user}$
- Mean: $\overline{y}_{user}$
- Standard deviation: sd_{user}
- Skewness: $skew\left(g\left(\lambda^*\right)\right)$

Bivariate Statistics Collection

For each target/predictor pair, the following statistics are collected according to the measurement levels of the target and predictor.

Continuous target or no target and all continuous predictors

If there is a continuous target and some continuous predictors, then we need to calculate the covariance and correlations between all pairs of continuous variables. If there is no continuous target, then we only calculate the covariance and correlations between all pairs of continuous predictors. We suppose there are there are *m* continuous variables, and denote the covariance matrix as $C_{m\times m}$, with element c_{ij}, and the correlation matrix as $R_{m\times m}$, with element r_{ij}.

We define the covariance between two continuous variables *X* and *Y* as

$$c_{XY} = \frac{1}{N_{XY} - 1}\sum_{i=1}^{n} f_i w_i \left(x_i - \overline{x}_y\right)\left(y_i - \overline{y}_x\right)$$

where $\overline{x}_y = \frac{1}{W_{XY}} \sum_{i=1}^{n} x_i I\left(x_i \text{ and } y_i \text{ are not missing}\right)$ and
$\overline{y}_x = \frac{1}{W_{XY}} \sum_{i=1}^{n} y_i I\left(x_j \text{ and } y_j \text{ are not missing}\right)$.

The covariance can be computed by a provisional means algorithm:

1. Start with $N_{XY}^{(0)} = W_{XY}^{(0)} = \overline{x}_y = \overline{y}_x = M_{XY}^{(0)} = 0$.
2. For *j*=1,..,*n* compute:

   $N_{XY}^{(j)} = N_{XY}^{(j-1)} + f_j I\left(x_j \text{ and } y_j \text{ are not missing}\right)$

   $W_{XY}^{(j)} = W_{XY}^{(j-1)} + f_j w_j I\left(x_j \text{ and } y_j \text{ are not missing}\right)$

   $v_{xj} = \frac{f_j w_j}{W_{XY}^{(j)}}\left(x_j - \overline{x}_y\right)$

   $\overline{x}_y = \overline{x}_y + v_{xj}$

   $v_{yj} = \frac{f_j w_j}{W_{XY}^{(j)}}\left(y_j - \overline{y}_x\right)$

   $\overline{y}_x = \overline{y}_x + v_{yj}$

   $M_{XY}^{(j)} = M_{XY}^{(j-1)} + \left(x_j - \overline{x}_y\right)\left(y_j - \overline{y}_x\right)\left(f_j w_j - \frac{\left(f_j w_j\right)^2}{W_{XY}^{(j)}}\right)$

   After the last case has been processed, we obtain:

   $M_{XY} = M_{XY}^{(n)} = \sum_{i=1}^{n} f_i w_i \left(x_i - \overline{x}_y\right)\left(y_i - \overline{y}_x\right)$
3. Compute bivariate statistics between *X* and *Y*:

   Number of valid cases: N_{XY}

   Covariance: $c_{XY} = \frac{M_{XY}}{N_{XY}-1}$

   Correlation: $r_{XY} = \frac{c_{XY}}{\sqrt{c_{XX}}\sqrt{c_{YY}}}$

   Note: If there are no valid cases when pairwise deletion is used, then we let $c_{XY} = 0$ and $r_{XY} = 0$.

Categorical target and all continuous predictors

For a categorical target *Y* with values $i = 1, 2, \cdots, J$ and a continuous predictor *X* with values $x_1, \cdots x_n$, the bivariate statistics are:

Mean of *X* for each *Y*=*i*, *i*=1,...,*J*:

$$\overline{x}_{\cdot i} = \frac{\sum_{j=1}^{n} f_j w_j x_j I\left(y_j = i\right)}{\sum_{j=1}^{n} f_j w_j I\left(y_j = i\right)}$$

Sum of squared errors of *X* for each *Y*=*i*, *i*=1,...,*J*:

$$M_{\cdot i}^2 = \sum_{j=1}^{n} f_j w_j (x_j - \overline{x}_{\cdot\ i})^2 I\left(y_j = i\right)$$

Sum of frequency weight for each *Y*=*i*, *i*=1,...,*J*:

$$N_{\cdot i} = \sum_{j=1}^{n} f_j I\left(y_j = i \wedge x_j \text{ is not missing}\right)$$

Number of invalid cases

$$N_{XY} = \sum_{i=1}^{J} N_{\cdot i}$$

Sum of weights (frequency weight times analysis weight) for each *Y*=*i*, *i*=1,...,*J*:

$$W_{\cdot i} = \sum_{j=1}^{n} f_j w_i I\left(y_j = i \wedge x_j \text{ is not missing}\right)$$

Continuous target and all categorical predictors

For a continuous target *Y* and a categorical predictor *X* with values *i*=1,...,*J*, the bivariate statistics include:

Mean of *Y* conditional upon *X*:

$$\overline{y}_x = \frac{\sum_{i=1}^{I} \sum_{j=1}^{n} f_j w_j y_j I\left(x_j = i\right)}{\sum_{i=1}^{I} \sum_{j=1}^{n} f_j w_j I\left(x_j = i\right)}$$

Sum of squared errors of *Y*:

$$M_{X\cdot}^2 = \sum_{j=1}^{n} f_j w_j (y_j - \overline{y}_x)^2$$

Mean of *Y* for each $X = i$, *i*=1,...,*J*:

$$\overline{y}_{i\cdot} = \frac{\sum_{j=1}^{n} f_j w_j y_j I\left(x_j = i\right)}{\sum_{j=1}^{n} f_j w_j I\left(x_j = i\right)}$$

Sum of squared errors of *Y* for each $X = i$, *i*=1,...,*J*:

$$M_{i\cdot}^2 = \sum_{j=1}^{n} f_j w_j (y_j - \overline{y}_{i\cdot})^2 I\left(x_j = i\right)$$

Sum of frequency weights for $X = i$, *i*=1,...,*J*:

$$N_{i\cdot} = \sum_{j=1}^{n} f_j I\left(x_j = i \wedge y_j \text{ is not missing}\right)$$

Sum of weights (frequency weight times analysis weight) for $X = i$, *i*=1,...,*J*:

$$W_{i\cdot} = \sum_{j=1}^{n} f_j w_j I\left(x_j = i \wedge y_j \text{ is not missing}\right)$$

Categorical target and all categorical predictors

For a categorical target *Y* with values *j*=1,...,*J* and a categorical predictor *X* with values *i*=1,...,*I*, then bivariate statistics are:

Sum of frequency weights for each combination of $x_k = i$ and $y_k = j$:

$$N_{ij} = \sum_{k=1}^{n} f_k I\left(x_k = i \wedge y_k = j\right)$$

Sum of weights (frequency weight times analysis weight) for each combination of $x_k = i$ and $y_k = j$:

$$W_{ij} = \sum_{k\in 1}^{n} f_k w_k I\left(x_k = i \wedge y_k = j\right)$$

Categorical Variable Handling

In this step, we use univariate or bivariate statistics to handle categorical predictors.

Reordering Categories

For a nominal predictor, we rearrange categories from lowest to highest counts. If there is a tie on counts, then ties will be broken by ascending sort or lexical order of the data values. The new field values start with 0 as the least frequent category. Note that the new field will be numeric even if the original field is a string. For example, if a nominal field's data values are "A", "A", "A", "B", "C", "C", then automated data preparation would recode "B" into 0, "C" into 1, and "A" into 2.

Identify Highly Associated Categorical Features

If there is a target in the data set, we select a ordinal/nominal predictor if its *p*-value is not larger than an alpha-level $\alpha_{selection}$ (default is 0.05). See "P-value Calculations " on p. 27 for details of computing these *p*-values.

Since we use pairwise deletion to handle missing values when we collect bivariate statistics, we may have some categories with zero cases; that is, $N_{i\cdot} = 0$ for a category *i* of a categorical predictor. When we calculate *p*-values, these categories will be excluded.

If there is only one category or no category after excluding categories with zero cases, we set the *p*-value to be 1 and this predictor will not be selected.

Supervised Merge

We merge categories of an ordinal/nominal predictor using a supervised method that is similar to a Chaid Tree with one level of depth.

1. Exclude all categories with zero case count.

2. If *X* has 0 categories, merge all excluded categories into one category, then stop.

3. If *X* has 1 category, go to step 7.

4. Else, find the allowable pair of categories of *X* that is most similar. This is the pair whose test statistic gives the largest *p*-value with respect to the target. An allowable pair of categories for an ordinal predictor is two adjacent categories; for a nominal predictor it is any two categories. Note that for an ordinal predictor, if categories between the *i*th category and *j*th categories are excluded because of zero cases, then the *i*th category and *j*th categories are two adjacent categories. See "P-value Calculations " on p. 27 for details of computing these *p*-values.

5. For the pair having the largest *p*-value, check if its *p*-value is larger than a specified alpha-level $\alpha_{selection}$ (default is 0.05). If it does, this pair is merged into a single compound category and at the same time we calculate the bivariate statistics of this new category. Then a new set of categories of *X* is formed. If it does not, then go to step 6.

6. Go to step 3.

7. For an ordinal predictor, find the maximum value in each new category. Sort these maximum values in ascending order. Suppose we have *r* new categories, and the maximum values are: $i_1 < i_2 < \cdots < i_r$, then we get the merge rule as: the first new category will contain all original categories such that $X \leq i_1$, the second new category will contain all original categories such that $i_1 < X \leq i_2$,..., and the last new category will contain all original categories such that $X > i_{r-1}$.

   For a nominal predictor, all categories excluded at step 1 will be merged into the new category with the lowest count. If there are ties on categories with the lowest counts, then ties are broken by selecting the category with the smallest value by ascending sort or lexical order of the original category values which formed the new categories with the lowest counts.

Bivariate statistics calculation of new category

When two categories are merged into a new category, we need to calculate the bivariate statistics of this new category.

Scale target. If the categories *i* and $i^{'}$ can be merged based on *p*-value, then the bivariate statistics should be calculated as:

$$N_{i,i'}\,. = N_i\,. + N_{i'}\,.$$

$$W_{i,i'}\,. = W_i\,. + W_{i'}\,.$$

$$\overline{y}_{i,i'}\,. = \overline{y}_i\,. + \frac{W_{i'}\,.}{W_{i,i'}\,.}\left(\overline{y}_{i'}\,. - \overline{y}_i\,.\right)$$

$$M^2_{i,i'}\,. = M^2_i. + M^2_{i'}\,. + W_i\,.\left(\overline{y}_{i,i'}\,. - \overline{y}_i\,.\right)^2 + W_{i'}\,.\left(\overline{y}_{i,i'}\,. - \overline{y}_{i'}\,.\right)^2$$

Categorical target. If the categories *i* and $i^{'}$ can be merged based on *p*-value, then the bivariate statistics should be calculated as:

$$N_{i,i'j} = N_{ij} + N_{i'j}$$

$$W_{i,i'j} = W_{ij} + W_{i'j}$$

Update univariate and bivariate statistics

At the end of the supervised merge step, we calculate the bivariate statistics for each new category. For univariate statistics, the counts for each new category will be sum of the counts of each original categories which formed the new category. Then we update other statistics according to the formulas in "Univariate Statistics Collection " on p. 15, though note that the statistics only need to be updated based on the new categories and the numbers of cases in these categories.

P-value Calculations

Each *p*-value calculation is based on the appropriate statistical test of association between the predictor and target.

Scale target

We calculate an *F* statistic:

$$F = \frac{\sum_{i=1}^{I} W_{i\cdot}\left(\overline{y}_{i\cdot} - \overline{y}_x\right)^2 / \left(I-1\right)\Big)}{\sum_{i=1}^{I} M^2_{i\cdot} / \left(\sum_{i=1}^{I} N_{i\cdot} - I\right)}$$

where $\overline{y}_x = \frac{\sum_{i=1}^{I} W_{i\cdot}\overline{y}_{i\cdot}}{\sum_{i=1}^{I} W_{i\cdot}}$.

Based on *F* statistics, the *p*-value can be derived as

$$p = \Pr\left(F\left(I-1, \sum_{i=1}^{I} N_{i\cdot} - I\right) > F\right)$$

where $F\left(I-1, \sum_{i=1}^{I} N_{i\cdot} - I\right)$ is a random variable following a *F* distribution with $I-1$ and $\sum_{i=1}^{I} N_{i\cdot} - I$ degrees of freedom.

At the merge step we calculate the *F* statistic and *p*-value between two categories *i* and $i^{'}$ of *X* as

$$F = \frac{W_{i\cdot}\left(\overline{y}_{i\cdot} - \overline{y}_{i,i'\cdot}\right)^2 + W_{i'\cdot}\left(\overline{y}_{i'\cdot} - \overline{y}_{i,i'\cdot}\right)^2}{\left(M_{i\cdot}^2 + M_{i'\cdot}^2\right) / \left(N_{i\cdot} + N_{i'\cdot} - 2\right)}$$

$$p = \Pr\left(F\left(1, N_{i\cdot} + N_{i'\cdot} - 2\right) > F\right)$$

where $\overline{y}_{i,i'\cdot}$ is the mean of *Y* for a new category $i, i^{'}$ merged by *i* and $i^{'}$:

$$\overline{y}_{i,i'\cdot} = \overline{y}_{i\cdot} + \frac{W_{i'\cdot}}{W_{i\cdot} + W_{i'\cdot}}\left(\overline{y}_{i'\cdot} - \overline{y}_{i\cdot}\right)$$

and $F\left(I-1, N_{i\cdot} + N_{i'\cdot} - 2\right)$ is a random variable following a *F* distribution with 1 and $N_{i\cdot} + N_{i'\cdot} - 2$ degrees of freedom.

Nominal target

The null hypothesis of independence of *X* and *Y* is tested. First a contingency (or count) table is formed using classes of *Y* as columns and categories of the predictor *X* as rows. Then the expected cell frequencies under the null hypothesis are estimated. The observed cell frequencies and the expected cell frequencies are used to calculate the Pearson chi-squared statistic and the *p*-value:

$$X^2 = \sum_{j=1}^{J}\sum_{i=1}^{I} \frac{\left(N_{ij} - \hat{m}_{ij}\right)^2}{\hat{m}_{ij}}$$

where $N_{ij} = \sum_{k \in D} f_k I\left(x_k = i \wedge y_k = j\right)$ is the observed cell frequency and $\hat{m}_{ij}$ is the estimated expected cell frequency for cell $\left(x_k = i, y_k = j\right)$ following the independence model. If $\hat{m}_{ij} = 0$, then $\frac{\left(N_{ij} - \hat{m}_{ij}\right)^2}{\hat{m}_{ij}} = 0$. How to estimate $\hat{m}_{ij}$ is described below.

The corresponding *p*-value is given by $p = \Pr\left(\chi_d^2 > X^2\right)$, where χ_d^2 follows a chi-squared distribution with $d = (J-1)(I-1)$ degrees of freedom.

When we investigate whether two categories *i* and $i^{'}$ of *X* can be merged, the Pearson chi-squared statistic is revised as

$$X^2 = \sum_{j=1}^{J} \left(\frac{\left(N_{ij} - \hat{m}_{ij}\right)^2}{\hat{m}_{ij}} + \frac{\left(N_{i'j} - \hat{m}_{i'j}\right)^2}{\hat{m}_{i'j}} \right)$$

and the *p*-value is given by $p = \Pr\left(\chi^2_{J-1} > X^2\right)$.

Ordinal target

Suppose there are *I* categories of *X*, and *J* ordinal categories of *Y*. Then the null hypothesis of the independence of *X* and *Y* is tested against the row effects model (with the rows being the categories of *X* and columns the classes of *Y*) proposed by Goodman (1979). Two sets of expected cell frequencies, $\hat{m}_{ij}$ (under the hypothesis of independence) and $\hat{\hat{m}}_{ij}$ (under the hypothesis that the data follow a row effects model), are both estimated. The likelihood ratio statistic is

$$H^2 = 2\sum_{i=1}^{I}\sum_{j=1}^{J} H^2_{ij}$$

where

$$H^2_{ij} = \begin{cases} \hat{\hat{m}}_{ij} \ln\left(\hat{\hat{m}}_{ij}/\hat{m}_{ij}\right) \; \hat{\hat{m}}_{ij}/\hat{m}_{ij} > 0 \\ 0 \quad else \end{cases}$$

The *p*-value is given by $p = \Pr\left(\chi^2_{I-1} > H^2\right)$.

Estimated expected cell frequencies (independence assumption)

If analysis weights are specified, the expected cell frequency under the null hypothesis of independence is of the form

$$m_{ij} = \overline{w}_{ij}^{-1} \alpha_i \beta_j$$

where α_i and β_j are parameters to be estimated, and $\overline{w}_{ij} = \frac{W_{ij}}{N_{ij}}$ if $N_{ij} > 0$, otherwise $\overline{w}_{ij} = 1$.

Parameter estimates $\hat{\alpha}_i$, $\hat{\beta}_j$, and hence $\hat{m}_{ij}$, are obtained from the following iterative procedure.

1. $k = 0$, $\alpha_i^{(0)} = \beta_j^{(0)} = 1$, $m_{ij}^{(0)} = \overline{w}_{ij}^{-1}$
2. $\alpha_i^{(k+1)} = \frac{N_{i.}}{\sum_j \overline{w}_{ij}^{-1} \beta_j^{(k)}} = \alpha_i^{(k)} \frac{N_{i.}}{\sum_j m_{ij}^{(k)}}$
3. $\beta_j^{(k+1)} = \frac{N_{.j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k+1)}}$
4. $m_{ij}^{(k+1)} = \overline{w}_{ij}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)}$

5. If $\max_{i,j}\left|m_{ij}^{(k+1)}-m_{ij}^{(k)}\right|<\epsilon$ (default is 0.001) or the number of iterations is larger than a threshold (default is 100), stop and output $\alpha_i^{(k+1)}, \beta_j^{(k+1)}$ and $m_{ij}^{(k+1)}$ as the final estimates $\hat{\alpha}_i, \hat{\beta}_j, \hat{m}_{ij}$. Otherwise, $k=k+1$ and go to step 2.

Estimated expected cell frequencies (row effects model)

In the row effects model, scores for classes of *Y* are needed. By default, s_j^* (the order of a class of *Y*) is used as the class score. These orders will be standardized via the following linear transformation such that the largest score is 100 and the lowest score is 0.

$$s_j = 100\left(s_j^* - s_{\min}^*\right) / \left(s_{\max}^* - s_{\min}^*\right)$$

Where $s_{\min}^*$ and $s_{\max}^*$ are the smallest and largest order, respectively.

The expected cell frequency under the row effects model is given by

$$m_{ij} = \overline{w}_{ij}^{-1}\alpha_i\beta_j\gamma_i$$

where $\overline{s}=\sum_{j=1}^{J} W_{.j}s_j / \sum_{j=1}^{J} W_{.j}$, in which $W_{.j}=\Sigma_i W_{ij}$, and α_i, β_j, and γ_i are unknown parameters to be estimated.

Parameter estimates $\hat{\alpha}_i, \hat{\hat{\beta}}_j, \hat{\hat{\gamma}}_i$ and hence $\hat{\hat{m}}_{ij}$ are obtained from the following iterative procedure.

1. $k=0$, $\alpha_i^{(0)}=\beta_j^{(0)}=\gamma_i^{(0)}=1$, $m_{ij}^{(0)}=\overline{w}_{ij}^{-1}$
2. $\alpha_i^{(k+1)} = \frac{N_{.j}}{\sum_j \overline{w}_{ij}^{-1}\beta_j^{(k)}\left(\gamma_i^{(k)}\right)^{(s_j-\overline{s})}} = \alpha_i^{(k)}\frac{N_{i.}}{\sum_j m_{ij}^{(k)}}$
3. $\beta_j^{(k+1)} = \frac{N_{.j}}{\sum_i \overline{w}_{ij}^{-1}\alpha_i^{(k+1)}\left(\gamma_i^{(k)}\right)^{(s_j-\overline{s})}}$
4. $m_{ij}^* = \overline{w}_{ij}^{-1}\alpha_i^{(k+1)}\beta_j^{(k+1)}\left(\gamma_i^{(k)}\right)^{(s_j-\overline{s})}$, $G_i = 1+\frac{\sum_j (s_j-\overline{s})\left(N_{ij}-m_{ij}^*\right)}{\sum_j (s_j-\overline{s})^2 m_{ij}^*}$
5. $\gamma_i^{(k+1)} = \begin{cases}\gamma_i^{(k)}G_i & G_i>0\\ \gamma_i^{(k)} & \text{otherwise}\end{cases}$
6. $m_{ij}^{(k+1)} = \overline{w}_{ij}^{-1}\alpha_i^{(k+1)}\beta_j^{(k+1)}\left(\gamma_i^{(k+1)}\right)^{(s_j-\overline{s})}$
7. If $\max_{i,j}\left|m_{ij}^{(k+1)}-m_{ij}^{(k)}\right|<\epsilon$ (default is 0.001) or the number of iterations is larger than a threshold (default is 100), stop and output $\alpha_i^{(k+1)}, \beta_j^{(k+1)}, \gamma_i^{(k+1)}$ and $m_{ij}^{(k+1)}$ as the final estimates $\hat{\alpha}_i, \hat{\hat{\beta}}_j, \hat{\hat{\gamma}}_i, \hat{\hat{m}}_{ij}$. Otherwise, $k=k+1$ and go to step 2.

Unsupervised Merge

If there is no target, we merge categories based on counts. Suppose that *X* has *I* categories which are sorted in ascending order. For an ordinal predictor, we sort it according to its values, while for nominal predictor we rearrange categories from lowest to highest count, with ties broken by ascending sort or lexical order of the data values. Let c_i be the number of cases for the *i*th category, and N_X be the total number of cases for *X*. Then we use the equal frequency method to merge sparse categories.

1. Start with $j_1 = j_2 = 1$ and *g*=1.
2. If $j_1 > I$, go to step 5.
3. If $\sum_{i=j_1}^{j_2} c_i < [b\% \times N_X]$, then $j_2 = j_2 + 1$; otherwise the original categories $j_1, j_1 + 1, \cdots, j_2$ will be merged into the new category *g* and let $j_1 = j_2 + 1$, $j_2 = j_1$ and $g = g + 1$, then go to step 2.
4. If $j_2 \geq I$, then merge categories using one of the following rules:

   i) If $g = 1$, then categories $1, 2, \cdots, I - 1$ will be merged into category *g* and *I* will be left unmerged.

   ii) If *g*=2, then $j_1, j_1 + 1, \cdots, I$ will be merged into category *g*=2.

   iii) If *g*>2, then $j_1, j_1 + 1, \cdots, I$ will be merged into category $g - 1$.

   If $j_2 < I$, then go to step 3.
5. Output the merge rule and merged predictor.

   After merging, one of the following rules holds:

- Neither the original category nor any category created during merging has fewer than $[b\% \times N_X]$ cases, where *b* is a user-specified parameter satisfying $1 < b < 100$ (default is 10) and [*x*] denotes the nearest integer of *x*.
- The merged predictor has only two categories.

Update univariate statistics. When original categories $j_1, j_1 + 1, \cdots, j_2$ are merged into one new category, then the number of cases in this new category will be $\sum_{i=j_1}^{j_2} c_j$. At the end of the merge step, we get new categories and the number of cases in each category. Then we update other statistics according to the formulas in “Univariate Statistics Collection ” on p. 15, though note that the statistics only need to be updated based on the new categories and the numbers of cases in these categories.

Continuous Predictor Handling

Continuous predictor handling includes supervised binning when the target is categorical, predictor selection when the target is continuous and predictor construction when the target is continuous or there is no target in the dataset.

After handling continuous predictors, we collect univariate statistics for derived or constructed predictors according to the formulas in “Univariate Statistics Collection ” on p. 15. Any derived predictors that are constant, or have all missing values, are excluded from further analysis.

Supervised Binning

If there is a categorical target, then we will transform each continuous predictor to an ordinal predictor using supervised binning. Suppose that we have already collected the bivariate statistics between the categorical target and a continuous predictor. Using the notations introduced in "Bivariate Statistics Collection " on p. 22, the homogeneous subset will be identified by the Scheffe method as follows:

If $|\overline{x}_{\cdot i} - \overline{x}_{\cdot j}| \leq c_{critical}$ then $\overline{x}_{\cdot i}$ and $\overline{x}_{\cdot j}$ will be a homogeneous subset, where $c_{critical} = \max(\overline{x}_{\cdot i}) - \min(\overline{x}_{\cdot i})$ if $N_{XY} = J$; otherwise$c_{critical} = R * C$, where $R = \sqrt{2(J-1)F_{1-\alpha}(J-1, N_{XY} - J)}$ and $C = MS \times \sqrt{\frac{\sum_{i=1}^{J} 1/W_{\cdot i}}{J}}$, $MS = \sqrt{\frac{\sum_{i=1}^{J} M_{\cdot i}^2}{N_{XY} - J}}$.

The supervised algorithm follows:

1. Sort the means $\overline{x}_{\cdot i}$ in ascending order, denote as $\overline{x}_{\cdot(1)} \leq \overline{x}_{\cdot(2)} \leq \cdots \leq \overline{x}_{\cdot(J)}$.
2. Start with *i*=1 and *q*=*J*.
3. If $\left|\overline{x}_{\cdot(q)} - \overline{x}_{\cdot(i)}\right| \leq c_{critical}$, then $\left\{\overline{x}_{\cdot(i)}, \cdots, \overline{x}_{\cdot(q)}\right\}$ can be considered a homogeneous subset. At the same time we compute the mean and standard deviation of this subset: $\overline{x}_{\cdot(i,q)} = \frac{\sum_{k=i}^{q} W_{\cdot(k)} \overline{x}_{\cdot(k)}}{\sum_{k=i}^{q} W_{\cdot(k)}}$ and $sd_{\cdot(i,q)} = \sqrt{\frac{M_{(i,q)}^2}{\sum_{k=i}^{q} N_{\cdot(k)} - 1}}$, where $M_{(i,q)}^2 = \sum_{k=i}^{q} A_{\cdot(k)}$ and $A_{\cdot(k)} = M_{\cdot(k)}^2 + W_{\cdot(k)}\left(\overline{x}_{\cdot(i,q)} - \overline{x}_{\cdot(k)}\right)^2$, then set $i = q + 1$ and $q = J$; Otherwise $q = q - 1$.
4. If $i \leq J$, go to step 3.
5. Else compute the cut point of bins. Suppose we have $r \leq J$ homogeneous subsets and we assume that the means of these subsets are $\overline{x}_{\cdot(1)}^*, \overline{x}_{\cdot(2)}^*, \cdots, \overline{x}_{\cdot(r)}^*$, and standard deviations are $sd_{\cdot(1)}^*, sd_{\cdot(2)}^*, \cdots, sd_{\cdot(r)}^*$, then the cut points between the *i*th and (*i*+1)th homogeneous subsets are computed as $cut_i = \overline{x}_{(i)}^* + \frac{sd_{(i)}^* + \epsilon}{\left(sd_{(i)}^* + sd_{(i+1)}^* + 2\epsilon\right)}\left(\overline{x}_{(i+1)}^* - \overline{x}_{(i)}^*\right)$.
6. Output the binning rules. Category 1: $X \leq cut_1$; Category 2: $cut_1 < X \leq cut_2$;...; Category : $cut_{r-1} < X$.

Feature Selection and Construction

If there is a continuous target, we perform predictor selection using *p*-values derived from the correlation or partial correlation between the predictors and the target. The selected predictors are grouped if they are highly correlated. In each group, we will derive a new predictor using principal component analysis. However, if there is no target, we will do not implement predictor selection.

To identify highly correlated predictors, we compute the correlation between a scale and a group as follows: suppose that *X* is a continuous predictor and continuous predictors $X_1, X_2, \cdots, X_m$ form a group *G*. Then the correlation between *X* and group *G* is defined as:

$$r_{XG} = \min\left\{\left|r_{XX_i}\right|, X_i \in G\right\}$$

where r_{XX_i} is correlation between *X* and X_i.

Let α_{group} be the correlation level at which the predictors are identified as groups. The predictor selection and predictor construction algorithm is as follows:

1. (Target is continuous and predictor selection is in effect) If the *p*-value between a continuous predictor and target is larger than a threshold (default is 0.05), then we remove this predictor from the correlation matrix and covariance matrix. See "Correlation and Partial Correlation " on p. 34 for details on computing these *p*-values.
2. Start with $\alpha_{group} = 0.9$ and *i*=1.
3. If $\alpha_{group} \leq 0.1$, stop and output all the derived predictors, their source predictors and coefficient of each source predictor. In addition, output the remaining predictors in the correlation matrix.
4. Find the two most correlated predictors such that their correlation in absolute value is larger than α_{group}, and put them in group *i*. If there are no predictors to be chosen, then go to step 9.
5. Add one predictor to group *i* such that the predictor is most correlated with group *i* and the correlation is larger than α_{group}. Repeat this step until the number of predictors in group *i* is greater than a threshold (default is 5) or there is no predictor to be chosen.
6. Derive a new predictor from the group *i* using principal component analysis. For more information, see the topic "Principal Component Analysis " on p. 33.
7. (Both predictor selection and predictor construction are in effect) Compute partial correlations between the other continuous predictors and the target, controlling for values of the new predictor. Also compute the *p*-values based on partial correlation. See "Correlation and Partial Correlation " on p. 34 for details on computing these *p*-values. If the *p*-value based on partial correlation between a continuous predictor and continuous target is larger than a threshold (default is 0.05), then remove this predictor from the correlation and covariance matrices.
8. Remove predictors that are in the group from the correlation matrix. Then let *i*=*i*+1 and go to step 4.
9. $\alpha_{group} = \alpha_{group} - 0.1$, then go to step 3.

Notes:

- If only predictor selection is needed, then only step 1 is implemented. If only predictor construction is needed, then we implement all steps except step 1 and step 7. If both predictor selection and predictor construction are needed, then all steps are implemented.
- If there are ties on correlations when we identify highly correlated predictors, the ties will be broken by selecting the predictor with the smallest index in dataset.

Principal Component Analysis

Let $X_1, X_2, \cdots, X_m$ be *m* continuous predictors. Principal component analysis can be described as follows:

1. Input $C_{m\times m}$, the covariance matrix of $X_1, X_2, \cdots, X_m$.
2. Calculate the eigenvectors and eigenvalues of the covariance matrix. Sort the eigenvalues (and corresponding eigenvectors) in descending order, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$.

3. Derive new predictors. Suppose the elements of the first component v_1 are $v_{11}, v_{12}, \cdots, v_{1m}$, then the new derived predictor is $\frac{v_{11}}{\sqrt{\lambda_1}}X_1 + \frac{v_{12}}{\sqrt{\lambda_1}}X_2 + \cdots + \frac{v_{1m}}{\sqrt{\lambda_1}}X_m$.

Correlation and Partial Correlation

Correlation and P-value

Let r_{XY} be the correlation between continuous predictor *X* and continuous target *Y*, then the *p*-value is derived form the *t* test:

$$p = \Pr\left(\left|t\left(N_{XY}-2\right)\right| > t\right)$$

where $t\left(N_{XY}-2\right)$ is a random variable with a *t* distribution with $N_{XY}-2$ degrees of freedom, and $t = r_{XY}\sqrt{\frac{N_{XY}-2}{1-r_{XY}^2}}$. If $r_{XY}^2 = 1$, then set *p*=0; If $N_{XY} \le 2$, then set *p*=1.

Partial correlation and P-value

For two continuous variables, *X* and *Y*, we can calculate the partial correlation between them controlling for the values of a new continuous variable Z:

$$r_{XY|Z} = \frac{r_{XY} - r_{XZ}r_{YZ}}{\sqrt{1-r_{XZ}^2}\sqrt{1-r_{YZ}^2}}$$

Since the new variable *Z* is always a linear combination of several continuous variables, we compute the correlation of *Z* and a continuous variable using a property of the covariance rather than the original dataset. Suppose the new derived predictor *Z* is a linear combination of original predictors $X_1, X_2, \cdots, X_m$:

$$Z = a_1X_1 + a_2X_2 + \cdots + a_mX_m$$

Then for any a continuous variable *X* (continuous predictor or continuous target), the correlation between *X* and *Z* is

$$r_{ZX} = \frac{c_{ZX}}{\sqrt{c_{ZZ}c_{XX}}}$$

where $c_{ZX} = \sum_{i=1}^{m} a_i c_{X_iX}$, and $c_{ZZ} = \sum_{i=1}^{m} a_i^2 c_{X_iX_i} + 2\sum_{i\neq j} a_i a_j c_{X_iX_j}$.

If $1-r_{XZ}^2$ or $1-r_{YZ}^2$ is less than 10^{-10}, let $r_{XY|Z} = 0$. If $r_{XY|Z}$ is larger than 1, then set it to 1; If $r_{XY|Z}$ is less than −1, then set it to −1. (This may occur with pairwise deletion). Based on partial correlation, the *p*-value is derived from the *t* test

$$p = \Pr\left(\left|t\left(N_{XY}-3\right)\right| > t\right)$$

where $t\left(N_{XY}-3\right)$ is a random variable with a *t* distribution with $N_{XY}-3$ degrees of freedom, and $t = r_{XY|Z}\sqrt{\frac{N_{XY}-3}{1-r_{XY|Z}^2}}$. If $r_{XY|Z}^2 = 1$, then set *p*=0; if $N_{XY} \le 3$, then set *p*=1.

Discretization of Continuous Predictors

Discretization is used for calculating predictive power and creating histograms.

Discretization for calculating predictive power

If the transformed target is categorical, we use the equal width bins method to discretize a continuous predictor into a number of bins equal to the number of categories of the target. Variables considered for discretization include:

- Scale predictors which have been recommended.
- Original continuous variables of recommended predictors.

Discretization for creating histograms

We use the equal width bins method to discretize a continuous predictor into a maximum of 400 bins. Variables considered for discretization include:

- Recommended continuous variables.
- Excluded continuous variables which have not been used to derive a new variable.
- Original continuous variables of recommended variables.
- Original continuous variables of excluded variables which have not been used to derive a new variable.
- Scale variables used to construct new variables. If their original variables are also continuous, then the original variables will be discretized.
- Date/time variables.

After discretization, the number of cases and mean in each bin are collected to create histograms.

Note: If an original predictor has been recast, then this recast version will be regarded as the “original” predictor.

Predictive Power

Collect bivariate statistics for predictive power

We collect bivariate statistics between recommended predictors and the (transformed) target. If an original predictor of a recommended predictor exists, then we also collect bivariate statistics between this original predictor and the target; if an original predictor has a recast version, then we use the recast version.

If the target is categorical, but a recommended predictor or its original predictor/recast version is continuous, then we discretize the continuous predictor using the method in “Discretization of Continuous Predictors ” on p. 35 and collect bivariate statistics between the categorical target and the categorical predictors.

Bivariate statistics between the predictors and target are same as those described in “Bivariate Statistics Collection ” on p. 22.

Computing predictive power

Predictive power is used to measure the usefulness of a predictor and is computed with respect to the (transformed) target. If an original predictor of a recommended predictor exists, then we also compute predictive power for this original predictor; if an original predictor has a recast version, then we use the recast version.

Scale target. When the target is continuous, we fit a linear regression model and predictive power is computed as follows.

- Scale predictor: $r_{XY}^2 = \left(\frac{c_{XY}}{\sqrt{c_{XX}}\sqrt{c_{YY}}}\right)^2$
- Categorical predictor: $1 - \frac{S_e}{S_T}$, where $S_e = \sum\limits_{i=1}^{I} M_{i\cdot}^2$and $S_T = \sum_{i=1}^{n} f_i w_i (y_i - \overline{y}_x)^2$.

Categorical target. If the (transformed) target is categorical, then we fit a naïve Bayes model and the classification accuracy will serve as predictive power. We discretize continuous predictors as described in “Discretization of Continuous Predictors ” on p. 35, so we only consider the predictive power of categorical predictors.

If N_{ij} is the of number cases where $X = i$ and $Y = j$, $N_{i.} = \sum_{j=1}^{J} N_{ij}$, and $N_{.j} = \sum_{i=1}^{I} N_{ij}$, then the chi-square statistic is calculated as

$$\chi^2 = \sum_{i=1}^{I}\sum_{j=1}^{J}\frac{\left(N_{ij} - \hat{N}_{ij}\right)^2}{\hat{N}_{ij}}$$

where $\hat{N}_{ij} = \frac{N_{i.}N_{.J}}{N_{XY}}$

and Cramer’s V is defined as

$$V = \left(\frac{\chi^2}{N_{XY}\left(\min\left(I, J\right) - 1\right)}\right)^{1/2}$$

References

Box, G. E. P., and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society, Series B*, 26, 211–246.

Goodman, L. A. 1979. Simple models for the analysis of association in cross-classifications having ordered categories. *Journal of the American Statistical Association*, 74, 537–552.

Bayesian Networks Algorithms

Bayesian Networks Algorithm Overview

A Bayesian network provides a succinct way of describing the joint probability distribution for a given set of random variables.

Let V be a set of categorical random variables and G = (V, E) be a directed acyclic graph with nodes V and a set of directed edges E. A Bayesian network model consists of the graph G together with a conditional probability table for each node given values of its parent nodes. Given the value of its parents, each node is assumed to be independent of all the nodes that are not its descendents. The joint probability distribution for variables V can then be computed as a product of conditional probabilities for all nodes, given the values of each node's parents.

Given set of variables V and a corresponding sample dataset, we are presented with the task of fitting an appropriate Bayesian network model. The task of determining the appropriate edges in the graph G is called **structure learning**, while the task of estimating the conditional probability tables given parents for each node is called **parameter learning**.

Primary Calculations

IBM® SPSS® Modeler offers two different methods for building Bayesian network models:

- **Tree Augmented Naïve Bayes.** This algorithm is used mainly for classification. It efficiently creates a simple Bayesian network model. The model is an improvement over the naïve Bayes model as it allows for each predictor to depend on another predictor in addition to the target variable. Its main advantages are its classification accuracy and favorable performance compared with general Bayesian network models. Its disadvantage is also due to its simplicity; it imposes much restriction on the dependency structure uncovered among its nodes.
- **Markov Blanket estimation.** The Markov blanket for the target variable node in a Bayesian network is the set of nodes containing target's parents, its children, and its children's parents. Markov blanket identifies all the variables in the network that are needed to predict the target variable. This can produce more complex networks, but also takes longer to produce. Using feature selection preprocessing can significantly improve performance of this algorithm.

Notation

The following notation is used throughout this algorithm description:

G	A directed acyclic graph representing the Bayesian Network model
D	A dataset
Y	Categorical target variable
X_i	The ith predictor
π_i	The parent set of the ith predictor besides target Y. For TAN models, its size is $\leq$1.
N	The number of cases in D

n	The number of predictors
N_{ijk}	Denote the number of records in D for which (π_i, Y) take its *j*th value and for which X_i takes its *k*th value.
N_{ij}	Denote the number of records in D for which (π_i, Y) takes its *j*th value.
θ_{ijk}	$\Pr\left(X_i = x_i^k \vert (\pi_i, Y) = (\pi_i, Y)^j\right)$
θ_{Y_i}	$\Pr(Y = Y_i)$
K	The number of non-redundant parameters of TAN
MB	The Markov blanket boundary about target Y
S	A subset of X
$S_{X_i X_j}$	A subset of $X \setminus X_i, X_j$, such that variables X_i and X_j are conditionally independent with respect to $S_{X_i X_j}$
$X_i - X_j$	An undirected arc between variables X_i, X_j in *G*. $X_i$and X_j are adjacent to each other.
$X_i \to X_j$	A directed arc from X_i to X_j in *G*. X_i is a parent of X_j, and X_j is a child of X_i.
ADJ_{X_i}	A variable set which represents all the adjacent variables of variable X_i in *G*, ignoring the edge directions.
$I(\cdot)$	The conditional independence (CI) test function which returns the *p*-value of the test.
α	The significance level for CI tests between two variables. If the *p*-value of the test is larger than α then they are independent, and vice-versa.
r_i	The cardinality of X_i, $r_i = \vert X_i \vert$
q_i	The cardinality of the parent set π_i of X_i .

Handling of Continuous Predictors

BN models in IBM® SPSS® Modeler can only accommodate discrete variables. Target variables must be discrete (flag or set type). Numeric predictors are discretized into 5 equal-width bins before the BN model is built. If any of the constructed bins is empty (there are no records with a value in the bin's range), that bin is merged to an adjacent non-empty bin.

Feature Selection via Breadth-First Search

Feature selection preprocessing works as follows:

- It begins by searching for the direct neighbors of a given target *Y*, based on statistical tests of independence. For more information, see the topic "Markov Blanket Conditional Independence Test" on p. 43. These variables are known as the parents or children of *Y*, denoted by $PC(Y)$.
- For each $X \in PC(Y)$, we look for $PC(X)$, or the parents and children of *X*.
- For each $Z \in PC(X)$, we add it to MB_Y if it is not independent of *Y*.

The explicit algorithm is given below.

```
RecognizeMB
(
 D : Dataset, eps : threshold
)
{
  // Recognize Y's parents/children
  CanADJ_Y = X \ {Y};
  PC = RecognizePC(Y,CanADJ_Y,D,eps);
  MB = PC;

  // Collect spouse candidates, and remove false
  // positives from PC
  for (each X_i in PC){
    CanADJ_X_i = X \ X_i;
    CanSP_X_i = RecognizePC(X_i,CanADJ_X_i,D,eps);
    if (Y notin CanSP_X_i)   // Filter out false positive
      MB = MB \ X_i;
  }
  // Discover true positives among candidates
  for (each X_i in MB)
    for (each Z_i in CanSP_X_i and Z_i notin MB)
      if (I(Y,Z_i|{S_Y,Z_i + X_i}) ≤ eps) then
        MB = MB + Z_i;
  return MB;
}
```

```
RecognizePC (
 T          : target to scan,
 ADJ_T      : Candidate adjacency set to search,
 D          : Dataset,
 eps        : threshold,
 maxSetSize :  )
{
  NonPC = {empty set};
  cutSetSize = 0;
  repeat
    for (each X_i in ADJ_T){
      for (each subset S of {ADJ_T \ X_i} with |S| = cutSetSize){
        if (I(X_i,T|S) > eps){
          NonPC = NonPC + X_i;
          S_T,X_i = S;
          break;
        }
      }
    }
    if (|NonPC| > 0){
      ADJ_T = ADJ_T \ NonPC;
      cutSetSize +=1;
      NonPC = {empty set};
    } else
      break;
  until (|ADJ_T| ≤ cutSetSize) or (cutSetSize > maxSetSize)
  return ADJ_T;
}
```

Tree Augmented Naïve Bayes Method

The Bayesian network classifier is a simple classification method, which classifies a case $d_j = \left(x_1^j, x_2^j, \ldots, x_n^j\right)$ by determining the probability of it belonging to the *i*th target category Y_i. These probabilities are calculated as

$$
\begin{aligned}
&\Pr\left(Y_i | X_1 = x_1^j, X_2 = x_2^j, \ldots, X_n = x_n^j\right) \\
&= \frac{\Pr(Y_i)\Pr\left(X_1 = x_1^j, X_2 = x_2^j, \ldots, X_n = x_n^j | Y_i\right)}{\Pr\left(X_1 = x_1^j, X_2 = x_2^j, \ldots, X_n = x_n^j\right)} \\
&\propto \Pr\left(Y_i\right) \prod_{k=1}^{n} \Pr\left(X_k = x_k^j | \pi_k^j, Y_i\right)
\end{aligned}
$$

where π_k is the parent set of X_k besides Y, and it maybe empty. $\Pr\left(X_k | \pi_k, Y\right)$ is the conditional probability table (CPT) associated with each node X_k. If there are *n* independent predictors, then the probability is proportional to

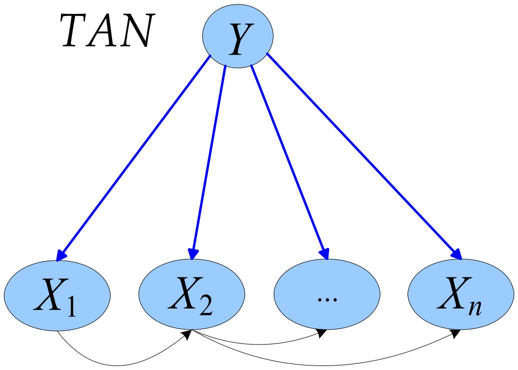
$$\Pr\left(Y_i\right)\prod_{k=1}^{n}\Pr\left(X_k = x_k^j | Y_i\right)$$

When this dependence assumption (conditional independence between the predictors given the class) is made, the classifier is called naïve Bayes (NB). Naïve Bayes has been shown to be competitive with more complex, state-of-the-art classifiers. In recent years, a lot of work has focused on improving the naïve Bayes classifier. One important method is to relax independence assumption. We use a tree augmented naïve Bayesian (TAN) classifier (Friedman, Geiger, and Goldszmidt, 1997), and it is defined by the following conditions:

- Each predictor has the target as a parent.
- Predictors may have one other predictor as a parent.

An example of this structure is shown below.

Figure 5-1
Structure of an simple tree augmented naïve Bayes model.



TAN Classifier Learning Procedure

Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ represent a categorical predictor vector. The algorithm for the TAN classifier first learns a tree structure over $\mathbf{X}$ using mutual information conditioned on Y. Then it adds a link (or arc) from the target node to each predictor node.

The TAN learning procedure is:

1. Take the training data D, $\mathbf{X}$ and Y as input.
2. Learn a tree-like network structure over $\mathbf{X}$ by using the Structure Learning algorithm outlined below.
3. Add Y as a parent of every X_i where $1 \le i \le n$.
4. Learning the parameters of TAN network.

TAN Structure Learning

We use a maximum weighted spanning tree (MWST) method to construct a tree Bayesian network from data (Chow and Liu, 1968). This method associates a weight to each edge corresponding to the mutual information between the two variables. When the weight matrix is created, the MWST algorithm (Prim, 1957) gives an undirected tree that can be oriented with the choice of a root.

The mutual information of two nodes X_i, X_j is defined as

$$I\left(X_i, X_j\right) = \sum_{x_i, x_j} \Pr\left(x_i, x_j\right) \log\left(\frac{\Pr\left(x_i, x_j\right)}{\Pr\left(x_i\right)\Pr\left(x_j\right)}\right)$$

We replace the mutual information between two predictors with the conditional mutual information between two predictors given the target (Friedman et al., 1997). It is defined as

$$I\left(X_i, X_j|Y\right) = \sum_{x_i, x_j, y_k} \Pr\left(x_i, x_j, y_k\right) \log\left(\frac{\Pr\left(x_i, x_j|y_k\right)}{\Pr\left(x_i|y_k\right)\Pr\left(x_j|y_k\right)}\right)$$

The network over can be constructed using the following steps:

1. Compute $I\left(X_i, X_j|Y\right), i = 1, \ldots, n, j = 1, \ldots, n, i \neq j$ between each pair of variables.
2. Use Prim's algorithm (Prim et al., 1957) to construct a maximum weighted spanning tree with the weight of an edge connecting X_i to X_j by $I\left(X_i, X_j|Y\right)$.

   This algorithm works as follows: it begins with a tree with no edges and marks a variable at a random as input. Then it finds an unmarked variable whose weight with one of the marked variables is maximal, then marks this variable and adds the edge to the tree. This process is repeated until all variables are marked.
3. Transform the resulting undirected tree to directed one by choosing X_1 as a root node and setting the direction of all edges to be outward from it.

TAN Parameter Learning

Let r_i be the cardinality of X_i. Let q_i denote the cardinality of the parent set (π_i, Y) of X_i, that is, the number of different values to which the parent of X_i can be instantiated. So it can be calculated as $q_i = r_{\pi_i} \times |Y|$. Note $\pi_i = \emptyset$ implies $q_i = |Y|$. We use N_{ij} to denote the number of records in *D* for which (π_i, Y) takes its *j*th value. We use N_{ijk} to denote the number of records in *D* for which (π_i, Y) take its *j*th value and for which X_i takes its *k*th value.

Maximum Likelihood Estimation

The closed form solution for the parameters $\theta_{Y_i}\left(1 \leq i \leq |Y|\right)$ and $\theta_{ijk}\left(1 \leq i \leq n, 1 \leq j \leq q_i, 1 \leq k \leq r_i\right)$ that maximize the log likelihood score is

$$\hat{\theta}_{Y_i} = \frac{N_{Y_i}}{N}$$
$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

where N_{Y_i} denotes the number of cases with $Y = Y_i$ in the training data.

Note that if $N_{ij} = 0$, then $\hat{\theta}_{ijk} = 0$.

The number of parameters *K* is

$$K = \sum_{i=1}^{n} (r_i - 1) \cdot q_i + |Y| - 1$$

TAN Posterior Estimation

Assume that Dirichlet prior distributions are specified for the set of parameters θ_{Y_i} $(1 \leq i \leq |Y|)$ as well as for each of the sets θ_{ijk} $(1 \leq k \leq r_i)$, $1 \leq i \leq n$, and $1 \leq j \leq q_i$ (Heckerman, 1999). Let $N^0_{Y_i}$ and N^0_{ijk} denote corresponding Dirichlet distribution parameters such that $N^0 = \sum_i N^0_{Y_i}$ and $N^0_{ij} = \sum_k N^0_{ijk}$. Upon observing the dataset *D*, we obtain Dirichlet posterior distributions with the following sets of parameters:

$$\begin{aligned} \hat{\theta}^P_{Y_i} &= \tfrac{N_{Y_i}+N^0_{Y_i}}{N+N^0} \\ \hat{\theta}^P_{ijk} &= \tfrac{N_{ijk}+N^0_{ijk}}{N_{ij}+N^0_{ij}} \end{aligned}$$

The posterior estimation is always used for model updating.

Adjustment for small cell counts

To overcome problems caused by zero or very small cell counts, parameters can be estimated as posterior parameters $\hat{\theta}^P_{Y_i}$ $(1 \leq i \leq |Y|)$ and $\hat{\theta}^P_{ijk}$ $(1 \leq k \leq r_i)$ $, 1 \leq i \leq n, 1 \leq j \leq q_i$ using uninformative Dirichlet priors $N^0_{Y_i} = \frac{2}{|Y|}$ and $N^0_{ijk} = \frac{2}{r_i \cdot q_i}$.

Markov Blanket Algorithms

The Markov blanket algorithm learns the BN structure by identifying the conditional independence relationships among the variables. Using statistical tests (such as chi-squared test or *G* test), this algorithm finds the conditional independence relationships among the nodes and uses these relationships as constraints to construct a BN structure. This algorithm is referred to as a dependency-analysis-based or constraint-based algorithm.

Markov Blanket Conditional Independence Test

The conditional independence (CI) test tests whether two variables are conditionally independent with respect to a conditional variable set. There are two familiar methods to compute the CI test: χ^2 (Pearson chi-square) test and G^2 (log likelihood ratio) test.

Suppose X, Y are two variables for testing and S is a conditional variable set such that $X, Y \notin S$. Let $O(x_i, y_j)$ be the observed count of cases that have $X = x_i$ and $Y = y_j$, and $E(x_i, y_j)$ is the expect number of cases that have $X = x_i$ and $Y = y_j$ under the hypothesis that X, Y are independent.

Chi-square Test

We assume the null hypothesis is that X, Y are independent. The χ^2 test statistic for this hypothesis is

$$\chi^2(X,Y) = \sum_{i,j} \frac{\left(O(x_i,y_j) - E(x_i,y_j)\right)^2}{E(x_i,y_j)}$$

Suppose that N is the total number of cases in D, $N(x_i)$ is the number of cases in D where X_i takes its ith category, and $N(y_j)$ and $N(s_k)$ are the corresponding numbers for Y and S. So $N(x_i, y_j)$ is the number of cases in D where X_i takes its ith category and Y_j takes its jth category. $N(x_i, s_k)$, $N(y_j, s_k)$ and $N(x_i, y_j, s_k)$ are defined similarly. We have:

$$\chi^2(X,Y) = \sum_{i,j} \frac{\left(N(x_i,y_j) - N(x_i)N(y_j)/N\right)^2}{N(x_i)N(y_j)/N} = \sum_{i,j} \frac{\left(N \cdot N(x_i,y_j) - N(x_i)N(y_j)\right)^2}{N(x_i)N(y_j) \cdot N}$$

Because $\chi^2(X,Y) \sim \chi^2_v$ where $v = (|X|-1)(|Y|-1)$ is the degrees of freedom for the χ^2 distribution, we get the p-value for $\chi^2(X,Y)$ as follows:

$$P\left(U > \chi^2(X,Y)\right)$$

As we know, the larger p-value, the less likely we are to reject the null hypothesis. For a given significance level α, if the p-value is greater than α we can not reject the hypothesis that X, Y are independent.

We can easily generalize this independence test into a conditional independence test:

$$\chi^2(X,Y|S) = \sum_k \chi^2(X,Y|S=s_k)$$

$$= \sum_{i,j,k} \frac{\left(N(x_i,y_j,s_k)N(s_k) - N(x_i,s_k)N(y_j,s_k)\right)^2}{N(x_i,s_k)N(y_j,s_k)N(s_k)}$$

The degree of freedom for $\chi^2 \sim \chi^2_v$ is:

$$\nu = (|X|-1)(|Y|-1) \cdot |S|$$

Likelihood Ratio Test

We assume the null hypothesis is that X, Y are independent. The G^2 test statistic for this hypothesis is

$$G^2\left(X,Y\right)=2\sum_{i,j} O\left(x_i,y_j\right)\ln\left(\frac{O\left(x_i,y_j\right)}{E\left(x_i,y_j\right)}\right)$$

or equivalently,

$$G^2\left(X,Y\right)=2\sum_{i,j} N\left(x_i,y_j\right)\ln\left(\frac{N\left(x_i,y_j\right)N}{N\left(x_i\right)N\left(y_j\right)}\right)$$

The conditional version of the G^2 independence test is

$$G^2\left(X,Y|S\right)=2\sum_{i,j,k} O\left(x_i,y_j|S=s_k\right)\ln\left(\frac{O\left(x_i,y_j|S=s_k\right)}{E\left(x_i,y_j|S=s_k\right)}\right)$$

$$=2\sum_{i,j,k} N\left(x_i,y_j,s_k\right)\ \ln\left(\frac{N\left(x_i,y_j,s_k\right)N\left(s_k\right)}{N\left(x_i,s_k\right)N\left(y_j,s_k\right)}\right)$$

The G^2 test is asymptotically distributed as a χ_v^2 distribution, where degrees of freedom are the same as in the χ^2 test. So the *p*-value for the G^2 test is

$$P\left(U>G^2\left(X,Y\right)\right)$$

In the following parts of this document, we use $I\left(\cdot\right)$ to uniformly represent the *p*-value of whichever test is applied. If $I\left(X,Y\right)\ >\alpha$, we say variable *X* and *Y* are independent, and if $I\left(X,Y|S\right)\ >\alpha$, we say variable *X* and *Y* are conditionally independent given variable set *S*.

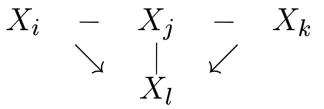
Markov Blanket Structure Learning

This algorithm aims at learning a Bayesian networks structure from a dataset. It starts with a complete graph *G*. Let $X_i,X_j\in\mathbf{X}$, and compute $I\left(X_i,X_j\right)$ for each variable pair in *G*. If $I\left(X_i,X_j\right)\ >\alpha$, remove the arc between X_i,X_j. Then for each arc X_i-X_j perform an exhaustive search in $ADJ_{X_i}\setminus\{X_j\}$ to find the smallest conditional variable set *S* such that $I\left(X_i,X_j|S\right)\ >\alpha$. If such *S* exist, delete arc X_i-X_j. After this, orientation rules are applied to orient the arcs in *G*.

Markov Blanket Arc Orientation Rules

Arcs in the derived structure are oriented based on the following rules:

1. All patterns of the of the form $X_i-X_j-X_k$ or $X_i\to X_j-X_k$ are updated to $X_i\to X_j\leftarrow X_k$ if $X_j\notin S_{X_iX_j}$
2. Patterns of the form $X_i\to X_j-X_k$ are updated so that $X_j\to X_k$
3. Patterns of the form X_i-X_j are updated to $X_i\to X_j$
4. Patterns of the form



are updated so that $X_j \to X_l$

After the last step, if there are still undirected arcs in the graph, return to step 2 and repeat until all arcs are oriented.

Deriving the Markov Blanket Structure

The Markov Blanket is a local structure of a Bayesian Network. Given a Bayesian Network *G* and a target variable *Y*, to derive the Markov Blanket of *Y*, we should select all the directed parents of *Y* in *G* denoted as π_Y, all the directed children of *Y* in *G* denoted as X_{Ch} and all the directed parents of X_{Ch} in *G* denoted as π . $\pi_Y \cup Y \cup X_{Ch} \cup \pi$ and their arcs inherited from *G* define the Markov Blanket MB_Y.

Markov Blanket Parameter Learning

Maximum Likelihood Estimation

The closed form solution for the parameters$\theta_{ijk}\,(1 \le i \le n, 1 \le j \le q_i,\, 1 \le k \le r_i)$ that maximize the log likelihood score is

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

Note that if $\pi_i = \emptyset$, then $\hat{\theta}_{ijk} = \frac{N_k}{N}$.

The number of parameters *K* is

$$K = \sum_{i=1}^{n} (r_i - 1) \cdot q_i$$

Posterior Estimation

Assume that Dirichlet prior distributions are specified for each of the sets $\theta_{ijk}\,(1 \le k \le r_i)\,, 1 \le i \le n, 1 \le j \le q_i$ (Heckerman et al., 1999). Let N^0_{ijk} denote corresponding Dirichlet distributed parameters such that $N^0_{ij} = \sum_k N^0_{ijk}$. Upon observing the dataset *D*, we obtain Dirichlet posterior distributions with the following sets of parameters:

$$\hat{\theta}^P_{ijk} = \frac{N_{ijk} + N^0_{ijk}}{N_{ij} + N^0_{ij}}$$

The posterior estimate is always used for model updating.

Adjustment for Small Cell Counts

To overcome problems caused by zero or very small cell counts, parameters can be estimated as posterior parameters $\theta_{ijk}\left(1 \le k \le r_i\right), 1 \le i \le n, 1 \le j \le q_i$ using uninformative Dirichlet priors specified by $N_{ijk}^0 = \frac{2}{r_i \cdot q_i}$.

Blank Handling

By default, records with missing values for any of the input or output fields are excluded from model building. If the Use only complete records option is deselected, then for each pairwise comparison between fields, all records containing valid values for the two fields in question are used.

Model Nugget/Scoring

The Bayesian Network Model Nugget produces predicted values and probabilities for scored records.

Tree Augmented Naïve Bayes Models

Using the estimated model from training data, for a new case $\mathbf{x} = (x_1, \ldots, x_n)$, the probability of it belonging to the *i*th target category Y_i is calculated as $\Pr\left(Y = Y_i | \mathbf{X} = \mathbf{x}\right)$. The target category with the highest posterior probability is the predicted category for this case, $Y\left(\mathbf{x}\right)$, is predicted by

$$\begin{aligned}
&\hat{Y}\left(\mathbf{x}\right) = arg\max\nolimits_i \left\{\Pr\left(Y = Y_i | \mathbf{X} = \mathbf{x}\right)\right\} \\
&= arg\max\nolimits_i \left\{\Pr\left(\mathbf{X} = \mathbf{x} | Y = Y_i\right)\Pr\left(Y = Y_i\right)\right\} \\
&= arg\max\nolimits_i \left\{\Pr\left(Y = Y_i\right)\prod_{i=1}^{n}\Pr\left(X_i = x_i | \pi_i = \pi_i, Y = Y_i\right)\right\}
\end{aligned}$$

Markov Blanket Models

The scoring function uses the estimated model to compute the probabilities of *Y* belongs to each category for a new case X_P. Suppose π_Y is the parent set of *Y*, and $\pi_{Y|P}$ denotes the configuration of π_Y given case X_P, $X_{Ch} = (X_1, \ldots, X_m)$ denotes the direct children set of *Y*, π_i denotes the parent set (excluding *Y*) of the *i*th variable in X_{Ch}. The score for each category of *Y* is computed by:

$$\Pr\left(Y = y_l | X_P = x_P\right) = \frac{Pr\left(Y = y_l, X_P = x_P\right)}{\sum_{y_l} Pr\left(Y = y_l, X_P = x_P\right)}$$

where the joint probability that $Y = y_l$ and $X_P = x_P$ is:

$$\Pr\left(Y = y_l, X_P = x_P\right) = c \cdot Pr\left(Y = y_l | \pi_Y = \pi_{y|P}\right)\prod_{i=1}^{m} Pr\left(X_i = x_i | \pi_i = \pi_{i|P}, Y = y_l\right)$$

where

$$c = \Pr\left(\pi_Y = \pi_{y|P}\right) \prod_{i=1}^{m} Pr\left(\pi_i = \pi_{i|P}\right)$$

Note that *c* is never actually computed during scoring because its value cancels from the numerator and denominator of the scoring equation given above.

Binary Classifier Comparison Metrics

The Binary Classifier node generates multiple models for a flag output field. For details on how each model type is built, see the appropriate algorithm documentation for the model type.

The node also reports several comparison metrics for each model, to help you select the optimal model for your application. The following metrics are available:

Maximum Profit

This gives the maximum amount of profit, based on the model and the profit and cost settings. It is calculated as

$$\text{Profit}_{\max} = \sum_{i=1}^{j} \left(h(x_i) \cdot r - c\right)$$

where $h(x_i)$ is defined as

$$h(x_i) = \begin{cases} 1 & \text{if } x_i \text{ is a hit} \\ 0 & \text{otherwise} \end{cases}$$

r is the user-specified revenue amount per hit, and *c* is the user-specified cost per record. The sum is calculated for the *j* records with the highest $\hat{p}_i$, such that $(\hat{p}_{j+1} \cdot (r - c)) - \left(\left(1 - \hat{p}_{j+1}\right) \cdot c\right) \leq 0$

Maximum Profit Occurs in %

This gives the percentage of the training records that provide positive profit based on the predictions of the model,

$$\text{Profit}_{\%} = \frac{j}{n} \cdot 100\%$$

where *n* is the overall number of records included in building the model.

Lift

This indicates the response rate for the top *q*% of records (sorted by predicted probability), as a ratio relative to the overall response rate,

$$\text{Lift} = \frac{\sum_{i=1}^{k} \hat{p}_i / k}{\sum_{i=1}^{n} h(x_i)/n}$$

where *k* is *q*% of *n*, the number of training records used to build the model. The default value of *q* is 30, but this value can be modified in the binary classifier node options.

Overall Accuracy

This is the percentage of records for which the outcome is correctly predicted,

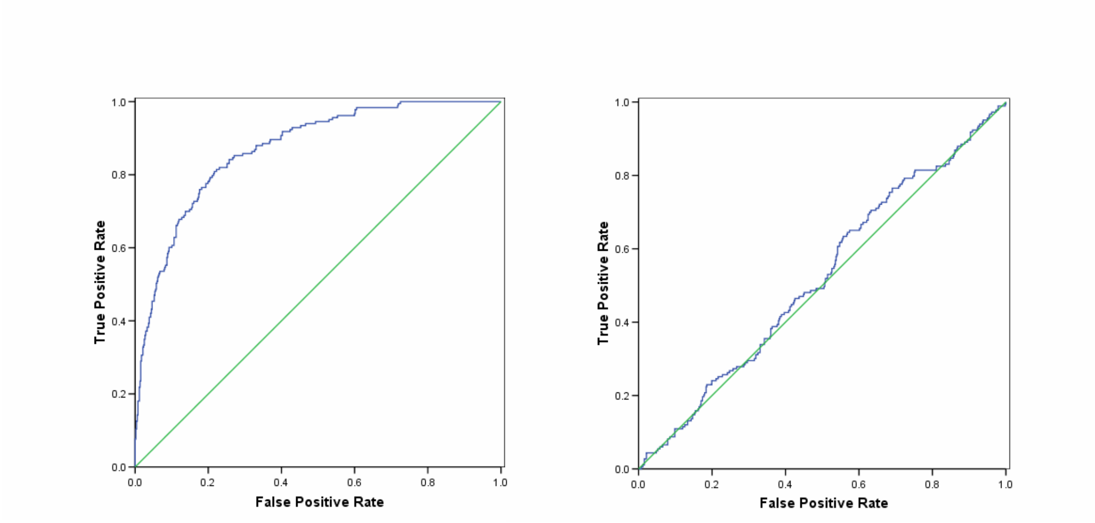
$$a = \frac{\sum_{i=1}^{n} m(i)}{n} \cdot 100\%, m(i) = \begin{cases} 1 & \text{if } (\hat{x}_i = x_i) \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{x}_i$ is the predicted outcome value for record *i* and x_i is the observed value.

Area Under the Curve (AUC)

This represents the area under the Receiver Operating Characteristic (ROC) curve for the model. The ROC curve plots the true positive rate (where the model predicts the target response and the response is observed) against the false positive rate (where the model predicts the target response but a nonresponse is observed). For a good model, the curve will rise sharply near the left axis and cut across near the top, so that nearly all the area in the unit square falls below the curve. For an uninformative model, the curve will approximate a diagonal line from the lower left to the upper right corner of the graph. Thus, the closer the AUC is to 1.0, the better the model.

Figure 6-1
ROC curves for a good model (left) and an uninformative model (right)



The AUC is computed by identifying segments as unique combinations of predictor values that determine subsets of records which all have the same predicted probability of the target value. The *s* segments defined by a given model's predictors are sorted in descending order of predicted probability, and the AUC is calculated as

$$AUC = \sum_{i=1}^{s} |f_i - f_{i-1}| \cdot \frac{t_i + t_{i-1}}{2}$$

where f_i is the cumulative number of false positives for segment *i*, that is, false positives for segment *i* and all preceding segments $j < i$, t_i is the cumulative number of true positives, and $f_0 = t_0 = 0$.

C5.0 Algorithms

The code for training C5.0 models is licensed from RuleQuest Research Ltd Pty, and the algorithms are proprietary. For more information, see the RuleQuest website at *http://www.rulequest.com/*.

Note: Modeler 13 upgraded the C5.0 version from 2.04 to 2.06. See the RuleQuest website for more information.

Scoring

A record is scored with the class and confidence of the rule that fires for that record.

If a rule set is directly generated from the C5.0 node, then the confidence for the rule is calculated as

$$\frac{(\text{number correct in leaf} + 1)}{(\text{total number of records in leaf} + 2)}$$

If a rule set is generated from a decision tree generated from the C5.0 node, then the confidence is calculated as

$$\frac{(\text{number correct in leaf} + 1)}{(\text{total number of records in leaf} + \text{number of categories in the target})}$$

Scores with rule set voting

When voting occurs between rules within a rule set the final scores assigned to a record are calculated in the following way. For each record, all rules are examined and each rule that applies to the record is used to generate a prediction and an associated confidence. The sum of confidence figures for each output value is computed, and the value with the greatest confidence sum is chosen as the final prediction. The confidence for the final prediction is the confidence sum for that value divided by the number of rules that fired for that record.

Scores with boosted C5.0 classifiers (decision trees and rule sets)

When scoring with a boosted C5.0 rule set the *n* rule sets that make up the boosted rule set (one rule set for each boosting trial) vote using their individual scores (as obtained above) to arrive at the final score assigned to the case by the boosted rule set.

The voting for boosted C5 classifiers is as follows. For each record, each composite classifier (rule set or decision tree) assigns a prediction and a confidence. The sum of confidence figures for each output value is computed, and the value with the greatest confidence sum is chosen as the final prediction. The confidence for the final prediction by the boosted classifier is the confidence sum for that value divided by confidence sum for all values.

Carma Algorithms

Overview

The continuous association rule mining algorithm (Carma) is an alternative to Apriori that reduces I/O costs, time, and space requirements (Hidber, 1999). It uses only two data passes and delivers results for much lower support levels than Apriori. In addition, it allows changes in the support level during execution.

Carma deals with items and itemsets that make up transactions. **Items** are flag-type conditions that indicate the presence or absence of a particular thing in a specific transaction. An **itemset** is a group of items which may or may not tend to co-occur within transactions.

Deriving Rules

Carma proceeds in two stages. First it identifies frequent itemsets in the data, and then it generates rules from the lattice of frequent itemsets.

Frequent Itemsets

Carma uses a two-phase method of identifying frequent itemsets.

Phase I: Estimation

In the estimation phase, Carma uses a single data pass to identify frequent itemset candidates. A **lattice** is used to store information on itemsets. Each node in the lattice stores the items comprising the itemset, and three values for the associated itemset:

- *count*: number of transactions containing the itemset since the itemset was added to the lattice
- *firstTrans*: the record index of the transaction for which the itemset was added to the lattice
- *maxMissed*: upper bound on the number of occurrences of the itemset before it was added to the lattice

The lattice also encodes information on relationships between itemsets, which are determined by the items in the itemset. An itemset *Y* is an **ancestor** of itemset *X* if *X* contains every item in *Y*. More specifically, *Y* is a **parent** of *X* if *X* contains every item in *Y* plus one additional item. Conversely, *Y* is a **descendant** of *X* if *Y* contains every item in *X*, and *Y* is a **child** of *X* if *Y* contains every item in *X* plus one additional item.

For example, if X = {milk, cheese, bread}, then Y = {milk, cheese} is a parent of X, and Z = {milk, cheese, bread, sugar} is a child of X.

Initially the lattice contains no itemsets. As each transaction is read, the lattice is updated in three steps:

- **Increment statistics.** For each itemset in the lattice that exists in the current transaction, increment the count value.

- **Insert new itemsets.** For each itemset *v* in the transaction that is not already in the lattice, check all subsets of the itemset in the lattice. If all possible subsets of the itemset are in the lattice with $maxSupport \geq \sigma_i$, then add the itemset to the lattice and set its values:
  - *count* is set to 1
  - *firstTrans* is set to the record index of the current transaction
  - *maxMissed* is defined as

$$maxMissed(v) = \min_{w \subset v} \left\{ \left(\lfloor (i-1) avg\left(\lceil \sigma \rceil_{i-1} \right) \rfloor + |v| - 1 \right), (maxMissed(w) + count(w) - 1) \right\}$$

  where *w* is a subset of itemset *v*, $\lceil \sigma \rceil_{i-1}$ is the ceiling of σ up to transaction i for varying support (or simply σ for constant support), and |*v*| is the number of items in itemset *v*.

- **Prune the lattice.** Every *k* transactions (where *k* is the **pruning value**, set to 500 by default), the lattice is examined and small itemsets are removed. A small itemset is defined as an itemset for which *maxSupport* < σ_i, where *maxSupport* = (*maxMissed* + *count*)/*i*.

Phase II: Validation

After the frequent itemset candidates have been identified, a second data pass is made to compute exact frequencies for the candidates, and the final list of frequent itemsets is determined based on these frequencies.

The first step in Phase II is to remove infrequent itemsets from the lattice. The lattice is pruned using the same method described under Phase I, with σ_n as the user-specified support level for the model.

After initial pruning, the training data are processed again and each itemset *v* in the lattice is checked and updated for each transaction record with index *i*:

- If *firstTrans*(*v*) < *i*, *v* is marked as exact and is no longer considered for any updates. (When all nodes in the lattice are marked as exact, phase II terminates.)
- If *v* appears in the current transaction, *v* is updated as follows:
  - Increment *count*(*v*)
  - Decrement *maxMissed*(*v*)
  - If *firstTrans*(*v*) = *i*, set *maxMissed*(*v*) = 0, and adjust *maxMissed* for every superset *w* of *v* in the lattice for which *maxSupport*(*w*) > *maxSupport*(*v*). For such supersets, set *maxMissed*(*w*) = *count*(*v*) - *count*(*w*).
  - If *maxSupport*(*v*) < σ_n, remove *v* from the lattice.

Generating Rules

Carma uses a common rule-generating algorithm for extracting rules from the lattice of itemsets that tends to eliminate redundant rules (Aggarwal and Yu, 1998). Rules are generated from the lattice of itemsets (see “Frequent Itemsets” on p. 53) as follows:

- For each itemset in the lattice, get the set of maximal ancestor itemsets. An itemset *Y* is a maximal ancestor of itemset *X* if $\frac{support(Y)}{support(X)} \leq \frac{1}{c}$, where *c* is the specified confidence threshold for rules.

- Prune the list of maximal ancestors by removing maximal ancestors of all of *X*'s child itemsets.
- For each itemset in the pruned maximal ancestor list, generate a rule $Y \Rightarrow X - Y$, where *X*−*Y* is the itemset *X* with the items in itemset *Y* removed.

  For example, if *X* the itemset {milk, cheese, bread} and *Y* is the itemset {milk, bread}, then the resulting rule would be milk, bread ⇒ cheese

Blank Handling

Blanks are ignored by the Carma algorithm. The algorithm will handle records containing blanks for input fields, but such a record will not be considered to match any rule containing one or more of the fields for which it has blank values.

Effect of Options

Minimum rule support/confidence. These values place constraints on which rules may be entered into the table. Only rules whose support and confidence values exceed the specified values can be entered into the rule table.

Maximum rule size. Sets the limit on the number of items that will be considered as an itemset.

Exclude rules with multiple consequents. This option restricts rules in the final rule list to those with a single item as consequent.

Set pruning value. Sets the number of transactions to process between pruning passes. For more information, see the topic "Frequent Itemsets" on p. 53.

Vary support. Allows support to vary in order to enhance training during the early transactions in the training data. For more information, see "Varying support" below.

Allow rules without antecedents. Allows rules that are consequent only, which are simple statements of co-occuring items, along with traditional if-then rules.

Varying support

If the vary support option is selected, the target support value changes as transactions are processed to provide more efficient training. The support value starts large and decreases in four steps as transactions are processed. The first support value s_1 applies to the first 9 transactions, the second value s_2 applies to the next 90 transactions, the third value s_3 applies to transactions 100-4999, and the fourth value s_4 applies to all remaining transactions. If we call the final support value *s*, and the estimated number of transactions *t*, then the following constraints are used to determine the support values:

- If $s \geq 0.2$ or $t < 19$, set $s_1 = s_2 = s_3 = s_4$.
- If $19 \leq t < 190$, set $s_1 = 5s_2, s_3 = s_4 = s_2$, such that $\frac{(9s_1+(t-9)s_2)}{t} = s$.
- If $190 \leq t < 7000$, set $s_1 = 5s_2, s_2 = 2s_3, s_4 = s_3$, such that $\frac{(9s_1+90s_2+(t-99)s_3)}{t} = s$.

- If $t \geq 7000$, set $s_1 = 5s_2, s_2 = 2s_3, s_3 = 5s_4$, such that $\frac{(9s_1+90s_2+4900s_3+(t-4999)s_4)}{t} = s$.

In all cases, if solving the equation yields $s_1 > 0.5$, s_1 is set to 0.5, and the other values adjusted accordingly to preserve the relation $\frac{\sum_{i=1}^{n} s(i)}{t} = s$, where $s(i)$ is the target support (one of the values s_1, s_2, s_3, or s_4) for the *i*th transaction.

Generated Model/Scoring

The Carma algorithm generates an unrefined rule node. To create a model for scoring new data, the unrefined rule node must be refined to generate a ruleset node. Details of scoring for generated ruleset nodes are given below.

Predicted Values

Predicted values are based on the rules in the ruleset. When a new record is scored, it is compared to the rules in the ruleset. How the prediction is generated depends on the user's setting for Ruleset Evaluation in the stream options.

- **Voting.** This method attempts to combine the predictions of all of the rules that apply to the record. For each record, all rules are examined and each rule that applies to the record is used to generate a prediction. The sum of confidence figures for each predicted value is computed, and the value with the greatest confidence sum is chosen as the final prediction.
- **First hit.** This method simply tests the rules in order, and the first rule that applies to the record is the one used to generate the prediction.

There is a **default rule**, which specifies an output value to be used as the prediction for records that don't trigger any other rules from the ruleset. For rulesets derived from decision trees, the value for the default rule is the modal (most prevalent) output value in the overall training data. For association rulesets, the default value is specified by the user when the ruleset is generated from the unrefined rule node.

Confidence

Confidence calculations also depend on the user's Ruleset Evaluation stream options setting.

- **Voting.** The confidence for the final prediction is the sum of the confidence values for rules triggered by the current record that give the winning prediction divided by the number of rules that fired for that record.
- **First hit.** The confidence is the confidence value for the first rule in the ruleset triggered by the current record.

If the default rule is the only rule that fires for the record, it's confidence is set to 0.5.

Blank Handling

Blanks are ignored by the algorithm. The algorithm will handle records containing blanks for input fields, but such a record will not be considered to match any rule containing one or more of the fields for which it has blank values.

There is an exception to this: when a numeric field is examined based on a split point, user-defined missing values are included in the comparison. For example, if you define -999 as a missing value for a field, Carma will still compare it to the split point for that field, and may return a match if the rule is of the form (X < 50). You may need to preprocess specially coded numeric missing values (replacing them with $null$, for example) before scoring data with Carma.

C&RT Algorithms

Overview of C&RT

C&RT stands for **Classification and Regression Trees**, originally described in the book by the same name (Breiman, Friedman, Olshen, and Stone, 1984). C&RT partitions the data into two subsets so that the records within each subset are more homogeneous than in the previous subset. It is a **recursive** process—each of those two subsets is then split again, and the process repeats until the homogeneity criterion is reached or until some other stopping criterion is satisfied (as do all of the tree-growing methods). The same predictor field may be used several times at different levels in the tree. It uses surrogate splitting to make the best use of data with missing values.

C&RT is quite flexible. It allows unequal misclassification costs to be considered in the tree growing process. It also allows you to specify the prior probability distribution in a classification problem. You can apply automatic cost-complexity pruning to a C&RT tree to obtain a more generalizable tree.

Primary Calculations

The calculations directly involved in building the model are described below.

Frequency and Case Weight Fields

Frequency and case weight fields are useful for reducing the size of your dataset. Each has a distinct function, though. If a case weight field is mistakenly specified to be a frequency field, or vice versa, the resulting analysis will be incorrect.

For the calculations described below, if no frequency or case weight fields are specified, assume that frequency and case weights for all records are equal to 1.0.

Frequency Fields

A **frequency field** represents the total number of observations represented by each record. It is useful for analyzing aggregate data, in which a record represents more than one individual. The sum of the values for a frequency field should always be equal to the total number of observations in the sample. Note that output and statistics are the same whether you use a frequency field or case-by-case data. The table below shows a hypothetical example, with the predictor fields *sex* and *employment* and the target field *response*. The frequency field tells us, for example, that 10 employed men responded *yes* to the target question, and 19 unemployed women responded *no*.

Table 9-1
Dataset with frequency field

Sex	Employment	Response	Frequency
M	Y	Y	10
M	Y	N	17
M	N	Y	12
M	N	N	21
F	Y	Y	11
F	Y	N	15

Sex	Employment	Response	Frequency
F	N	Y	15
F	N	N	19

The use of a frequency field in this case allows us to process a table of 8 records instead of case-by-case data, which would require 120 records.

Case weights

The use of a case weight field gives unequal treatment to the records in a dataset. When a **case weight field** is used, the contribution of a record in the analysis is weighted in proportion to the population units that the record represents in the sample. For example, suppose that in a direct marketing promotion, 10,000 households respond and 1,000,000 households do not respond. To reduce the size of the data file, you might include all of the responders but only a 1% sample (10,000) of the nonresponders. You can do this if you define a case weight equal to 1 for responders and 100 for nonresponders.

Model Parameters

C&RT works by choosing a split at each node such that each child node created by the split is more pure than its parent node. Here **purity** refers to similarity of values of the target field. In a completely pure node, all of the records have the same value for the target field. C&RT measures the impurity of a split at a node by defining an **impurity measure**. For more information, see the topic “Impurity Measures” on p. 62.

The following steps are used to build a C&RT tree (starting with the root node containing all records):

Find each predictor’s best split. For each predictor field, find the best possible split for that field, as follows:

- **Range (numeric) fields.** Sort the field values for records in the node from smallest to largest. Choose each point in turn as a split point, and compute the impurity statistic for the resulting child nodes of the split. Select the best split point for the field as the one that yields the largest decrease in impurity relative to the impurity of the node being split.
- **Symbolic (categorical) fields.** Examine each possible combination of values as two subsets. For each combination, calculate the impurity of the child nodes for the split based on that combination. Select the best split point for the field as the one that yields the largest decrease in impurity relative to the impurity of the node being split.

Find the best split for the node. Identify the field whose best split gives the greatest decrease in impurity for the node, and select that field’s best split as the best overall split for the node.

Check stopping rules, and recurse. If no stopping rules are triggered by the split or by the parent node, apply the split to create two child nodes. (For more information, see the topic “Stopping Rules” on p. 65.) Apply the algorithm again to each child node.

Blank Handling

Records with missing values for the target field are ignored in building the tree model.

Surrogate splitting is used to handle blanks for predictor fields. If the best predictor field to be used for a split has a blank or missing value at a particular node, another field that yields a split similar to the predictor field in the context of that node is used as a surrogate for the predictor field, and its value is used to assign the record to one of the child nodes.

Note: If Surrogate splitting is used (where a particular rule does not fit into a node) the Confidence score is reduced by multiplying it by 0.9. This can result in multiple Confidence scores being present within a single node.

For example, suppose that X^* is the predictor field that defines the best split s^* at node t. The surrogate-splitting process finds another split s, the surrogate, based on another predictor field X such that this split is most similar to s^* at node t (for records with valid values for both predictors). If a new record is to be predicted and it has a missing value on X^* at node t, the surrogate split s is applied instead. (Unless, of course, this record also has a missing value on X. In such a situation, the next best surrogate is used, and so on, up to the limit of number of surrogates specified.)

In the interest of speed and memory conservation, only a limited number of surrogates is identified for each split in the tree. If a record has missing values for the split field and all surrogate fields, it is assigned to the child node with the higher weighted probability, calculated as

$$\frac{N_{f,j}(t)}{N_f(t)}$$

where $N_{\mathrm{f,j}}(t)$ is the sum of frequency weights for records in category j for node t, and $N_{\mathrm{f}}(t)$ is the sum of frequency weights for all records in node t.

If the model was built using equal or user-specified priors, the priors are incorporated into the calculation:

$$\frac{\pi(j)}{p_f(t)} \times \frac{N_{f,j}(t)}{N_f(t)}$$

where $\pi(j)$ is the prior probability for category j, and $p_{\mathrm{f}}(t)$ is the weighted probability of a record being assigned to the node,

$$p_f(t) = \sum_j \frac{\pi(j)N_{f,j}(t)}{N_{f,j}}$$

where $N_{\mathrm{f,j}}(t)$ is the sum of the frequency weights (or the number of records if no frequency weights are defined) in node t belonging to category j, and $N_{\mathrm{f,j}}$ is the sum of frequency weights for records belonging to category in the entire training sample.

Predictive measure of association

Let $\hbar_{X^* \cap X}$ (resp. $\hbar_{X^* \cap X}(t)$) be the set of learning cases (resp. learning cases in node *t*) that has non-missing values of both *X** and *X*. Let $p(s^* \approx s_X|t)$ be the probability of sending a case in $\hbar_{X^* \cap X}(t)$ to the same child by both s^* and s_X, and $\tilde{s}_X$ be the split with maximized probability $p(s^* \approx \tilde{s}_X|t) = \max_{s_X}(p(s^* \approx s_X|t))$.

The predictive measure of association $\lambda(s^* \approx \tilde{s}_X|t)$ between *s** and $\tilde{s}_X$ at node *t* is

$$\lambda(s^* \approx \tilde{s}_X|t) = \frac{\min(p_L, p_R) - (1 - p(s^* \approx \tilde{s}_X|t))}{\min(p_L, p_R)}$$

where p_L (resp. p_R) is the relative probability that the best split *s** at node *t* sends a case with non-missing value of *X** to the left (resp. right) child node. And where

$$p(s^* \approx s_X|t) = \begin{cases} \sum_j \dfrac{\pi(j) N_{w,j}(s^* \approx s_X, t)}{N_{w,j}(X^* \cap X)} & \text{if} Y \text{ is categorical} \\ \frac{N_w(s^* \approx s_X, t)}{N_w(X^* \cap X)} & \text{if} Y \text{ is continuous} \end{cases}$$

with

$$N_w(X^* \cap X) = \sum_{n \in \hbar_{X^* \cap X}} w_n f_n,\ N_w(X^* \cap X, t) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n$$

$$N_w(s^* \approx s_X, t) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n I(n : s^* \approx s_X)$$

$$N_{w,j}(X^* \cap X) = \sum_{n \in \hbar_{X^* \cap X}} w_n f_n I(y_n = j),\ N_{w,j}(X^* \cap X) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n I(y_n = j)$$

$$N_{w,j}(s^* \approx s_X, t) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n I(y_n = j) I(n : s^* \approx s_X)$$

and $I(n : s^* \approx s_X)$ being the indicator function taking value 1 when both splits *s** and s_X send the case *n* to the same child, 0 otherwise.

Effect of Options

Impurity Measures

There are three different impurity measures used to find splits for C&RT models, depending on the type of the target field. For symbolic target fields, you can choose Gini or twoing. For continuous targets, the least-squared deviation (LSD) method is automatically selected.

Gini

The Gini index *g*(*t*) at a node *t* in a C&RT tree, is defined as

$$g(t) = \sum_{j \neq i} p(j|t)p(i|t)$$

where *i* and *j* are categories of the target field, and

$$p(j|t) = \frac{p\,(j,t)}{p(t)}$$

$$p\,(j,t) = \frac{\pi(j)N_j(t)}{N_j}$$

$$p(t) = \sum_j p\,(j,t)$$

where *π*(*j*) is the prior probability value for category *j*, $N_j(t)$ is the number of records in category *j* of node *t*, and N_j is the number of records of category *j* in the root node. Note that when the Gini index is used to find the improvement for a split during tree growth, only those records in node *t* and the root node with valid values for the split-predictor are used to compute $N_j(t)$ and N_j, respectively.

The equation for the Gini index can also be written as

$$g(t) = 1 - \sum_j p^2(j|t)$$

Thus, when the records in a node are evenly distributed across the categories, the Gini index takes its maximum value of 1 - $^1/_k$, where *k* is the number of categories for the target field. When all records in the node belong to the same category, the Gini index equals 0.

The Gini criterion function *Φ*(*s*, *t*) for split *s* at node *t* is defined as

$$\Phi\,(s,t) = g(t) - p_L g(t_L) - p_R g(t_R)$$

where p_L is the proportion of records in *t* sent to the left child node, and p_R is the proportion sent to the right child node. The proportions p_L and p_R are defined as

$$p_L = \frac{p(t_L)}{p(t)}$$

and

$$p_R = \frac{p(t_R)}{p(t)}$$

The split *s* is chosen to maximize the value of *Φ*(*s*, *t*).

Twoing

The twoing index is based on splitting the target categories into two superclasses, and then finding the best split on the predictor field based on those two superclasses. The superclasses C_1 and C_2 are defined as

$$C_1 = \{j : p(j|t_L) \geq p(j|t_R)\}$$

and

$$C_2 = C - C_1$$

where *C* is the set of categories of the target field, and $p(j|t_R)$ and $p(j|t_L)$ are $p(j|t)$, as defined as in the Gini formulas, for the right and left child nodes, respectively. For more information, see the topic “Gini” on p. 62.

The twoing criterion function for split *s* at node *t* is defined as

$$\Phi\left(s,t\right) = p_L p_R \left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2$$

where t_L and t_R are the nodes created by the split *s*. The split s is chosen as the split that maximizes this criterion.

Least Squared Deviation

For continuous target fields, the **least squared deviation** (LSD) impurity measure is used. The LSD measure *R*(*t*) is simply the weighted within-node variance for node *t*, and it is equal to the resubstitution estimate of risk for the node. It is defined as

$$R(t) = \frac{1}{N_W(t)} \sum_{i \in t} w_i f_i (y_i - \overline{y}(t))^2$$

where $N_W(t)$ is the weighted number of records in node *t*, w_i is the value of the weighting field for record *i* (if any), f_i is the value of the frequency field (if any), y_i is the value of the target field, and *y*(*t*) is the (weighted) mean for node *t*. The LSD criterion function for split *s* at node *t* is defined as

$$\Phi\left(s,t\right) = R(t) - p_L R(t_L) - p_R R(t_R)$$

The split *s* is chosen to maximize the value of *Φ*(*s*,*t*).

Stopping Rules

Stopping rules control how the algorithm decides when to stop splitting nodes in the tree. Tree growth proceeds until every leaf node in the tree triggers at least one stopping rule. Any of the following conditions will prevent a node from being split:

- The node is pure (all records have the same value for the target field)
- All records in the node have the same value for all predictor fields used by the model
- The tree depth for the current node (the number of recursive node splits defining the current node) is the *maximum tree depth* (default or user-specified).
- The number of records in the node is less than the *minumum parent node size* (default or user-specified)
- The number of records in any of the child nodes resulting from the node's best split is less than the *minimum child node size* (default or user-specified)
- The best split for the node yields a decrease in impurity that is less than the *minimum change in impurity* (default or user-specified).

Profits

Profits are numeric values associated with categories of a (symbolic) target field that can be used to estimate the gain or loss associated with a segment. They define the relative value of each value of the target field. Values are used in computing gains but not in tree growing.

Profit for each node in the tree is calculated as

$$\sum_j f_j(t)P_j$$

where *j* is the target field category, $f_j(t)$ is the sum of frequency field values for all records in node *t* with category *j* for the target field, and P_j is the user-defined profit value for category *j*.

Priors

Prior probabilities are numeric values that influence the misclassification rates for categories of the target field. They specify the proportion of records expected to belong to each category of the target field prior to the analysis. The values are involved both in tree growing and risk estimation.

There are three ways to derive prior probabilities.

Empirical Priors

By default, priors are calculated based on the training data. The prior probability assigned to each target category is the weighted proportion of records in the training data belonging to that category,

$$\pi(j) = \frac{N_{w,j}}{N_w}$$

In tree-growing and class assignment, the *N*s take both case weights and frequency weights into account (if defined); in risk estimation, only frequency weights are included in calculating empirical priors.

Equal Priors

Selecting equal priors sets the prior probability for each of the *J* categories to the same value,

$$\pi(j) = \frac{1}{J}$$

User-Specified Priors

When user-specified priors are given, the specified values are used in the calculations involving priors. The values specified for the priors must conform to the probability constraint: the sum of priors for all categories must equal 1.0. If user-specified priors do not conform to this constraint, adjusted priors are derived which preserve the proportions of the original priors but conform to the constraint, using the formula

$$\pi'(j) = \frac{\pi(j)}{\sum_J \pi(j)}$$

where *π'*(*j*) is the adjusted prior for category *j*, and *π*(*j*) is the original user-specified prior for category *j*.

Costs

Gini. If costs are specified, the Gini index is computed as

$$g(t) = \sum_{j \neq i} C(i|j)p(j|t)p(i|t)$$

where *C*(*i*|*j*) specifies the cost of misclassifying a category *j* record as category *i*.

Twoing. Costs, if specified, are not taken into account in splitting nodes using the twoing criterion. However, costs will be incorporated into node assignment and risk estimation, as described in Predicted Values and Risk Estimates, below.

LSD. Costs do not apply to regression trees.

Pruning

Pruning refers to the process of examining a fully grown tree and removing bottom-level splits that do not contribute significantly to the accuracy of the tree. In pruning the tree, the software tries to create the smallest tree whose misclassification risk is not too much greater than that of the largest tree possible. It removes a tree branch if the cost associated with having a more complex tree exceeds the gain associated with having another level of nodes (branch).

It uses an index that measures both the misclassification risk and the complexity of the tree, since we want to minimize both of these things. This cost-complexity measure is defined as follows:

$$R_\alpha(T) = R(T) + \alpha\left|\tilde{T}\right|$$

R(*T*) is the misclassification risk of tree *T*, and $\left|\tilde{T}\right|$ is the number of terminal nodes for tree *T*. The term *α* represents the complexity cost *per terminal node* for the tree. (Note that the value of *α* is calculated by the algorithm during pruning.)

Any tree you might generate has a maximum size (T_{max}), in which each terminal node contains only one record. With no complexity cost (*α* = 0), the maximum tree has the lowest risk, since every record is perfectly predicted. Thus, the larger the value of *α*, the fewer the number of terminal nodes in *T*(*α*), where *T*(*α*) is the tree with the lowest complexity cost for the given *α*. As *α* increases from 0, it produces a finite sequence of subtrees (T_1, T_2, T_3), each with progressively fewer terminal nodes. Cost-complexity pruning works by removing the weakest split.

The following equations represent the cost complexity for $\{t\}$, which is any single node, and for T_{t}, the subbranch of $\{t\}$.

$$R_\alpha(\{t\}) = R(t) + \alpha$$

$$R_\alpha(T_t) = R(T_t) + \alpha\left|\tilde{T}_t\right|$$

If $R_\alpha(T_t)$ is less than $R_\alpha(\{t\})$, then the branch T_{t} has a smaller cost complexity than the single node $\{t\}$.

The tree-growing process ensures that $R_\alpha(\{t\}) \geq R_\alpha(T_t)$ for (*α* = 0). As *α* increases from 0, both $R_\alpha(\{t\})$ and $R_\alpha(T_t)$ grow linearly, with the latter growing at a faster rate. Eventually, you will reach a threshold *α'*, such that $R_\alpha(\{t\}) < R_\alpha(T_t)$ for all *α* > *α*'. This means that when *α* grows larger than *α'*, the cost complexity of the tree can be reduced if we cut the subbranch T_{t} under $\{t\}$. Determining the threshold is a simple computation. You can solve this first inequality, $R_\alpha(\{t\}) \geq R_\alpha(T_t)$, to find the largest value of *α* for which the inequality holds, which is also represented by *g*(*t*). You end up with

$$\alpha \leq g(t) = \frac{R(t) - R(T_t)}{\left|\tilde{T}_t\right| - 1}$$

You can define the weakest link (*t*) in tree *T* as the node that has the smallest value of *g*(*t*):

$$g(\bar{t}) = \min_{t \in T} g(t)$$

Therefore, as *α* increases, $\bar{t}$ is the first node for which $R_\alpha(\{t\}) = R_\alpha(T_t)$. At that point, $\{\bar{t}\}$ becomes preferable to $T_{\bar{t}}$, and the subbranch is pruned.

With that background established, the pruning algorithm follows these steps:

- Set α_1 = 0 and start with the tree T_1 = *T*(0), the fully grown tree.

- Increase α until a branch is pruned. Prune the branch from the tree, and calculate the risk estimate of the pruned tree.
- Repeat the previous step until only the root node is left, yielding a series of trees, T_1, T_2, ... T_k.
- If the standard error rule option is selected, choose the smallest tree T_{opt} for which

$$R(T_{opt}) \leq \min_k \; R(T_k) + m \times SE(R(T))$$

- If the standard error rule option is not selected, then the tree with the smallest risk estimate $R(T)$ is selected.

Secondary Calculations

Secondary calculations are not directly related to building the model, but give you information about the model and its performance.

Risk Estimates

Risk estimates describe the risk of error in predicted values for specific nodes of the tree and for the tree as a whole.

Risk Estimates for Symbolic Target Field

For classification trees (with a symbolic target field), the risk estimate $r(t)$ of a node t is computed as

$$r\,(t) = \frac{1}{N_f} \sum_j N_{f,j}(t) C(j^*(t)|j)$$

where $C(j^*(t)|j)$ is the misclassification cost of classifying a record with target value j as $j^*(t)$, $N_{f,j}(t)$ is the sum of the frequency weights for records in node t in category j (or the number of records if no frequency weights are defined), and N_f is the sum of frequency weights for all records in the training data.

If the model uses user-specified priors, the risk estimate is calculated as

$$\sum_j \frac{\pi(j) N_{f,j}(t)}{N_{f,j}} C(j^*(t)|j)$$

Note that case weights are not considered in calculating risk estimates.

Risk Estimates for numeric target field

For regression trees (with a numeric target field), the risk estimate $r(t)$ of a node t is computed as

$$r\,(t) = \frac{1}{N_f(t)} \sum_{i \in t} f_i (y_i - \overline{y}(t))^2$$

where f_i is the frequency weight for record i (a record assigned to node t), y_i is the value of the target field for record i, and $\overline{y}(t)$ is the weighted mean of the target field for all records in node t.

Tree Risk Estimate

For both classification trees and regression trees, the risk estimate $R(T)$ for the tree (T) is calculated by taking the sum of the risk estimates for the terminal nodes $r(t)$:

$$R(T) = \sum_{t \in T'} r(t)$$

where T' is the set of terminal nodes in the tree.

Gain Summary

The **gain summary** provides descriptive statistics for the terminal nodes of a tree.

If your target field is continuous (scale), the gain summary shows the weighted mean of the target value for each terminal node,

$$g(t) = \sum_{i \in t} w_i f_i x_i$$

If your target field is symbolic (categorical), the gain summary shows the weighted percentage of records in a selected target category,

$$g\,(t, j) = \frac{\sum_{i \in t} f_i x_i(j)}{\sum_{i \in t} f_i}$$

where $x_i(j) = 1$ if record x_i is in target category j, and 0 otherwise. If profits are defined for the tree, the gain is the average profit value for each terminal node,

$$g(t) = \sum_{i \in t} f_i P(x_i)$$

where $P(x_i)$ is the profit value assigned to the target value observed in record x_i.

Generated Model/Scoring

Calculations done by the C&RT generated model are described below

Predicted Values

New records are scored by following the tree splits to a terminal node of the tree. Each terminal node has a particular predicted value associated with it, determined as follows:

Classification Trees

For trees with a symbolic target field, each terminal node's predicted category is the category with the lowest weighted cost for the node. This weighted cost is calculated as

$$\min_i \sum_j C(i|j)p(j|t)$$

where $C(i|j)$ is the user-specified misclassification cost for classifying a record as category i when it is actually category j, and $p(j|t)$ is the conditional weighted probability of a record being in category j given that it is in node t, defined as

$$p(j|t) = \frac{p(j,t)}{\sum_j p(j,t)}, p(j,t) = \pi(j)\frac{N_{w,j}(t)}{N_{w,j}}$$

where $\pi(j)$ is the prior probability for category j, $N_{\mathrm{w,j}}(t)$ is the weighted number of records in node t with category j (or the number of records if no frequency or case weights are defined),

$$N_{w,j}(t) = \sum_{i \in t} w_i f_i j(i)$$

and $N_{\mathrm{w,j}}$ is the weighted number records in category j (any node),

$$N_{w,j} = \sum_{i \in T} w_i f_i j(i)$$

Regression Trees

For trees with a numeric target field, each terminal node's predicted category is the weighted mean of the target values for records in the node. This weighted mean is calculated as

$$\overline{y}(t) = \frac{1}{N_w(t)} \sum_{i \in t} w_i f_i y_i$$

where $N_{\mathrm{w}}(t)$ is defined as

$$N_w(t) = \sum_{i \in t} w_i f_i$$

Confidence

For classification trees, confidence values for records passed through the generated model are calculated as follows. For regression trees, no confidence value is assigned.

Classification Trees

Confidence for a scored record is the proportion of weighted records in the training data in the scored record's assigned terminal node that belong to the predicted category, modified by the Laplace correction:

$$\frac{N_{f,j}(t)+1}{N_f(t)+k}$$

Note: If Surrogate Splitting is used (where a particular rule does not fit into a node) the Confidence score is reduced by multiplying it by 0.9. This can result in multiple Confidence scores being present within a single node.

Blank Handling

In classification of new records, blanks are handled as they are during tree growth, using surrogates where possible, and splitting based on weighted probabilities where necessary. For more information, see the topic "Blank Handling" on p. 61.

CHAID Algorithms

Overview of CHAID

CHAID stands for Chi-squared Automatic Interaction Detector. It is a highly efficient statistical technique for segmentation, or tree growing, developed by (Kass, 1980). Using the significance of a statistical test as a criterion, CHAID evaluates all of the values of a potential predictor field. It merges values that are judged to be statistically homogeneous (similar) with respect to the target variable and maintains all other values that are heterogeneous (dissimilar).

It then selects the best predictor to form the first branch in the decision tree, such that each child node is made of a group of homogeneous values of the selected field. This process continues recursively until the tree is fully grown. The statistical test used depends upon the measurement level of the target field. If the target field is continuous, an *F* test is used. If the target field is categorical, a chi-squared test is used.

CHAID is not a binary tree method; that is, it can produce *more* than two categories at any particular level in the tree. Therefore, it tends to create a wider tree than do the binary growing methods. It works for all types of variables, and it accepts both case weights and frequency variables. It handles missing values by treating them all as a single valid category.

Exhaustive CHAID

Exhaustive CHAID is a modification of CHAID developed to address some of the weaknesses of the CHAID method (Biggs, de Ville, and Suen, 1991). In particular, sometimes CHAID may not find the optimal split for a variable, since it stops merging categories as soon as it finds that all remaining categories are statistically different. Exhaustive CHAID remedies this by continuing to merge categories of the predictor variable until only two supercategories are left. It then examines the series of merges for the predictor and finds the set of categories that gives the strongest association with the target variable, and computes an adjusted *p*-value for that association. Thus, Exhaustive CHAID can find the best split for each predictor, and then choose which predictor to split on by comparing the adjusted *p*-values.

Exhaustive CHAID is identical to CHAID in the statistical tests it uses and in the way it treats missing values. Because its method of combining categories of variables is more thorough than that of CHAID, it takes longer to compute. However, if you have the time to spare, Exhaustive CHAID is generally safer to use than CHAID. It often finds more useful splits, though depending on your data, you may find no difference between Exhaustive CHAID and CHAID results.

Primary Calculations

The calculations directly involved in building the model are described below.

Frequency and Case Weight Fields

Frequency and case weight fields are useful for reducing the size of your dataset. Each has a distinct function, though. If a case weight field is mistakenly specified to be a frequency field, or vice versa, the resulting analysis will be incorrect.

For the calculations described below, if no frequency or case weight fields are specified, assume that frequency and case weights for all records are equal to 1.0.

Frequency Fields

A **frequency field** represents the total number of observations represented by each record. It is useful for analyzing aggregate data, in which a record represents more than one individual. The sum of the values for a frequency field should always be equal to the total number of observations in the sample. Note that output and statistics are the same whether you use a frequency field or case-by-case data. The table below shows a hypothetical example, with the predictor fields *sex* and *employment* and the target field *response*. The frequency field tells us, for example, that 10 employed men responded *yes* to the target question, and 19 unemployed women responded *no*.

Table 10-1
Dataset with frequency field

Sex	Employment	Response	Frequency
M	Y	Y	10
M	Y	N	17
M	N	Y	12
M	N	N	21
F	Y	Y	11
F	Y	N	15
F	N	Y	15
F	N	N	19

The use of a frequency field in this case allows us to process a table of 8 records instead of case-by-case data, which would require 120 records.

Case weights

The use of a case weight field gives unequal treatment to the records in a dataset. When a **case weight field** is used, the contribution of a record in the analysis is weighted in proportion to the population units that the record represents in the sample. For example, suppose that in a direct marketing promotion, 10,000 households respond and 1,000,000 households do not respond. To reduce the size of the data file, you might include all of the responders but only a 1% sample (10,000) of the nonresponders. You can do this if you define a case weight equal to 1 for responders and 100 for nonresponders.

Binning of Scale-Level Predictors

Scale level (continuous) *predictor* fields are automatically discretized or **binned** into a set of ordinal categories. This process is performed once for each scale-level predictor in the model, prior to applying the CHAID (or Exhaustive CHAID) algorithm. The binned categories are determined as follows:

1. The data values y_i are sorted.

2. For each unique value, starting with the smallest, calculate the relative (weighted) frequency of values less than or equal to the current value y_i:

$$cf_i = \sum_{y_k < y_i} w_k$$

where w_k is the weight for record k (or 1.0 if no weights are defined).

3. Determine the bin to which the value belongs by comparing the relative frequency with the ideal bin percentile cutpoints of 0.10, 0.20, 0.30, etc.

$$binindex = \frac{g}{W+1} \times 10$$

where W is the total weighted frequency for all records in the training data, $\sum_i w_i$, and

$$g = \begin{cases} cf_{i-1} + \frac{w_i+1}{2}, & w_i \geq 1 \\ cf_{i-1} + \frac{w_i}{2}, & w_i < 1 \end{cases}$$

- If the bin index for this value is different from the bin index for the previous data value, add a new bin to the bin list and set its cutpoint to the current data value.
- If the bin index is the same as the bin index for the previous value, update the cut point for that bin to the current data value.

Normally, CHAID will try to create $k = 10$ bins by default. However, when the number of records having a single value is large (or a set of records with the same value has a large combined weighted frequency), the binning may result in fewer bins. This will happen if the weighted frequency for records with the same value is greater than the expected weighted frequency in a bin (1/*k*th of the total weighted frequency). This will also happen if there are fewer than *k* distinct values for the binned field for records in the training data.

Model Parameters

CHAID works with all types of continuous or categorical fields. However, continuous *predictor* fields are automatically categorized for the purpose of the analysis.For more information, see the topic “Binning of Scale-Level Predictors” on p. 74.

Note that you can set some of the options mentioned below using the Expert Options for CHAID. These include the choice of the Pearson chi-squared or likelihood-ratio test, the level of α_{merge}, the level of α_{split}, score values, and details of stopping rules.

The CHAID algorithm proceeds as follows:

Merging Categories for Predictors (CHAID)

To determine each split, all predictor fields are merged to combine categories that are not statistically different with respect to the target field. Each final category of a predictor field *X* will represent a child node if *X* is used to split the node. The following steps are applied to each predictor field *X*:

1. If *X* has one or two categories, no more categories are merged, so proceed to node splitting below.
2. Find the eligible pair of categories of *X* that is least significantly different (most similar) as determined by the *p*-value of the appropriate statistical test of association with the target field. For more information, see the topic “Statistical Tests Used” on p. 77.

   For ordinal fields, only adjacent categories are eligible for merging; for nominal fields, all pairs are eligible.
3. For the pair having the largest *p*-value, if the *p*-value is greater than α_{merge}, then merge the pair of categories into a single category. Otherwise, skip to step 6.
4. If the user has selected the Allow splitting of merged categories option, and the newly formed compound category contains three or more original categories, then find the best binary split within the compound category (that for which the *p*-value of the statistical test is smallest). If that *p*-value is less than or equal to $\alpha_{\text{split-merge}}$, perform the split to create two categories from the compound category.
5. Continue merging categories from step 1 for this predictor field.
6. Any category with fewer than the user-specified minimum segment size records is merged with the most similar other category (that which gives the largest *p*-value when compared with the small category).

Merging Categories for Predictors (Exhaustive CHAID)

Exhaustive CHAID works much the same as CHAID, except that the category merging is more thoroughly tested to find the ideal set of categories for each predictor field. As with regular CHAID, each final category of a predictor field *X* will represent a child node if *X* is used to split the node. The following steps are applied to each predictor field *X*:

1. For each predictor variable *X*, find the pair of categories of *X* that is least significantly different (that is, has the largest *p*-value) with respect to the target variable *Y*. The method used to calculate the *p*-value depends on the measurement level of *Y*. For more information, see the topic “Statistical Tests Used” on p. 77.
2. Merge into a compound category the pair that gives the largest *p*-value.
3. Calculate the *p*-value based on the new set of categories of *X*. This represents one set of categories for *X*. Remember the *p*-value and its corresponding set of categories.

4. Repeat steps 1, 2, and 3 until only two categories remain. Then, compare the sets of categories of *X* generated during each step of the merge sequence, and find the one for which the *p*-value in step 3 is the smallest. That set is the set of merged categories for *X* to be used in determining the split at the current node.

Splitting Nodes

When categories have been merged for all predictor fields, each field is evaluated for its association with the target field, based on the adjusted *p*-value of the statistical test of association, as described below.

The predictor with the strongest association, indicated by the smallest adjusted *p*-value, is compared to the split threshold, α_{split}. If the *p*-value is less than or equal to α_{split}, that field is selected as the split field for the current node. Each of the merged categories of the split field defines a child node of the split.

After the split is applied to the current node, the child nodes are examined to see if they warrant splitting by applying the merge/split process to each in turn. Processing proceeds recursively until one or more stopping rules are triggered for every unsplit node, and no further splits can be made.

Statistical Tests Used

Calculations of the unadjusted *p*-values depend on the type of the target field. During the merge step, categories are compared pairwise, that is, one (possibly compound) category is compared against another (possibly compound) category. For such comparisons, only records belonging to one of the comparison categories in the current node are considered. During the split step, all categories are considered in calculating the *p*-value, thus all records in the current node are used.

Scale Target Field (F Test).

For models with a scale-level target field, the *p*-value is calculated based on a standard ANOVA *F*-test comparing the target field means across categories of the predictor field under consideration. The *F* statistic is calculated as

$$F = \frac{\sum_{i=1}^{I} \sum_{n \in D} w_n f_n I\left(x_n = i\right)\left(\overline{y}_i - \overline{y}\right)^2 / (I-1)}{\sum_{i=1}^{I} \sum_{n \in D} w_n f_n I\left(x_n = i\right)\left(y_n - \overline{y}_i\right)^2 / \left(N_f - I\right)}$$

and the *p*-value is

$$p = \Pr\left(F\left(I-1, N_f - I\right) > F\right)$$

where

$$\overline{y}_i = \frac{\sum_{n \in D} w_n f_n y_n I(x_n = i)}{\sum_{n \in D} w_n f_n I(x_n = i)}, \overline{y} = \frac{\sum_{n \in D} w_n f_n y_n}{\sum_{n \in D} w_n f_n}, N_f = \sum_{n \in D} f_n$$

and $F(I-1, N_f-I)$ is a random variable following an F-distribution with $(I-1)$ and (N_f-I) degrees of freedom.

Nominal Target Field (Chi-Squared Test)

If the target field Y is a set (categorical) field, the null hypothesis of independence of X and Y is tested. To do the test, a contingency (count) table is formed using classes of Y as columns and categories of the predictor X as rows. The expected cell frequencies under the null hypothesis of independence are estimated. The observed cell frequencies and the expected cell frequencies are used to calculate the chi-squared statistic, and the p-value is based on the calculated statistic.

Pearson Chi-squared test

The Pearson chi-square statistic is calculated as

$$X^2 = \sum_{j=1}^{J} \sum_{i=1}^{I} \frac{(n_{ij} - \hat{m}_{ij})^2}{\hat{m}_{ij}}$$

where $n_{ij} = \sum_n f_n I(x_n = i \wedge y_n = j)$ is the observed cell frequency and $\hat{m}_{ij}$ is the expected cell frequency for cell $(x_n = i, y_n = j)$ from the independence model as described below. The corresponding p value is calculated as $p = \Pr\left(\chi_d^2 > X^2\right)$, where χ_d^2 follows a chi-square distribution with $d = (J-1)(I-1)$ degrees of freedom.

Expected Frequencies for Chi-Square Test

Likelihood-ratio Chi-squared test

The likelihood-ratio chi-square is calculated based on the expected and observed frequencies, as described above. The likelihood ratio chi-square is calculated as

$$G^2 = 2 \sum_{j=1}^{J} \sum_{i=1}^{I} n_{ij} \ln\left(n_{ij} / \hat{m}_{ij}\right)$$

and the p-value is calculated as $p = \Pr\left(\chi_d^2 > G^2\right)$

Expected frequencies for chi-squared tests

For models with no case weights, expected frequencies are calculated as

$$\hat{m}_{ij} = \frac{n_{i.} n_{.j}}{n_{..}}$$

where

$$n_{i.} = \sum_{j=1}^{J} n_{ij},\ n_{.j} = \sum_{i=1}^{I} n_{ij},\ n_{..} = \sum_{j=1}^{J} \sum_{i=1}^{I} n_{ij}.$$

If case weights are specified, the expected cell frequency under the null hypothesis of independence takes the form

$$m_{ij} = \overline{w}_{ij}^{-1} \alpha_i \beta_j$$

where α_{i} and βj are parameters to be estimated, and

$$\overline{w}_{ij} = \frac{w_{ij}}{n_{ij}},\ w_{ij} = \sum_{n \in D} w_n f_n I(x = i \wedge y_n = j).$$

The parameter estimates $\hat{\alpha}_i$, $\hat{\beta}_j$, and hence $\hat{m}_{ij}$, are calculated based on the following iterative procedure:

1. Initially, k = 0, $\alpha_i^{(0)} = \beta_j^{(0)} = 1$, $m_{ij}^{(0)} = \overline{w}_{ij}^{-1}$.
2. $\alpha_i^{(k+1)} = \frac{n_{i.}}{\sum_J \overline{w}_{ij}^{-1} \beta_j^{(k)}} = \alpha_i^{(k)} \frac{n_{i.}}{\sum_j m_{ij}^{(k)}}$.
3. $\beta_j^{(k+1)} = \frac{n_{.j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k+1)}}$
4. $m_{ij}^{(k+1)} = \tilde{w}_{ij}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)}$.
5. If $\max_{i,j} \left| m_{ij}^{(k+1)} - m_{ij}^{(k)} \right| < \epsilon$, stop and output $\alpha_i^{(k+1)}$, $\beta_j^{(k+1)}$, and $m_{ij}^{(k+1)}$ as the final estimates of $\hat{\alpha}_i$, $\hat{\beta}_j$, and $\hat{m}_{ij}$. Otherwise, increment *k* and repeat from step 2.

Ordinal Target Field (Row Effects Model)

If the target field *Y* is ordinal, the null hypothesis of independence of *X* and *Y* is tested against the row effects model, with the rows being the categories of *X* and the columns the categories of *Y*(Goodman, 1979). Two sets of expected cell frequencies, $\hat{m}_{ij}$ (under the hypothesis of independence and $\hat{\hat{m}}_{ij}$ (under the hypothesis that the data follow the row effects model), are both estimated. The likelihood ratio statistic is computed as

$$H^2 = 2 \sum_{i=1}^{I} \sum_{j=1}^{J} \hat{\hat{m}}_{ij} \ln \left(\hat{\hat{m}}_{ij} / \hat{m}_{ij} \right)$$

and the *p*-value is calculated as

$$p = \Pr \left(\chi_{I-1}^2 > H^2 \right)$$

Expected Cell Frequencies for the Row Effects Model

For the row effects model, scores for categories of Y are needed. By default, the order of each category is used as the category score. Users can specify their own set of scores. The expected cell frequency under the row effects model is

$$m_{ij} = \overline{w}_{ij}^{-1} \alpha_i \beta_j \gamma_i^{(s_j - \overline{s})}$$

where s_j is the score for category *j* of *Y*, and

$$\overline{s} = \frac{\sum_{j=1}^{J} w_{\cdot j} s_j}{\sum_{j=1}^{J} w_{\cdot j}}$$

in which $w_{\cdot j} = \sum_i w_{ij}$, α_i, γ_j and γ_i are unknown parameters to be estimated.

Parameter estimates $\hat{\hat{\alpha}}_i$, $\hat{\hat{\beta}}_j$, $\hat{\hat{\gamma}}_i$, and hence $\hat{\hat{m}}_{ij}$ are calculated using the following iterative procedure:

1. $k = 0, \alpha_i^{(0)} = \beta_j^{(0)} = \gamma_i^{(0)} = 1, m_{ij}^{(0)} = \overline{w}_{ij}^{-1}$
2. $\alpha_i^{(k+1)} = \frac{n_{\cdot j}}{\sum_j \overline{w}_{ij}^{-1} \beta_j^{(k)} \left(\gamma_i^{(k)}\right)^{(s_j - \overline{s})}} = \alpha_i^{(k)} \frac{n_{i\cdot}}{\sum_j m_{ij}^{(k)}}$
3. $\beta_j^{(k+1)} = \frac{n_{\cdot j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k)} \left(\gamma_i^{(k)}\right)^{(s_j - \overline{s})}}$
4. $m_{ij}^{*} = \overline{w}_{ij}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)} \left(\gamma_i^{(k)}\right)^{(s_j - \overline{s})}, \; G_i = 1 + \frac{\sum_j (s_j - \overline{s})\left(n_{ij} - m_{ij}^{*}\right)}{\sum_j (s_j - \overline{s})^2 m_{ij}^{*}}$
5. $\gamma_i^{(k+1)} = \begin{cases} \gamma_i^{(k)} G_i & G_i > 0 \\ \gamma_i^{(k)} & \text{otherwise} \end{cases}$
6. $m_{ij}^{(k+1)} = \overline{w}_{ij}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)} \left(\gamma_i^{(k+1)}\right)^{(s_j - \overline{s})}$
7. If $\max_{i,j} \left| m_{ij}^{(k+1)} - m_{ij}^{(k)} \right| < \epsilon$, stop and set $\alpha_i^{(k+1)}$, $\beta_j^{(k+1)}$, $\gamma_i^{(k+1)}$, and $m_{ij}^{(k+1)}$ as the final estimates of $\hat{\hat{\alpha}}_i$, $\hat{\hat{\beta}}_j$, $\hat{\hat{\gamma}}_i$, and $\hat{\hat{m}}_{ij}$. Otherwise, increment *k* and repeat from step 2.

Bonferroni Adjustment

The **adjusted *p*-value** is calculated as the *p*-value times a Bonferroni multiplier. The Bonferroni multiplier controls the overall *p*-value across multiple statistical tests.

Suppose that a predictor field originally has *I* categories, and it is reduced to *r* categories after the merging step. The Bonferroni multiplier *B* is the number of possible ways that *I* categories can be merged into *r* categories. For *r* = *I*, *B* = 1. For 2 ≤ *r* < *I*,

$$B = \begin{cases} \binom{I-1}{r-1} & \text{Ordinal predictor} \\ \sum_{v=0}^{r-1} (-1)^v \frac{(r-v)^I}{v!(r-v)!} & \text{Nominal predictor} \\ \binom{I-2}{r-2} + r\binom{I-2}{r-1} & \text{Ordinal with a missing value} \end{cases}$$

Blank Handling

If the target field for a record is blank, or all the predictor fields are blank, the record is ignored in model building. If case weights are specified and the case weight for a record is blank, zero, or negative, the record is ignored, and likewise for frequency weights.

For other records, blanks in predictor fields are treated as an additional category for the field.

Ordinal Predictors

The algorithm first generates the best set of categories using all non-blank information. Then the algorithm identifies the category that is most similar to the blank category. Finally, two *p*-values are calculated: one for the set of categories formed by merging the blank category with its most similar category, and the other for the set of categories formed by adding the blank category as a separate category. The set of categories with the smallest *p*-value is used.

Nominal Predictors

The missing category is treated the same as other categories in the analysis.

Effect of Options

Stopping Rules

Stopping rules control how the algorithm decides when to stop splitting nodes in the tree. Tree growth proceeds until every leaf node in the tree triggers at least one stopping rule. Any of the following conditions will prevent a node from being split:

- The node is pure (all records have the same value for the target field)
- All records in the node have the same value for all predictor fields used by the model
- The tree depth for the current node (the number of recursive node splits defining the current node) is the *maximum tree depth* (default or user-specified).
- The number of records in the node is less than the *minumum parent node size* (default or user-specified)
- The number of records in any of the child nodes resulting from the node's best split is less than the *minimum child node size* (default or user-specified)
- The best split for the node yields a *p*-value that is greater than the α_{split} (default or user-specified).

Profits

Profits are numeric values associated with categories of a (symbolic) target field that can be used to estimate the gain or loss associated with a segment. They define the relative value of each value of the target field. Values are used in computing gains but not in tree growing.

Profit for each node in the tree is calculated as

$$\sum_j f_j(t)P_j$$

where *j* is the target field category, $f_{\mathrm{j}}(t)$ is the sum of frequency field values for all records in node *t* with category *j* for the target field, and P_{j} is the user-defined profit value for category *j*.

Score Values

Scores are available in CHAID and Exhaustive CHAID. They define the order and distance between categories of an ordinal categorical target field. In other words, the scores define the field's scale. Values of scores are involved in tree growing.

If user-specified scores are provided, they are used in calculation of expected cell frequencies, as described above.

Costs

Costs, if specified, are not taken into account in growing a CHAID tree. However, costs will be incorporated into node assignment and risk estimation, as described in Predicted Values and Risk Estimates, below.

Secondary Calculations

Secondary calculations are not directly related to building the model, but give you information about the model and its performance.

Risk Estimates

Risk estimates describe the risk of error in predicted values for specific nodes of the tree and for the tree as a whole.

Risk Estimates for Symbolic Target Field

For classification trees (with a symbolic target field), the risk estimate *r*(*t*) of a node *t* is computed as

$$r\left(t\right) = \frac{1}{N_f}\sum_j N_{f,j}(t)C(j^*(t)|j)$$

where $C(j^*(t)|j)$ is the misclassification cost of classifying a record with target value j as $j^*(t)$, $N_{\mathrm{f,j}}(t)$ is the sum of the frequency weights for records in node t in category j (or the number of records if no frequency weights are defined), and N_{f} is the sum of frequency weights for all records in the training data.

Note that case weights are not considered in calculating risk estimates.

Risk Estimates for numeric target field

For regression trees (with a numeric target field), the risk estimate $r(t)$ of a node t is computed as

$$r\,(t) = \frac{1}{N_f(t)} \sum_{i \in t} f_i (y_i - \overline{y}(t))^2$$

where f_{i} is the frequency weight for record i (a record assigned to node t), y_{i} is the value of the target field for record i, and $\overline{y}(t)$ is the weighted mean of the target field for all records in node t.

Tree Risk Estimate

For both classification trees and regression trees, the risk estimate $R(T)$ for the tree (T) is calculated by taking the sum of the risk estimates for the terminal nodes $r(t)$:

$$R(T) = \sum_{t \in T'} r(t)$$

where T' is the set of terminal nodes in the tree.

Gain Summary

The **gain summary** provides descriptive statistics for the terminal nodes of a tree.

If your target field is continuous (scale), the gain summary shows the weighted mean of the target value for each terminal node,

$$g(t) = \sum_{i \in t} w_i f_i x_i$$

If your target field is symbolic (categorical), the gain summary shows the weighted percentage of records in a selected target category,

$$g\,(t, j) = \frac{\sum_{i \in t} f_i x_i(j)}{\sum_{i \in t} f_i}$$

where $x_{\mathrm{i}}(j) = 1$ if record x_{i} is in target category j, and 0 otherwise. If profits are defined for the tree, the gain is the average profit value for each terminal node,

$$g(t) = \sum_{i \in t} f_i P(x_i)$$

where $P(x_i)$ is the profit value assigned to the target value observed in record x_i.

Generated Model/Scoring

Calculations done by the CHAID generated model are described below

Predicted Values

New records are scored by following the tree splits to a terminal node of the tree. Each terminal node has a particular predicted value associated with it, determined as follows:

Classification Trees

For trees with a symbolic target field, each terminal node's predicted category is the category with the lowest weighted cost for the node. This weighted cost is calculated as

$$\min_i \sum_j C(i|j)p(j|t)$$

where $C(i|j)$ is the user-specified misclassification cost for classifying a record as category *i* when it is actually category *j*, and $p(j|t)$ is the conditional weighted probability of a record being in category *j* given that it is in node *t*, defined as

$$p(j|t) = \frac{p(j,t)}{\sum_j p(j,t)}, p(j,t) = \pi(j)\frac{N_{w,j}(t)}{N_{w,j}}$$

where $\pi(j)$ is the prior probability for category *j*, $N_{w,j}(t)$ is the weighted number of records in node *t* with category *j* (or the number of records if no frequency or case weights are defined),

$$N_{w,j}(t) = \sum_{i \in t} w_i f_i j(i)$$

and $N_{w,j}$ is the weighted number records in category *j* (any node),

$$N_{w,j} = \sum_{i \in T} w_i f_i j(i)$$

Regression Trees

For trees with a numeric target field, each terminal node's predicted category is the weighted mean of the target values for records in the node. This weighted mean is calculated as

$$\overline{y}(t) = \frac{1}{N_w(t)} \sum_{i \in t} w_i f_i y_i$$

where $N_w(t)$ is defined as

$$N_w(t) = \sum_{i \in t} w_i f_i$$

Confidence

For classification trees, confidence values for records passed through the generated model are calculated as follows. For regression trees, no confidence value is assigned.

Classification Trees

Confidence for a scored record is the proportion of weighted records in the training data in the scored record's assigned terminal node that belong to the predicted category, modified by the Laplace correction:

$$\frac{N_{f,j}(t) + 1}{N_f(t) + k}$$

Note: If Surrogate Splitting is used (where a particular rule does not fit into a node) the Confidence score is reduced by multiplying it by 0.9. This can result in multiple Confidence scores being present within a single node.

Blank Handling

In classification of new records, blanks are handled as they are during tree growth, being treated as an additional category (possibly merged with other non-blank categories). For more information, see the topic "Blank Handling" on p. 81.

For nodes where there were no blanks in the training data, a blank category will not exist for the split of that node. In that case, records with a blank value for the split field are assigned a null value.

Cluster Evaluation Algorithms

This document describes measures used for evaluating clustering models.

- The **Silhouette coefficient** combines the concepts of cluster cohesion (favoring models which contain tightly cohesive clusters) and cluster separation (favoring models which contain highly separated clusters). It can be used to evaluate individual objects, clusters, and models.
- The **sum of squares error (SSE)** is a measure of prototype-based cohesion, while **sum of squares between (SSB)** is a measure of prototype-based separation.
- **Predictor importance** indicates how well the variable can differentiate different clusters. For both range (numeric) and discrete variables, the higher the importance measure, the less likely the variation for a variable between clusters is due to chance and more likely due to some underlying difference.

Notation

The following notation is used throughout this chapter unless otherwise stated:

x_{ik}	Continuous variable *k* in case *i* (standardized).
x_{iks}	The *s*th category of variable *k* in case *i* (one-of-c coding).
N	Total number of valid cases.
N_j	The number of cases in cluster *j*.
Y	Variable with *J* cluster labels.
μ_{jk}	The centroid of cluster *j* for variable *k*.
D_{ij}	The distance between case *i* and the centroid of cluster *j*.
D_j	The distance between the overall mean u and the centroid of cluster *j*.

Goodness Measures

The average Silhouette coefficient is simply the average over all cases of the following calculation for each individual case:

$$(B - A) \, / \max\left(A, B\right)$$

where *A* is the average distance from the case to every other case assigned to the same cluster and *B* is the minimal average distance from the case to cases of a different cluster across all clusters.

Unfortunately, this coefficient is computationally expensive. In order to ease this burden, we use the following definitions of *A* and *B*:

- *A* is the distance from the case to the centroid of the cluster which the case belongs to;
- *B* is the minimal distance from the case to the centroid of every other cluster.

Distances may be calculated using Euclidean distances. The Silhouette coefficient and its average range between −1, indicating a very poor model, and 1, indicating an excellent model. As found by Kaufman and Rousseeuw (1990), an average silhouette greater than 0.5 indicates reasonable partitioning of data; less than 0.2 means that the data do not exhibit cluster structure.

Data Preparation

Before calculating Silhouette coefficient, we need to transform cases as follows:

1. **Recode categorical variables using one-of-c coding.** If a variable has *c* categories, then it is stored as *c* vectors, with the first category denoted (1,0,...,0), the next category (0,1,0,...,0), ..., and the final category (0,0,...,0,1). The order of the categories is based on the ascending sort or lexical order of the data values.
2. **Rescale continuous variables.** Continuous variables are normalized to the interval [−1, 1] using the transformation [2*(x−min)/(max−min)]−1. This normalization tries to equalize the contributions of continuous and categorical features to the distance computations.

Basic Statistics

The following statistics are collected in order to compute the goodness measures: the centroid μ_{jk} of variable *k* for cluster *j*, the distance between a case and the centroid, and the overall mean *u*.

For μ_{jk} with an ordinal or continuous variable *k*, we average all standardized values of variable *k* within cluster *j*. For nominal variables, μ_{jk} is a vector $\{\varphi_{jks}\}$ of probabilities of occurrence for each state *s* of variable *k* for cluster *j*. Note that in counting , we do not consider cases with missing values in variable *k*. If the value of variable *k* is missing for all cases within cluster *j*, μ_{jk} is marked as missing.

The distance D_{ij}^2 between case *i* and the centroid of cluster *j* can be calculated in terms of the weighted sum of the distance components d_{ijk}^2 across all variables; that is

$$D_{ij}^2 = \frac{\Sigma_k w_{ijk} d_{ijk}^2}{\Sigma_k w_{ijk}}$$

where w_{ijk} denotes a weight. At this point, we do not consider differential weights, thus w_{ijk} equals 1 if the variable *k* in case *i* is valid, 0 if not. If all w_{ijk} equal 0, set $D_{ij}^2 = 0$.

The distance component d_{ijk}^2 is calculated as follows for ordinal and continuous variables

$$d_{ijk}^2 = \left(x_{ik} - \mu_{jk}\right)^2$$

For binary or nominal variables, it is

$$d_{ijk}^2 = \frac{1}{S_k} \sum_{s=1}^{S_k} \left(x_{iks} - \varphi_{jks}\right)^2$$

where variable *k* uses one-of-c coding, and S_k is the number of its states.

The calculation of D_j is the same as that of D_{ij}, but the overall mean *u* is used in place of μ_{jk} and μ_{jk} is used in place of x_{ik}.

Silhouette Coefficient

The Silhouette coefficient of case *i* is

$$\frac{\min\left\{D_{ij}, j \in C_{-i}\right\} - D_{ic_i}}{\max\left(\min\left\{D_{ij}, j \in C_{-i}\right\}, D_{ic_i}\right)}$$

where C_{-i} denotes cluster labels which do not include case *i* as a member, while c_i is the cluster label which includes case *i*. If $\max\left(\min\left\{D_{ij}, j \in C_{-i}\right\}, D_{ic_i}\right)$ equals 0, the Silhouette of case *i* is not used in the average operations.

Based on these individual data, the total average Silhouette coefficient is:

$$SC = \frac{1}{N}\sum_{i=1}^{N} \frac{\min\left\{D_{ij}, j \in C_{-i}\right\} - D_{ic_i}}{\max\left(\min\left\{D_{ij}, j \in C_{-i}\right\}, D_{ic_i}\right)}$$

Sum of Squares Error (SSE)

SSE is a prototype-based cohesion measure where the squared Euclidean distance is used. In order to compare between models, we will use the averaged form, defined as:

$$\text{Average SSE} = \frac{1}{N}\sum_{j \in C}\sum_{i \in j} D_{ij}^2$$

Sum of Squares Between (SSB)

SSB is a prototype-based separation measure where the squared Euclidean distance is used. In order to compare between models, we will use the averaged form, defined as:

$$\text{Average SSB} = \frac{1}{N}\sum_{j \in C} N_j D_j^2$$

Predictor Importance

The importance of field *i* is defined as

$$VI_i = \frac{-\log_{10}\left(sig_i\right)}{\max_{j \in \Omega}\left(-\log_{10}\left(sig_j\right)\right)}$$

where Ω denotes the set of predictor and evaluation fields, sig_i is the significance or *p*-value computed from applying a certain test, as described below. If sig_i equals zero, set $sig_i = MinDouble$, where *MinDouble* is the minimal double value.

Across Clusters

The *p*-value for **categorical** fields is based on Pearson's chi-square. It is calculated by

p-value = Prob($\chi_d^2 > X^2$),

where

$$X^2 = \sum_{i=1}^{I}\sum_{j=1}^{J}\left(N_{ij} - \hat{N}_{ij}\right)^2/\hat{N}_{ij}$$

where $\hat{N}_{ij} = N_{i\cdot}N_{\cdot j}/N\left(X\right)$.

- If $N\left(X\right) = 0$, the importance is set to be undefined or unknown;
- If $N_{i\cdot} = 0$, subtract one from *I* for each such category to obtain $I^{'}$;
- If $N_{\cdot j} = 0$, subtract one from *J* for each such cluster to obtain $J^{'}$;
- If $J^{'} \leq 1$ or $I^{'} \leq 1$, the importance is set to be undefined or unknown.

The degrees of freedom are $\left(I^{'} - 1\right)\left(J^{'} - 1\right)$.

The *p*-value for **continuous** fields is based on an *F* test. It is calculated by

p-value = Prob$\{F\left(J-1, N-J\right) > F\}$,

where

$$F = \frac{\sum_{j=1}^{J} N_j\left(\overline{x}_j - \overline{\overline{x}}\right)^2/(J-1)}{\sum_{j=1}^{J}\left(N_j - 1\right)s_j/(N-J)}$$

- If *N*=0, the importance is set to be undefined or unknown;
- If $N_j = 0$, subtract one from *J* for each such cluster to obtain $J^{'}$;
- If $J^{'} \leq 1$ or $N \leq J^{'}$, the importance is set to be undefined or unknown;
- If the denominator in the formula for the *F* statistic is zero, the importance is set to be undefined or unknown;
- If the numerator in the formula for the *F* statistic is zero, set *p*-value = 1;

The degrees of freedom are $\left(J^{'} - 1, N - J^{'}\right)$.

Within Clusters

The null hypothesis for **categorical** fields is that the proportion of cases in the categories in cluster *j* is the same as the overall proportion.

The chi-square statistic for cluster *j* is computed as follows

$$X^2 = \sum_{i=1}^{I} \frac{\left(N_{ij} - N_j p_i\right)^2}{N_j p_i}$$

If $N_j = 0$, the importance is set to be undefined or unknown;

If $p_i = 0$, subtract one from *I* for each such category to obtain $I^{'}$;

If $I^{'} \leq 1$, the importance is set to be undefined or unknown.

The degrees of freedom are $d = I^{'} - 1$.

The null hypothesis for **continuous** fields is that the mean in cluster *j* is the same as the overall mean.

The Student's *t* statistic for cluster *j* is computed as follows

$$t = \frac{\left(\overline{x}_j - \overline{\overline{x}}\right)}{s_j / \sqrt{N_j}}$$

with $d = N_j - 1$ degrees of freedom.

If $N_j \leq 1$ or $s_j = 0$, the importance is set to be undefined or unknown;

If the numerator is zero, set *p*-value = 1;

Here, the *p*-value based on Student's *t* distribution is calculated as

p-value = 1 − Prob$\{|T\,(d)| \leq |t|\}$.

References

Kaufman, L., and P. J. Rousseeuw. 1990. *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley and Sons.

Tan, P., M. Steinbach, and V. Kumar. 2006. *Introduction to Data Mining*. : Addison-Wesley.

COXREG Algorithms

Cox Regression Algorithms

Cox (1972) first suggested the models in which factors related to lifetime have a multiplicative effect on the hazard function. These models are called proportional hazards models. Under the proportional hazards assumption, the hazard function *h* of *t* given *X* is of the form

$$h(t|\mathbf{x}) = h_0(t)e^{\mathbf{x}'\beta}$$

where $\mathbf{x}$ is a known vector of regressor variables associated with the individual, β is a vector of unknown parameters, and $h_0(t)$ is the baseline hazard function for an individual with $\mathbf{x} = 0$. Hence, for any two covariates sets $\mathbf{x}_1$ and $\mathbf{x}_2$, the log hazard functions $h(t|\mathbf{x}_1)$ and $h(t|\mathbf{x}_2)$ should be parallel across time.

When a factor does not affect the hazard function multiplicatively, stratification may be useful in model building. Suppose that individuals can be assigned to one of *m* different strata, defined by the levels of one or more factors. The hazard function for an individual in the *j*th stratum is defined as

$$h_j(t|\mathbf{x}) = h_{0j}(t)e^{\mathbf{x}'\beta}$$

There are two unknown components in the model: the regression parameter β and the baseline hazard function $h_{0j}(t)$. The estimation for the parameters is described below.

Estimation

We begin by considering a nonnegative random variable *T* representing the lifetimes of individuals in some population. Let $f(t|\mathbf{x})$ denote the probability density function (pdf) of *T* given a regressor *x* and let $S(t|\mathbf{x})$ be the survivor function (the probability of an individual surviving until time *t*). Hence

$$S(t|\mathbf{x}) = \int_t^\infty f(u|\mathbf{x})du$$

The hazard $h(t|\mathbf{x})$ is then defined by

$$h(t|\mathbf{x}) = \tfrac{f(t|\mathbf{x})}{S(t|\mathbf{x})}$$

Another useful expression for $S(t|\mathbf{x})$ in terms of $h(t|\mathbf{x})$ is

$$S(t|\mathbf{x}) = \exp\left(-\int_0^t h(u|\mathbf{x})du\right)$$

Thus,

$$\ln S(t|\mathbf{x}) = -\int_0^t h(u|\mathbf{x})du$$

For some purposes, it is also useful to define the cumulative hazard function

$$H(t|\mathbf{x}) = \int_0^t h(u|\mathbf{x})du = -\ln S(t|\mathbf{x})$$

Under the proportional hazard assumption, the survivor function can be written as

$$S(t|\mathbf{x}) = [S_0(t)]^{\exp\left(\mathbf{x}'\beta\right)}$$

where $S_0(t)$ is the baseline survivor function defined by

$$S_0(t) = \exp\left(-H_0(t)\right)$$

and

$$H_0(t) = \int_0^t h_0(u)du$$

Some relationships between $S(t|\mathbf{x})$, $H(t|\mathbf{x})$ and $H_0(t)$, $S_0(t)$ and $h_0(t)$ which will be used later are

$$\ln S(t|\mathbf{x}) = -H(t|\mathbf{x}) = -\exp\left(\mathbf{x}'\beta\right)H_0(t)$$

$$\ln\left(-\ln S(t|\mathbf{x})\right) = \mathbf{x}'\beta + \ln H_0(t)$$

To estimate the survivor function $S(t|\mathbf{x})$, we can see from the equation for the survivor function that there are two components, β and $S_0(t)$, which need to be estimated. The approach we use here is to estimate β from the partial likelihood function and then to maximize the full likelihood for $S_0(t)$.

Estimation of Beta

Assume that

- There are *m* levels for the stratification variable.
- Individuals in the same stratum have proportional hazard functions.
- The relative effect of the regressor variables is the same in each stratum.

Let $t_{j1} < \cdots < t_{jk_j}$ be the observed uncensored failure time of the n_j individuals in the *j*th stratum and $x_{j1}, \ldots, x_{jk_j}$ be the corresponding covariates. Then the partial likelihood function is defined by

$$L(\beta) = \prod_{j=1}^{m}\prod_{i=1}^{k_j} \frac{e^{\mathbf{S}'_{ji}\beta}}{\left(\sum_{l\in R_{ji}} w_l e^{\mathbf{x}'_l\beta}\right)^{d_{ji}}}$$

where d_{ji} is the sum of case weights of individuals whose lifetime is equal to t_{ji} and $\mathbf{S}_{ji}$ is the weighted sum of the regression vector **x** for those d_{ji} individuals, w_l is the case weight of individual *l*, and R_{ji} is the set of individuals alive and uncensored just prior to t_{ji} in the *j*th stratum. Thus the log-likelihood arising from the partial likelihood function is

$$l = \ln L(\beta) = \sum_{j=1}^{m}\sum_{i=1}^{k_j} \mathbf{S}'_{ji}\beta - \sum_{j=1}^{m}\sum_{i=1}^{k_j} d_{ji} \ln\left(\sum_{l\in R_{ji}} w_l e^{\mathbf{x}'_l\beta}\right)$$

and the first derivatives of *l* are

$$D_{\beta_r} = \frac{\partial l}{\partial \beta_r} = \sum_{j=1}^{m}\sum_{i=1}^{k_j}\left(S_{ji}^{(r)} - d_{ji}\frac{\sum\limits_{l\in R_{ji}} w_l x_{lr} e^{\mathbf{x}'_l\beta}}{\sum\limits_{l\in R_{ji}} w_l e^{\mathbf{x}'_l\beta}}\right), \quad r = 1, \ldots, p$$

$S_{ji}^{(r)}$ is the *r*th component of $\mathbf{S}_{ji} = \left(S_{ji}^{(1)}, \ldots, S_{ji}^{(p)}\right)'$. The maximum partial likelihood estimate (MPLE) of β is obtained by setting $\frac{\partial l}{\partial \beta_r}$ equal to zero for $r = 1, \ldots, p$, where p is the number of independent variables in the model. The equations $\frac{\partial l}{\partial \beta_r} = 0 \quad (r = 1, \ldots, p)$ can usually be solved by using the Newton-Raphson method.

Note that from its equation that the partial likelihood function $L(\beta)$ is invariant under translation. All the covariates are centered by their corresponding overall mean. The overall mean of a covariate is defined as the sum of the product of weight and covariate for all the censored and uncensored cases in each stratum. For notational simplicity, $\mathbf{x}_l$ used in the Estimation Section denotes centered covariates.

Three convergence criteria for the Newton-Raphson method are available:

- Absolute value of the largest difference in parameter estimates between iterations (δ) divided by the value of the parameter estimate for the previous iteration; that is,

  $\text{BCON} = \left|\frac{\delta}{\text{parameter estimate for previous iteration}}\right|$
- Absolute difference of the log-likelihood function between iterations divided by the log-likelihood function for previous iteration.
- Maximum number of iterations.

The asymptotic covariance matrix for the MPLE $\hat{\beta} = \left(\hat{\beta}_1, ..., \hat{\beta}_p\right)'$ is estimated by $\mathbf{I}^{-1}$ where $\mathbf{I}$ is the information matrix containing minus the second partial derivatives of $\ln L$. The (*r*, *s*)-th element of $\mathbf{I}$ is defined by

$$\begin{aligned}\mathbf{I}_{rs} &= -E\frac{\partial^2}{\partial\beta_r\partial\beta_s}\ln L \\ &= \sum_{j=1}^{m}\sum_{i=1}^{k_j} d_{ji}\left[\frac{\sum\limits_{l\in R_{ji}} w_l x_{ls} x_{lr} e^{\mathbf{x}'_l\beta}}{\sum\limits_{l\in R_{ji}} w_l e^{\mathbf{x}'_l\beta}} - \frac{\left(\sum\limits_{l\in R_{ji}} w_l x_{lr} e^{\mathbf{x}'_l\beta}\right)\left(\sum\limits_{l\in R_{ji}} w_l x_{ls} e^{\mathbf{x}'_l\beta}\right)}{\left(\sum\limits_{l\in R_{ji}} w_l e^{\mathbf{x}'_l\beta}\right)^2}\right]\end{aligned}$$

We can also write $\mathbf{I}$ in a matrix form as

$$I_{rs} = \sum_{j=1}^{m}\sum_{i=1}^{k_j} d_{ji}\left(x'(t_{ji})\right)V(t_{ji})(x(t_{ji}))$$

where $\mathbf{x}(t_{ji})$ is a $n_{ji} \times p$ matrix which represents the p covariate variables in the model evaluated at time t_{ji}, n_{ji} is the number of distinct individuals in R_{ji}, and $\mathbf{V}(t_{ji})$ is a $n_{ji} \times n_{ji}$ matrix with the *l*th diagonal element $v_{ll}(t_{ji})$ defined by

$$v_{ll}(t_{ji}) = p_l(t_{ji})w_l - (w_l p_l(t_{ji}))^2$$

$$p_l(t_{ji}) = \frac{\exp\left(\mathbf{x}'_l\hat{\beta}\right)}{\sum\limits_{h\in R_{ji}} w_h \exp\left(\mathbf{x}'_h\hat{\beta}\right)}$$

and the (*l*, *k*) element $v_{lk}(t_{ji})$ defined by

$$v_{lk}(t_{ji}) = w_l p_l(t_{ji}) \times w_k p_k(t_{ji})$$

Estimation of the Baseline Function

After the MPLE $\hat{\beta}$ of β is found, the baseline survivor function $S_{0j}(t)$ is estimated separately for each stratum. Assume that, for a stratum, $t_1 < \cdots < t_k$ are observed lifetimes in the sample. There are n_i at risk and d_i deaths at t_i, and in the interval $[t_{i-1}, t_i)$ there are λ_i censored times. Since $S_0(t)$ is a survivor function, it is non-increasing and left continuous, and thus $\hat{S}_0(t)$ must be constant except for jumps at the observed lifetimes $t_1, \ldots, t_k$.

Further, it follows that

$$\hat{S}_0(t_1) = 1$$

and

$$\hat{S}_0(t_i+) = \hat{S}_0(t_{i+1})$$

Writing $\hat{S}_0(t_i+) = p_i (i = 1, \ldots, k)$, the observed likelihood function is of the form

$$L_1 = \prod_{i=1}^{k} \left\{ \prod_{l\in D_i} \left(p_{i-1}^{\exp\left(\mathbf{x}'_l\beta\right)} - p_i^{\exp\left(\mathbf{x}'_l\beta\right)} \right)^{w_l} \prod_{l\in C_i} \left(p_{i-1}^{\exp\left(\mathbf{x}'_l\beta\right)} \right)^{w_l} \right\} \prod_{l\in C_{k+1}} \left(p_k^{\exp\left(\mathbf{x}'_l\beta\right)} \right)^{w_l}$$

where D_i is the set of individuals dying at t_i and C_i is the set of individuals with censored times in $[t_{i-1}, t_i)$. (Note that if the last observation is uncensored, C_{k+1} is empty and $p_k = 0$)

If we let $\alpha_i = p_i/p_{i-1} (i = 1, \ldots, k)$, L_1 can be written as

$$L_1 = \prod_{i=1}^{k} \prod_{l\in D_i} \left(1 - \alpha_i^{\exp\left(\mathbf{x}'_l\beta\right)}\right)^{w_l} \prod_{l\in R_i - D_i} \alpha_i^{w_l \exp\left(\mathbf{x}'_l\beta\right)}$$

Differentiating $\ln L_1$ with respect to $\alpha_1, \ldots, \alpha_k$ and setting the equations equal to zero, we get

$$\sum_{l\in D_i} \frac{w_l \exp\left(\mathbf{x}'_l\beta\right)}{1 - \alpha_i^{\exp\left(\mathbf{x}'_l\beta\right)}} = \sum_{l\in R_i} w_l \exp\left(\mathbf{x}'_l\beta\right) \quad i = 1, \ldots, k$$

We then plug the MPLE $\hat{\beta}$ of β into this equation and solve these *k* equations separately.

There are two things worth noting:

- If any $|D_i| = 1$, $\hat{\alpha}_i$ can be solved explicitly.

$$\hat{\alpha}_i = \left[1 - \frac{w_i \exp\left(\mathbf{x}'_i\hat{\beta}\right)}{\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l\hat{\beta}\right)} \right]^{\exp\left(-\mathbf{x}'_l\overline{\beta}\right)}$$

- If $|D_i| > 1$, the equation for the cumulative hazard function must be solved iteratively for $\hat{\alpha}_i$. A good initial value for $\hat{\alpha}_i$ is

$$\hat{\alpha}_i = \exp\left(\frac{-d_i}{\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l\hat{\beta}\right)} \right)$$

  where $d_i = \sum_{l \in D_i} w_l$ is the weight sum for set D_i. (See Lawless, 1982, p. 361.)

Once the $\hat{\alpha}_i$, $i = 1, \ldots, k$ are found, $S_0(t)$ is estimated by

$$\hat{S}_0(t) = \prod_{i:(t_i<t)} \hat{\alpha}_i$$

Since the above estimate of $S_0(t)$ requires some iterative calculations when ties exist, Breslow (1974) suggests using the equation for α_i when $|D_i| > 1$ as an estimate; however, we will use this as an initial estimate.

The asymptotic variance for $-\ln\hat{S}_0(t)$ can be found in Chapter 4 of Kalbfleisch and Prentice (1980). At a specified time *t*, it is consistently estimated by

$$var\left(-\ln\hat{S}_0(t)\right) = \sum_{t_i<t} |D_i| \left(\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l\hat{\beta}\right) \right)^{-2} + \mathbf{a}'\mathbf{I}^{-1}a$$

where a is a *p*×1 vector with the *j*th element defined by

$$\sum_{t_i<t} |D_i| \frac{\sum_{l \in R_i} w_l x_{lj} \exp\left(\mathbf{x}'_l\hat{\beta}\right)}{\left(\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l\hat{\beta}\right) \right)^2}$$

and **I** is the information matrix. The asymptotic variance of $\hat{S}(t|x)$ is estimated by

$$e^{2\mathbf{x}'\hat{\beta}} \left(\hat{S}(t|\mathbf{x})\right)^2 var\left(-\ln\hat{S}_0(t)\right)$$

Selection Statistics for Stepwise Methods

The same methods for variable selection are offered as in binary logistic regression. For more information, see the topic "Stepwise Variable Selection " on p. 281. Here we will only define the three removal statistics—Wald, LR, and Conditional—and the Score entry statistic.

Score Statistic

The score statistic is calculated for every variable not in the model to decide which variable should be added to the model. First we compute the information matrix **I** for all eligible variables based on the parameter estimates for the variables in the model and zero parameter estimates for the variables not in the model. Then we partition the resulting **I** into four submatrices as follows:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

where $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are square matrices for variables in the model and variables not in the model, respectively, and $\mathbf{A}_{12}$ is the cross-product matrix for variables in and out. The score statistic for variable $\mathbf{x}_i$ is defined by

$$\mathbf{D}'_{x_i}\mathbf{B}_{22,i}\mathbf{D}_{x_i}$$

where $\mathbf{D}_{x_i}$ is the first derivative of the log-likelihood with respect to all the parameters associated with $\mathbf{x}_i$ and $\mathbf{B}_{22,i}$ is equal to $\left(\mathbf{A}_{22,i} - \mathbf{A}_{21,i}\mathbf{A}_{11}^{-1}\mathbf{A}_{12,i}\right)^{-1}$, and $\mathbf{A}_{22,i}$ and $\mathbf{A}_{12,i}$ are the submatrices in $\mathbf{A}_{22}$ and $\mathbf{A}_{12}$ associated with variable $\mathbf{x}_i$.

Wald Statistic

The Wald statistic is calculated for the variables in the model to select variables for removal. The Wald statistic for variable $\mathbf{x}_j$ is defined by

$$\hat{\beta}'_j\mathbf{B}_{11,j}\hat{\beta}_j$$

where $\hat{\beta}_j$ is the parameter estimate associated with $\mathbf{x}_j$ and $\mathbf{B}_{11,j}$ is the submatrix of $\mathbf{A}_{11}^{-1}$ associated with $\mathbf{x}_j$.

LR (Likelihood Ratio) Statistic

The LR statistic is defined as twice the log of the ratio of the likelihood functions of two models evaluated at their own MPLES. Assume that *r* variables are in the current model and let us call the current model the full model. Based on the MPLES of parameters for the full model, *l(full)* is defined in "Estimation of Beta ". For each of *r* variables deleted from the full model, MPLES are found and the reduced log-likelihood function, *l(reduced)*, is calculated. Then LR statistic is defined as

–2(l(reduced) – l(full))

Conditional Statistic

The conditional statistic is also computed for every variable in the model. The formula for conditional statistic is the same as LR statistic except that the parameter estimates for each reduced model are conditional estimates, not MPLES. The conditional estimates are defined as follows. Let $\hat{\beta}=\left(\hat{\beta}_1,\ldots,\hat{\beta}_r\right)'$ be the MPLES for the *r* variables (blocks) and *C* be the asymptotic covariance for the parameters left in the model given $\hat{\beta}_i$ is

$$\tilde{\beta}_{(i)}=\hat{\beta}_{(i)}-\mathbf{C}_{12}^{(i)}\left(\mathbf{C}_{22}^{(i)}\right)^{-1}\hat{\beta}_i$$

where $\hat{\beta}_i$ is the MPLE for the parameter(s) associated with $\mathbf{x}_i$ and $\hat{\beta}_{(i)}$ is $\hat{\beta}$ without $\hat{\beta}_i$, $\mathbf{C}_{12}^{(i)}$ is the covariance between the parameter estimates left in the model $\hat{\beta}_{(i)}$ and $\hat{\beta}_i$, and $\mathbf{C}_{22}^{(i)}$ is the covariance of $\hat{\beta}_i$. Then the conditional statistic for variable $\mathbf{x}_i$ is defined by

$$-2\left(l\left(\mathrm{b}_{(i)}\right)-l(full)\right)$$

where $l\left(\tilde{\beta}_{(i)}\right)$ is the log-likelihood function evaluated at $\tilde{\beta}_{(i)}$.

Note that all these four statistics have a chi-square distribution with degrees of freedom equal to the number of parameters the corresponding model has.

Statistics

The following output statistics are available.

Initial Model Information

The initial model for the first method is for a model that does not include covariates. The log-likelihood function *l* is equal to

$$l(0)=-\sum_{j=1}^{m}\sum_{i=1}^{k_j}d_{ji}\ln\left(n_{ji}^{*}\right)$$

where n_{ji}^{*} is the sum of weights of individuals in set R_{ji}.

Model Information

When a stepwise method is requested, at each step, the −2 log-likelihood function and three chi-square statistics (model chi-square, improvement chi-square, and overall chi-square) and their corresponding degrees of freedom and significance are printed.

–2 Log-Likelihood

$$-2\sum_{j=1}^{m}\sum_{i=1}^{k_j}\left(\mathbf{s}'_{ji}\hat{\beta}-d_{ji}\ln\left(\sum_{l\in R_{ji}}w_l\exp\left(\mathbf{x}'_l\hat{\beta}\right)\right)\right)$$

where $\hat{\beta}$ is the MPLE of β for the current model.

Improvement Chi-Square

(–2 log-likelihood function for previous model) – (–2 log-likelihood function for current model).

The previous model is the model from the last step. The degrees of freedom are equal to the absolute value of the difference between the number of parameters estimated in these two models.

Model Chi-Square

(–2 log-likelihood function for initial model) – (–2 log-likelihood function for current model).

The initial model is the final model from the previous method. The degrees of freedom are equal to the absolute value of the difference between the number of parameters estimated in these two model.

Note: The values of the model chi-square and improvement chi-square can be less than or equal to zero. If the degrees of freedom are equal to zero, the chi-square is not printed.

Overall Chi-Square

The overall chi-square statistic tests the hypothesis that all regression coefficients for the variables in the model are identically zero. This statistic is defined as

$$\mathbf{u}^{'}(0)\mathbf{I}^{-1}\mathbf{u}(0)$$

where $\mathbf{u}(0)$ represents the vector of first derivatives of the partial log-likelihood function evaluated at $\beta = 0$. The elements of **u** and **I** are defined in “Estimation of Beta ”.

Information for Variables in the Equation

For each of the single variables in the equation, MPLE, SE for MPLE, Wald statistic, and its corresponding *df*, significance, and partial *R* are given. For a single variable, *R* is defined by

$$R = \left[\frac{\text{Wald}_{-2}}{\text{–2 log-likelihood for the intial model}}\right]^{1/2} \times \text{sign of MPLE}$$

if Wald $>$ 2. Otherwise *R* is set to zero. For a multiple category variable, only the Wald statistic, *df*, significance, and partial *R* are printed, where *R* is defined by

$$R = \left[\frac{\text{Wald}_{-2*}\text{df}}{\text{–2 log-likelihood for the intial model}}\right]^{1/2}$$

if Wald $>$ 2df. Otherwise *R* is set to zero.

Information for the Variables Not in the Equation

For each of the variables not in the equation, the Score statistic is calculated and its corresponding degrees of freedom, significance, and partial *R* are printed. The partial *R* for variables not in the equation is defined similarly to the *R* for the variables in the equation by changing the Wald statistic to the Score statistic.

There is one overall statistic called the residual chi-square. This statistic tests if all regression coefficients for the variables not in the equation are zero. It is defined by

$$\mathbf{u}'\left(\hat{\beta}\right)\mathbf{B}_{22}\mathbf{u}\left(\hat{\beta}\right)$$

where $\mathbf{u}\left(\hat{\beta}\right)$ is the vector of first derivatives of the partial log-likelihood function with respect to all the parameters not in the equation evaluated at MPLE $\hat{\beta}$ and $\mathbf{B}_{22}$ is equal to $\left(\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}$ and $\mathbf{A}$ is defined in "Score Statistic ".

Survival Table

For each stratum, the estimates of the baseline cumulative survival (S_0) and hazard (H_0) function and their standard errors are computed. $H_0(t)$ is estimated by

$$\hat{H}_0(t) = -\ln \hat{S}_0(t)$$

and the asymptotic variance of $\hat{H}_0(t)$ is defined in "Estimation of the Baseline Function ". Finally, the cumulative hazard function $H(t|\mathbf{x})$ and survival function $S(t|\mathbf{x})$ are estimated by

$$\hat{H}(t|\mathbf{x}) = \exp\left(\mathbf{x}'\hat{\beta}\right)\hat{H}_0(t)$$

and, for a given **x**,

$$\hat{S}(t|\mathbf{x}) = \left[\hat{S}_0(t)\right]^{\exp\left(\mathbf{x}'\hat{\beta}\right)}$$

The asymptotic variances are

$$var\left(\hat{H}(t|\mathbf{x})\right) = \exp\left(2\mathbf{x}'\hat{\beta}\right)var\left(\hat{H}_0(t)\right)$$

and

$$var\left(\hat{S}(t|\mathbf{x})\right) = \exp\left(2\mathbf{x}'\hat{\beta}\right)\left(\hat{S}(t|\mathbf{x})\right)^2 var\left(\hat{H}_0(t)\right)$$

Plots

For a specified pattern, the covariate values $\mathbf{x}_c$ are determined and $\mathbf{x}_c$ is computed. There are three plots available for Cox regression.

Survival Plot

For stratum j, $\left(t_i, \hat{S}_0(t_i|\mathbf{x}_c)\right), i = 1, \ldots, k_j$ are plotted where

$$\hat{S}(t_i|\mathbf{x}_c) = \left(\hat{S}_0(t_i)\right)^{\exp\left(\mathbf{x}'_c\hat{\beta}\right)}$$

Hazard Plot

For stratum *j*, $\left(t_i, \hat{H}(t_i|\mathbf{x}_c)\right), i = 1, \ldots, k_j$ are plotted where

$$\hat{H}(t_i|\mathbf{x}_c) = \exp\left(\mathbf{x}'_c\hat{\beta}\right)\hat{H}_0(t_i)$$

LML Plot

The log-minus-log plot is used to see whether the stratification variable should be included as a covariate. For stratum *j*, $\left(t_i, \mathbf{x}'_c\hat{\beta} + \ln \hat{H}_0(t_i)\right), i = 1, \ldots, k_j$ are plotted. If the plot shows parallelism among strata, then the stratum variable should be a covariate.

Blank Handling

All records with missing values for any input or output field are excluded from the estimation of the model.

Scoring

Survival and cumulative hazard estimates are given in “Survival Table ” on p. 101.

Conditional upon survival until time t_0, the probability of survival until time *t* is

$$\hat{S}\left(t + t_0|t_0\right) = \frac{\hat{S}\left(t + t_0\right)}{\hat{S}\left(t_0\right)}$$

Blank Handling

Records with missing values for any input field in the final model cannot be scored, and are assigned a predicted value of $null$.

Additionally, records with “total” survival time (past + future) greater than the record with the longest observed uncensored survival time are also assigned a predicted value of $null$.

References

Breslow, N. E. 1974. Covariance analysis of censored survival data. *Biometrics*, 30, 89–99.

Cain, K. C., and N. T. Lange. 1984. Approximate case influence for the proportional hazards regression model with censored data. *Biometrics*, 40, 493–499.

Cox, D. R. 1972. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society, Series B*, 34, 187–220.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The statistical analysis of failure time data*, 2 ed. New York: John Wiley & Sons, Inc.

Lawless, R. F. 1982. *Statistical models and methods for lifetime data*. New York: John Wiley & Sons, Inc..

Storer, B. E., and J. Crowley. 1985. A diagnostic for Cox regression and general conditional likelihoods. *Journal of the American Statistical Association*, 80, 139–147.

Decision List Algorithms

The objective of decision lists is to find a group of individuals with a distinct behavior pattern; for example, a high probability of buying a product. A decision list model consists of a set of decision rules. A decision rule is an if-then rule, which has two parts: antecedent and consequent. The **antecedent** is a Boolean expression of predictors, and the **consequent** is the predicted value of the target field when the antecedent is true. The simplest construct of a decision rule is a segment based on one predictor; for example, Gender = 'Male' or $10 < Age \leq 20$.

A record is **covered** by a rule if the rule antecedent is true. If a case is covered by one of the rules in a decision list, then it is considered to be covered by the list.

In a decision list, order of rules is significant; if a case is covered by a rule, it will be ignored by subsequent rules.

Algorithm Overview

The decision list algorithm can be summarized as follows:

- Candidate rules are found from the original dataset.
- The best rules are appended to the decision list.
- Records covered by the decision list are removed from the dataset.
- New rules are found based on the reduced dataset.

The process repeats until one or more of the stopping criteria are met.

Terminology of Decision List Algorithm

The following terms are used in describing the decision list algorithm:

Model. A decision list model.

Cycle. In every rule discovery cycle, a set of candidate rules will be found. They will then be added to the model under construction. The resulting models will be inputs to the next cycle.

Attribute. Another name for a variable or field in the dataset.

Source attribute. Another name for predictor field.

Extending the model. Adding decision rules to a decision list or adding segments to a decision rule.

Group. A subset of records in the dataset.

Segment. Another name for group.

Main Calculations

Notation

The following notations are used in describing the decision list algorithm:

X	Data matrix. Columns are fields (attributes), and rows are records (cases).
L	A collection of list models.
L_i	The *i*th list model of *L*.
L_{null}	A list model that contains no rules.
$\hat{P}_{L_i}$	The estimated response probability of list L_i.
N	Total population size
$X_{m,n}$	The value of the *m*th field (column) for the *n*th record (row) of *X*.
X_{L_i}	The subset of records in *X* that are covered by list model L_i.
Y	The target field in *X*.
Y_n	The value of the target field for the *n*th record.
A	Collection of all attributes (fields) of *X*.
A_j	The *j*th attribute of *X*.
R	A collection of rules to extend a preceding rule list.
R_k	The *k*th rule in rule collection *R*.
T	A set of candidate list models.
ResultSet	A collection of decision list models.

Primary Algorithm

The primary algorithm for creating a decision list model is as follows:

1. Initialize the model.

- Let *d* = Search depth, and *w* = Search width.
- If *L* = ∅, add L_{null} to *L*.
- *T* = ∅.

2. Loop over all elements L_i of L.

- Select the records $X_{\overline{L}_i}$ *not* covered by rules of L_i:

$$X_{\overline{L}_i} = X - X_{L_i}$$

- Call the decision rule algorithm to create an alternative rule set *R* on $X_{\overline{L}_i}$. For more information, see the topic "Decision Rule Algorithm" on p. 107.

- Construct a set of new candidate models by appending each rule in *R* to L_i.
- Save extended list(s) to *T*.

3. Select list models from *T*.

- Calculate the estimated response probability $\hat{P}_{L_i}$ of each list model in *T* as

$$\hat{P}_{L_i} = \frac{N(Y_n = 1, X_n \in X_{L_i})}{N(X_n \in X_{L_i})}$$

- Select the *w* lists in *T* with the highest $\hat{P}_{L_i}$ as L^*.

4. Add L^* to ResultSet.
5. If $d = 1$ or $L^* = \varnothing$, return ResultSet and terminate; otherwise, reduce *d* by one and repeat from step 2.

Decision Rule Algorithm

Each rule is extended in decision rule cycles. With decision rules, groups are searched for significantly increased occurrence of the target value. Decision rules will search for groups with a higher or lower probability as required.

Notation

The following notations are used in describing the decision list algorithm:

X	Data matrix. Columns are fields (attributes), and rows are records (cases).
R	A collection of rules to extend a preceding rule list.
R_i	The *i*th rule in rule collection *R*.
R_{all}	A special rule that covers all the cases in *X*.
$\hat{P}_{R_i}$	The estimated response probability of R_i.
N	Total population size.
$X_{m,n}$	The value of the *m*th field (column) for the *n*th record (row) of *X*.
X_{R_i}	The subset of records in *X* that are covered by rule R_i.
Y	The target field in *X*.
Y_n	The value of the target field for the *n*th record.
A	Collection of all attributes (fields) of *X*.
A_j	The *j*th attribute of *X*. If Allow attribute re-use is false, A excludes attributes existing in the preceding rule.
SplitRule(*X*, A_j)	The rule split algorithm for deriving rules about A_j and records in *X*. For more information, see the topic “Decision Rule Split Algorithm” on p. 108.
T	A set of candidate list models.
ResultSet	A collection of decision list models.

Algorithm Steps

The decision rule algorithm proceeds as follows:

1. Initialize the rule set.

- Let d = Search depth, and w = Search width.
- If $R = \varnothing$, add R_{all} to R.
- $T = \varnothing$.

2. Loop over all rules R_i in R.

- Select records X_{R_i} covered by rule R_i.
- Create an empty set S of new segments.
- Loop over attributes A_j in A.
  - Generate new segments based on attribute A_j:

    $\text{SplitRule}(X_{R_i}, A_j)$

  - Add new segments to S.
- Construct a set of new candidate rules by extending R_i with each segment in S.
- Save extended rules to T. If $S = \varnothing$, add R_i to ResultSet.

3. Select rules from T.

- Calculate the estimated response probability $\hat{P}_{R_i}$ for each extended rule in T as

$$\hat{P}_{R_i} = \frac{N(Y_n = 1, X_n \in X_{R_i})}{N(X_n \in X_{R_i})}$$

- Select the w rules with the highest $\hat{P}_{R_i}$ as R^*.

  Add R^* to ResultSet.
- If $d = 1$, return ResultSet and terminate. Otherwise, set $R = R^*$, $T = \varnothing$, reduce d by one, and repeat from step 2.

Decision Rule Split Algorithm

The decision rule split algorithm is used to generate high response segments from a single attribute (field). The records and the attribute from which to generate segments should be given. This algorithm is applicable to all ordinal attributes, and the ordinal attribute should have values that are unambiguously ordered. The segments generated by the algorithm can be used to expand an n-dimensional rule to an $(n + 1)$-dimensional rule. This decision rule split algorithm is sometimes referred to as the sea-level method.

Notation

The following notations are used in describing the decision rule split algorithm:

X	Data matrix. Columns are fields (attributes), and rows are records (cases).
C	A sorted list of attribute values (categories) to split. Values are sorted in ascending order.
C_i	The *i*th category in the list of categories *C*.
$X_{n,c}$	The value of the split field (attribute) for the *n*th record (row) of *X*.
Y	The target field in *X*.
Y_n	The value of the target field for the *n*th record.
N	Total population size.
M	Number of categories in *C*.
P_i	Observed response probability of category C_i.
$S_{L,R}$	A segment of categories, $S_{L,R} = \{C_i \vert C_i \in C, 1 \leq L \leq i \leq R \leq M\}$.
$\left(p^-_{S_{L,R}}, p^+_{S_{L,R}}\right)$	The confidence interval (CI) for the response probability of $S_{L,R}$.
$\max_p(C_i, C_j)$	The category with the higher response probability from $\{C_i, C_j\}$.
$\max_n(C_i, C_j)$	The category with the larger number of records from $\{C_i, C_j\}$.

Algorithm Steps

The decision rule split algorithm proceeds as follows:

1. Compute P_i of each category C_i.

$$P_i = \frac{N(Y_n = 1, X_{n,c} \in C_i)}{N(X_{n,c} \in C_i)}, P_0 = P_{(M+1)} = 0$$

If $N(X_{n,c} \in C_i) = 0$, C_i will be skipped.

2. Find local maxima of P_i to create a segment set.

$$PeakSet = \{C_i \vert C_i \in C, 0 \leq i = I \leq M\}$$

where *I* is a positive integer satisfying the conditions

$$P_I > P_{(I-1)}$$

$$P_I = P_{(I+l)}, 0 \leq l \leq L - I$$

$$P_L > P_{(L+1)}$$

The segment set is the ordered segments based on P_{S_i}

$$SegmentSet \; = \; \left\{ S_{L,R} | C_i \in PeakSet, L = R = i, P_{S_i} \geq P_{S_{i+1}} \right\}$$

3. Select a segment in *SegmentSet*.

- If *SegmentSet* is empty, return ResultSet and terminate.
- Select the segment $S_{L,R}$ with the highest response probability $P_{S_{L,R}}$.
- If $R - L + 1 = M$ or $P_{S_{L,R}} \leq P_{S_{1,M}}$, remove the segment from *SegmentSet* and choose another.

4. Validate the segment.

- If the following conditions are satisfied:
  - The size of the segment exceeds the minimum segment size criterion

    $$Size\left(S_{L,R}\right) > Max(gs_{\text{min}}, d, Max(g \cdot Size(parent)\,)$$

    where

    $$parent \;\in ResultSet, L_{parent} \geq L, R_{parent} \leq R$$

  - Response probability for the segment is significantly higher than that for the overall sample, as indicated by non-overlapping confidence intervals

    $$p^{-}_{S_{L,R}} > p^{+}_{Pop}$$

    For more information, see the topic “Confidence Intervals” on p. 110.

  - Extending the segment would lower the response probability

    $$P_{S_{L-1,R}} < P_{S_{L,R}} \text{ and } P_{S_{L,R+1}} < P_{S_{L,R}}$$

  then add the segment $S_{L,R}$ to ResultSet, and remove any segments $S^{'}_{L,R}$ from ResultSet that have $S_{L,R}$ as parent and for which $Size\left(S^{'}_{L,R}\right) \leq g \cdot Size(S_{L,R})$.

5. Extend the segment.

- Add $C_{adjacent}$ to $S_{L,R}$, where

$$C_{adjacent} = \begin{cases} Max_p(C_{L-1}, C_{R+1}) & \text{if } P_{L-1} \neq P_{R+1} \\ Max_n(C_{L-1}, C_{R+1}) & \text{if } P_{L-1} = P_{R+1} \text{ and } N\left(C_{L-1}\right) \neq N(C_{R+1}) \\ C_{R+1} & \text{otherwise} \end{cases}$$

- Adjust R or L accordingly, i.e. if $C_{adjacent} = C_{L-1}$, set $L = L - 1$; if $C_{adjacent} = C_{R+1}$, set $R = R + 1$.
- Return $S_{L,R}$ to SegmentSet, and repeat from step 3.

Confidence Intervals

The confidence limits (p^{-}, p^{+}) for $\hat{p}$ are calculated as

$$p^{-}=\begin{cases}\frac{x}{x+(n-x+1)F_{2(n-x+1),2x;1-\alpha/2}} & \text{if } x\neq 0\\ 0 & \text{if } x=0\end{cases}$$

$$p^{+}=\begin{cases}\frac{(x+1)F_{2(x+1),2(n-x);1-\alpha/2}}{n-x+(x+1)F_{2(x+1),2(n-x);1-\alpha/2}} & \text{if } x\neq n\\ 1 & \text{if } x=n\end{cases}$$

where *n* is the coverage of the rule or list, *x* is the response frequency of the rule or list, *α* is the desired confidence level, and $F_{a,b;c}$ is the inverse cumulative distribution function for *F* with *a* and *b* degrees of freedom, for percentile $100c$.

Secondary Measures

For each segment, the following measures are reported:

Coverage. The number of records in the segment, $N(S)$.

Frequency. The number of records in the segment for which the response is true, $N(Y_n=1,X_n\in S)$.

Probability. The proportion of records in the segment for which the response is true, $\frac{N(Y_n=1,X_n\in S)}{N(S)}$, or $\frac{\text{Frequency}}{\text{Coverage}}$.

Blank Handling

In decision list models, blank values for input fields can be treated as a separate category that can be used to define segments, or can be excluded from the model, depending on the expert model option. The default is to use blanks as a category for defining segments. Records with blank values for the target field are excluded from model building.

Generated Model/Scoring

The decision list generated model consists of a set of segments. When scoring new data, each record is evaluated for membership in each segment, in order. The first segment in model order that describes the record based on the predictor fields claims the record and determines the predicted value and the probability. Records where the predicted value is not the response value will have a value of $null. Probabilities are calculated as described above.

Blank Handling

In scoring data with a decision list generated model, blanks are considered valid values for defining segments. If the model was built with the expert option Allow missing values in conditions disabled, a record with a missing value for one of the input fields will not match any segment that depends on that field for its definition.

DISCRIMINANT Algorithms

No analysis is done for any subfile group for which the number of non-empty groups is less than two or the number of cases or sum of weights fails to exceed the number of non-empty groups. An analysis may be stopped if no variables are selected during variable selection or the eigenanalysis fails.

Notation

The following notation is used throughout this chapter unless otherwise stated:

Table 14-1
Notation

Notation	Description
g	Number of groups
p	Number of variables
q	Number of variables selected
X_{ijk}	Value of variable *i* for case *k* in group *j*
f_{jk}	Case weights for case *k* in group *j*
m_j	Number of cases in group *j*
n_j	Sum of case weights in group *j*
n	Total sum of weights

Basic Statistics

The procedure calculates the following basic statistics.

Mean

$$\overline{X}_{ij} = \left(\sum_{k=1}^{m_j} f_{jk} X_{ijk} \right) / n_j \qquad \text{(variable } i \text{ in group } j\text{)}$$

$$\overline{X}_{i\bullet} = \left(\sum_{j=1}^{g} \sum_{k=1}^{m_j} f_{jk} X_{ijk} \right) / n \qquad \text{(variable } i\text{)}$$

Variances

$$S_{ij}^2 = \frac{\left(\sum_{k=1}^{m_j} f_{jk} X_{ijk}^2 - n_j \overline{X}_{ij}^2 \right)}{(n_j - 1)} \qquad \text{(variable } i \text{ in group } j\text{)}$$

$$S_{i\bullet}^2 = \frac{\left(\sum_{j=1}^{g} \sum_{k=1}^{m_j} f_{jk} X_{ijk}^2 - n \overline{X}_i^2 \right)}{(n-1)} \qquad \text{(variable } i\text{)}$$

Within-Groups Sums of Squares and Cross-Product Matrix (W)

$$w_{il}=\sum_{j=1}^{g}\sum_{k=1}^{m_j} f_{jk}X_{ijk}X_{ljk}-\sum_{j=1}^{g}\left(\sum_{k=1}^{m_j} f_{jk}X_{ijk}\right)\left(\sum_{k=1}^{m_j} f_{jk}X_{ljk}\right)/n_j \quad i,l=1,\ldots,p$$

Total Sums of Squares and Cross-Product Matrix (T)

$$t_{il}=\sum_{j=1}^{g}\sum_{k=1}^{m_j} f_{jk}X_{ijk}X_{ljk}-\left(\sum_{j=1}^{g}\sum_{k=1}^{m_j} f_{jk}X_{ijk}\right)\left(\sum_{j=1}^{g}\sum_{k=1}^{m_j} f_{jk}X_{ljk}\right)/n$$

Within-Groups Covariance Matrix

$$\mathbf{C}=\frac{\mathbf{W}}{(n-g)} \quad n>g$$

Individual Group Covariance Matrices

$$c_{il}^{(j)}=\frac{\left(\sum_{k=1}^{m_j} f_{jk}X_{ijk}X_{ljk}-\overline{X}_{ij}\overline{X}_{lj}n_j\right)}{(n_j-1)}$$

Within-Groups Correlation Matrix (R)

$$r_{il}=\begin{cases}\frac{w_{il}}{\sqrt{w_{ii}w_{ll}}} & \text{if } w_{ii}w_{ll}>0\\ \text{SYSMIS} & \text{otherwise}\end{cases}$$

Total Covariance Matrix

$$\mathbf{T}^{'}=\frac{\mathbf{T}}{n-1}$$

Univariate F and $\Lambda$for Variable I

$$F_i=\frac{(t_{ii}-w_{ii})(n-g)}{w_{ii}(g-1)}$$

with g–1 and n–g degrees of freedom

$$\Lambda_i=\frac{w_{ii}}{t_{ii}}$$

with 1, g–1 and n–g degrees of freedom

Rules of Variable Selection

Both direct and stepwise variable entry are possible. Multiple inclusion levels may also be specified.

Method = Direct

For direct variable selection, variables are considered for inclusion in the order in which they are passed from the upstream node. A variable is included in the analysis if, when it is included, no variable in the analysis will have a tolerance less than the specified tolerance limit (default = 0.001).

Stepwise Variable Selection

At each step, the following rules control variable selection:

- Eligible variables with higher inclusion levels are entered before eligible variables with lower inclusion levels.
- The order of entry of eligible variables with the same even inclusion level is determined by their order in the upstream node.
- The order of entry of eligible variables with the same odd level of inclusion is determined by their value on the entry criterion. The variable with the "best" value for the criterion statistic is entered first.
- When level-one processing is reached, prior to inclusion of any eligible variables, all already-entered variables which have level one inclusion numbers are examined for removal. A variable is considered eligible for removal if its *F*-to-remove is less than the *F* value for variable removal, or, if probability criteria are used, the significance of its *F*-to-remove exceeds the specified probability level. If more than one variable is eligible for removal, that variable is removed that leaves the "best" value for the criterion statistic for the remaining variables. Variable removal continues until no more variables are eligible for removal. Sequential entry of variables then proceeds as described previously, except that after each step, variables with inclusion numbers of one are also considered for exclusion as described before.
- A variable with a zero inclusion level is never entered, although some statistics for it are printed.

Ineligibility for Inclusion

A variable with an odd inclusion number is considered ineligible for inclusion if:

- The tolerance of any variable in the analysis (including its own) drops below the specified tolerance limit if it is entered, or
- Its *F*-to-enter is less than the *F*-value for a variable to enter value, or
- If probability criteria are used, the significance level associated with its *F*-to-enter exceeds the probability to enter.

A variable with an even inclusion number is ineligible for inclusion if the first condition above is met.

Computations During Variable Selection

During variable selection, the matrix **W** is replaced at each step by a new matrix $\mathbf{W}^*$ using the symmetric sweep operator described by Dempster (1969). If the first q variables have been included in the analysis, **W** may be partitioned as:

$$\begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{21} & \mathbf{W}_{22} \end{bmatrix}$$

where $\mathbf{W}_{11}$ is q×q. At this stage, the matrix $\mathbf{W}^*$ is defined by

$$\mathbf{W}^* = \begin{bmatrix} -\mathbf{W}_{11}^{-1} & \mathbf{W}_{11}^{-1}\mathbf{W}_{12} \\ \mathbf{W}_{21}\mathbf{W}_{11}^{-1} & \mathbf{W}_{22} - \mathbf{W}_{21}\mathbf{W}_{11}^{-1}\mathbf{W}_{12} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{11}^* & \mathbf{W}_{12}^* \\ \mathbf{W}_{21}^* & \mathbf{W}_{22}^* \end{bmatrix}$$

In addition, when stepwise variable selection is used, T is replaced by the matrix T^*, defined similarly.

The following statistics are computed.

Tolerance

$$\text{TOL}_i = \begin{cases} 0 & \text{if } w_{ii} = 0 \\ w_{ii}^*/w_{ii} & \text{if variable } i \text{ is not in the analysis and } w_{ii} \neq 0 \\ -1/(w_{ii}^* w_{ii}) & \text{if variable } i \text{ is in the analysis and } w_{ii} \neq 0. \end{cases}$$

If a variable’s tolerance is less than or equal to the specified tolerance limit, or its inclusion in the analysis would reduce the tolerance of another variable in the equation to or below the limit, the following statistics are not computed for it or any set including it.

F-to-Remove

$$F_i = \frac{(w_{ii}^* - t_{ii}^*)(n-q-g+1)}{t_{ii}^*(g-1)}$$

with degrees of freedom g–1 and n–q–g+1.

F-to-Enter

$$F_i = \frac{(t_{ii}^* - w_{ii}^*)(n-q-g)}{w_{ii}^*(g-1)}$$

with degrees of freedom g–1 and n–q–g.

Wilks’ Lambda for Testing the Equality of Group Means

$$\Lambda = |\mathbf{W}_{11}|/|\mathbf{T}_{11}|$$

with degrees of freedom q, g–1 and n–g.

The Approximate F Test for Lambda (the "overall F"), also known as Rao's R (Tatsuoka, 1971)

$F = \frac{(1-\Lambda^s)(r/s+1-qh/2)}{\Lambda^s qh}$

where

$$s = \begin{cases} \sqrt{\frac{q^2+h^2-5}{q^2h^2-4}} & \text{if } q^2+h^2 \neq 5 \\ 1 & \text{otherwise} \end{cases}$$
$$r = n-1-(q+g)/2$$
$$h = g-1$$

with degrees of freedom qh and $r/s+1-qh/2$. The approximation is exact if q or h is 1 or 2.

Rao's V (Lawley-Hotelling Trace) (Rao, 1952; Morrison, 1976)

$$V = -(n-g)\sum_{i=1}^{q}\sum_{l=1}^{q} w_{il}^{*}(t_{il}-w_{il})$$

When $n-g$ is large, V, under the null hypothesis, is approximately distributed as χ^2 with $q(g-1)$ degrees of freedom. When an additional variable is entered, the change in V, if positive, has approximately a χ^2 distribution with $g-1$ degrees of freedom.

The Squared Mahalanobis Distance (Morrison, 1976) between groups a and b

$$D_{ab}^2 = -(n-g)\sum_{i=1}^{q}\sum_{l=1}^{q} w_{il}^{*}\left(\overline{X}_{ia}-\overline{X}_{ib}\right)\left(\overline{X}_{la}-\overline{X}_{lb}\right)$$

The F Value for Testing the Equality of Means of Groups a and b (Smallest F ratio)

$F_{ab} = \frac{(n-q-g+1)n_a n_b}{q(n-g)(n_a+n_b)} D_{ab}^2$

The Sum of Unexplained Variations (Dixon, 1973)

$$R = \sum_{a=1}^{g-1}\sum_{b=a+1}^{g} 4/\left(4+D_{ab}^2\right)$$

Classification Functions

Once a set of q variables has been selected, the classification functions (also known as Fisher's linear discriminant functions) can be computed using

$$b_{ij} = (n-g)\sum_{l=1}^{q} w_{il}^{*}\overline{X}_{lj} \quad i=1,2,\ldots,q, j=1,2,\ldots,g$$

for the coefficients, and

$$a_j = \log p_j - \tfrac{1}{2}\sum_{i=1}^{q} b_{ij}\overline{X}_{ij} \quad j = 1, 2, \ldots, q$$

for the constant, where p_j is the prior probability of group *j*.

Canonical Discriminant Functions

The canonical discriminant function coefficients are determined by solving the general eigenvalue problem

$$(\mathbf{T} - \mathbf{W})\mathbf{V} = \lambda\mathbf{W}\mathbf{V}$$

where **V** is the unscaled matrix of discriminant function coefficients and **λ** is a diagonal matrix of eigenvalues. The eigensystem is solved as follows:

The Cholesky decomposition

$$\mathbf{W} = \mathbf{L}\mathbf{U}$$

is formed, where $\mathbf{L}$ is a lower triangular matrix, and $\mathbf{U} = \mathbf{L}'$.

The symmetric matrix $\mathbf{L}^{-1}\mathbf{B}\mathbf{U}^{-1}$ is formed and the system

$$\big(\mathbf{L}^{-1}(\mathbf{T} - \mathbf{W})\mathbf{U}^{-1} - \lambda\mathbf{I}\big)(\mathbf{U}\mathbf{V}) = 0$$

is solved using tridiagonalization and the QL method. The result is *m* eigenvalues, where $m = \min\,(q, g-1)$ and corresponding orthonormal eigenvectors, **UV**. The eigenvectors of the original system are obtained as

$$\mathbf{V} = \mathbf{U}^{-1}(\mathbf{U}\mathbf{V})$$

For each of the eigenvalues, which are ordered in descending magnitude, the following statistics are calculated.

Percentage of Between-Groups Variance Accounted for

$$\frac{100\lambda_k}{\sum\limits_{k=1}^{m}\lambda_k}$$

Canonical Correlation

$$\sqrt{\lambda_k/(1+\lambda_k)}$$

Wilks' Lambda

Testing the significance of all the discriminating functions after the first *k*:

$$\Lambda_k = \prod_{i=k+1}^{m} 1/(1+\lambda_i) \quad k = 0, 1, \ldots, m-1$$

The significance level is based on

$$\chi^2 = -(n - (q+g)/2 - 1)\ln\Lambda_k$$

which is distributed as a χ^2 with $(q{-}k)(g{-}k{-}1)$ degrees of freedom.

The Standardized Canonical Discriminant Coefficient Matrix D

The standard canonical discriminant coefficient matrix **D** is computed as

$$\mathbf{D} = \mathbf{S}_{11}\mathbf{V}$$

where

$\mathbf{S}$=diag$\left(\sqrt{w_{11}}, \sqrt{w_{22}}, \ldots, \sqrt{w_{pp}}\right)$

$\mathbf{S}_{11}$= partition containing the first *q* rows and columns of **S**

V is a matrix of eigenvectors such that $\mathbf{V}'\mathbf{W}_{11}\mathbf{V}$=*I*

The Correlations Between the Canonical Discriminant Functions and the Discriminating Variables

The correlations between the canonical discriminant functions and the discriminating variables are given by

$$\mathbf{R} = \mathbf{S}_{11}^{-1}\mathbf{W}_{11}\mathbf{V}$$

If some variables were not selected for inclusion in the analysis (*q*<*p*), the eigenvectors are implicitly extended with zeroes to include the nonselected variables in the correlation matrix. Variables for which $W_{ii} = 0$ are excluded from **S** and **W** for this calculation; *p* then represents the number of variables with non-zero within-groups variance.

The Unstandardized Coefficients

The unstandardized coefficients are calculated from the standardized ones using

$$\mathbf{B} = \sqrt{(n-g)}\mathbf{S}_{11}^{-1}\mathbf{D}$$

The associated constants are:

$$a_k = -\sum_{i=1}^{q} b_{ik}\overline{X}_{i\bullet}$$

The group centroids are the canonical discriminant functions evaluated at the group means:

$$\overline{f}_{kj} = a_k + \sum_{i=1}^{q} b_{ik}\overline{X}_{ij}$$

Tests For Equality Of Variance

Box's *M* is used to test for equality of the group covariance matrices.

$$M = (n-g)\log\left|\mathbf{C}'\right| - \sum_{j=1}^{g}(n_j-1)\log\left|\mathbf{C}^{(j)}\right|$$

where

$\mathbf{C}'$ = pooled within-groups covariance matrix excluding groups with singular covariance matrices

$\mathbf{C}^{(j)}$ = covariance matrix for group *j*.

Determinants of $\mathbf{C}'$ and $\mathbf{C}^{(j)}$ are obtained from the Cholesky decomposition. If any diagonal element of the decomposition is less than 10^{-11}, the matrix is considered singular and excluded from the analysis.

$$\log\left|\mathbf{C}^{(j)}\right| = 2\sum_{i=1}^{p}\log l_{ii} - p\log(n_j-1)$$

where l_{ii} is the *i*th diagonal entry of $\mathbf{L}$ such that $(n_j-1)\mathbf{C}^{(j)} = \mathbf{L}'\mathbf{L}$. Similarly,

$$\log\left|\mathbf{C}'\right| = 2\sum_{i=1}^{p}\log l_{ii} - p\log\left(n' - g\right)$$

where

$$\left(n' - g\right)\mathbf{C}' = \mathbf{L}'\mathbf{L}$$

n'= sum of weights of cases in all groups with nonsingular covariance matrices

The significance level is obtained from the *F* distribution with t_1 and t_2 degrees of freedom using (Cooley and Lohnes, 1971):

$$F = \begin{cases} M/b & \text{if } e_2 > e_1^2 \\ \frac{t_2 M}{t_1(b-M)} & \text{if } e_2 < e_1^2 \end{cases}$$

where

$$e_1 = \left(\sum_{j=1}^{g} \frac{1}{n_j - 1} - \frac{1}{n-g} \right) \frac{2p^2+3p-1}{6(g-1)(p+1)}$$

$$e_2 = \left(\sum_{j=1}^{g} \frac{1}{(n_j - 1)^2} - \frac{1}{(n-g)^2} \right) \frac{(p-1)(p+2)}{6(g-1)}$$

$$t_1 = (g-1)p(p+1)/2$$

$$t_2 = (t_1 + 2)/\left|e_2 - e_1^2\right|$$

$$b = \begin{cases} \frac{t_1}{1-e_1-t_1/t_2} & \text{if } e_2 > e_1^2 \\ \frac{t_2}{1-e_1-2/t_2} & \text{if } e_2 < e_1^2 \end{cases}$$

If $e_1^2 - e_2$ is zero, or much smaller than e_2, t_2 cannot be computed or cannot be computed accurately. If

$$e_2 = e_2 + 0.0001\left(e_2 - e_1^2\right)$$

the program uses Bartlett's χ^2 statistic rather than the *F* statistic:

$$\chi^2 = M(1 - e_1)$$

with t_1 degrees of freedom.

For testing the group covariance matrix of the canonical discriminant functions, the procedure is similar. The covariance matrices $\mathbf{C}'$ and $\mathbf{C}^{(j)}$ are replaced by $\mathbf{D}_j$ and $\mathbf{D}'$, where

$$\mathbf{D}_j = \mathbf{B}'\mathbf{C}^{(j)}\mathbf{B}$$

is the group covariance matrix of the discriminant functions.

The pooled covariance matrix in this case is an identity, so that

$$\mathbf{D}' = (n-g)\mathbf{I}_m - \sum_j (n_j - 1)\mathbf{D}_j$$

where the summation is only over groups with singular $\mathbf{D}_j$.

Blank Handling

All records with missing values for any input or output field are excluded from the estimation of the model.

Generated model/scoring

The basic procedure for classifying a case is as follows:

- If **X** is the 1×*q* vector of discriminating variables for the case, the 1×*m* vector of canonical discriminant function values is

$$\mathbf{f}=\mathbf{XB}+\mathbf{a}$$

- A chi-square distance from each centroid is computed

  $$\chi_j^2=\left(\mathbf{f}-\overline{\mathbf{f}}_j\right)\mathbf{D}_j^{-1}\left(\mathbf{f}-\overline{\mathbf{f}}_j\right)'$$

  where $\mathbf{D}_j$ is the covariance matrix of canonical discriminant functions for group *j* and $\overline{\mathbf{f}}_j$ is the group centroid vector. If the case is a member of group *j*, χ_j^2 has a χ^2 distribution with *m* degrees of freedom. *P*(**X**|**G**), labeled as *P*(**D**>*d*|**G**=*g*) in the output, is the significance level of such a χ_j^2 .

- The classification, or posterior probability, is

  $$P\left(\mathbf{G}_j|\mathbf{X}\right)=\frac{P_j|\mathbf{D}_j|^{-1/2}e^{-\chi_j^2/2}}{\sum\limits_{j=1}^{g}P_j|\mathbf{D}_j|^{-1/2}e^{-\chi_j^2/2}}$$

  where p_j is the prior probability for group *j*. A case is classified into the group for which $P\left(\mathbf{G}_j|\mathbf{X}\right)$ is highest.

The actual calculation of $P\left(\mathbf{G}_j|\mathbf{X}\right)$ is

$$g_j=\log P_j-\tfrac{1}{2}\left(\log|\mathbf{D}_j|+\chi_j^2\right)$$

$$P\left(\mathbf{G}_j|\mathbf{X}\right)=\begin{cases}\dfrac{\exp\left(g_j-\max_j g_j\right)}{\sum\limits_{j=1}^{g}\exp\left(g_j-\max\limits_{j} g_j\right)} & \text{if } g_j-\max_j g_j>-46\\ 0 & \text{otherwise}\end{cases}$$

If individual group covariances are not used in classification, the pooled within-groups covariance matrix of the discriminant functions (an identity matrix) is substituted for $\mathbf{D}_j$ in the above calculation, resulting in considerable simplification.

If any $\mathbf{D}_j$ is singular, a pseudo-inverse of the form

$$\begin{bmatrix}\mathbf{D}_{j11}^{-1} & 0\\ 0 & 0\end{bmatrix}$$

replaces $\mathbf{D}_j^{-1}$ and $|\mathbf{D}_{j11}|$ replaces $|\mathbf{D}_j|$. $\mathbf{D}_{j11}$ is a submatrix of $\mathbf{D}_j$ whose rows and columns correspond to functions not dependent on preceding functions. That is, function 1 will be excluded only if the rank of $\mathbf{D}_j=0$, function 2 will be excluded only if it is dependent on function 1, and so on. This choice of the pseudo-inverse is not optimal for the numerical stability of $\mathbf{D}_{j11}^{-1}$, but maximizes the discrimination power of the remaining functions.

Cross-Validation (Leave-one-out classification)

The following notation is used in this section:

Table 14-2
Notation

Notation	Description
$\underset{\sim}{X}_{jk}$	$(X_{1jk},\ldots,X_{qjk})^T$

Notation	Description
$\underset{\sim}{M}_j$	Sample mean of *j*th group $\underset{\sim}{M}_j = \frac{1}{n_j}\sum_{k=1}^{m_j} f_{jk}\underset{\sim}{X}_{jk}$
$\underset{\sim}{M}_{jk}$	Sample mean of *j*th group excluding point $\underset{\sim}{X}_{jk}$ $\underset{\sim}{M}_{jk} = \frac{1}{n_j - f_{jk}}\sum_{\substack{l=1\\ l \neq k}}^{m_j} f_{jl}\underset{\sim}{X}_{jl}$
Σ	Polled sample covariance matrix
Σ_j	Sample covariance matrix of *j*th group
Σ_{jk}	Polled sample covariance matrix without point $\underset{\sim}{X}_{jk}$
Σ_{jk}^{-1}	$= \frac{n-g-f_{jk}}{n-g}\left(\Sigma^{-1} + \frac{n_j\Sigma_j^{-1}\left(\underset{\sim}{X}_{jk}-\underset{\sim}{M}_j\right)\left(\underset{\sim}{X}_{jk}-\underset{\sim}{M}_j\right)^T\Sigma_j^{-1}}{(n_j-f_{jk})(n_j-g)-n_j\left(\underset{\sim}{X}_{jk}-\underset{\sim}{M}_j\right)^T\Sigma_j^{-1}\left(\underset{\sim}{X}_{jk}-\underset{\sim}{M}_j\right)}\right)$
$d_0^2\left(\underset{\sim}{a}, \underset{\sim}{b}\right)$	$\left(\underset{\sim}{a} - \underset{\sim}{b}\right)^T \Sigma_{jk}^{-1}\left(\underset{\sim}{a} - \underset{\sim}{b}\right)^T$

Cross-validation applies only to linear discriminant analysis (not quadratic). During cross-validation, all cases in the dataset are looped over. Each case, say $\underset{\sim}{X}_{jk}$, is extracted once and treated as test data. The remaining cases are treated as a new dataset.

Here we compute $d_0^2\left(\underset{\sim}{X}_{jk}, \underset{\sim}{M}_{jk}\right)$ and $d_0^2\left(\underset{\sim}{X}_{jk}, \underset{\sim}{M}_i\right)(i = 1, ..., g.i \neq j)$. If there is an $i(i \neq j)$ that satisfies ($\log\left(P_i\right) - d_0^2\left(\underset{\sim}{X}_{jk}, \underset{\sim}{M}_i\right)/2 > \log\left(P_j\right) - d_0^2\left(\underset{\sim}{X}_{jk}, \underset{\sim}{M}_{jk}\right)/2$), then the extracted point $\underset{\sim}{X}_{jk}$ is misclassified. The estimate of prediction error rate is the ratio of the sum of misclassified case weights and the sum of all case weights.

To reduce computation time, the linear discriminant method is used instead of the canonical discriminant method. The theoretical solution is exactly the same for both methods.

Blank Handling (discriminant analysis algorithms scoring)

Records with missing values for any input field in the final model cannot be scored, and are assigned a predicted value of \$null\$.

References

Anderson, T. W. 1958. *Introduction to multivariate statistical analysis*. New York: John Wiley & Sons, Inc..

Cooley, W. W., and P. R. Lohnes. 1971. *Multivariate data analysis*. New York: John Wiley & Sons, Inc..

Dempster, A. P. 1969. *Elements of Continuous Multivariate Analysis*. Reading, MA: Addison-Wesley.

Dixon, W. J. 1973. *BMD Biomedical computer programs*. Los Angeles: University of California Press.

Tatsuoka, M. M. 1971. *Multivariate analysis*. New York: John Wiley & Sons, Inc. .

Ensembles Algorithms

Ensembles are used to enhance model accuracy (boosting), enhance model stability (bagging), build models for very large datasets (pass, stream, merge), and generally combine scores from different models.

- For more information, see the topic “Very large datasets (pass, stream, merge) algorithms” on p. 130.
- For more information, see the topic “Bagging and Boosting Algorithms” on p. 125.
- For more information, see the topic “Ensembling model scores algorithms” on p. 136.

Bagging and Boosting Algorithms

Bootstrap aggregating (Bagging) and boosting are algorithms used to improve model stability and accuracy. Bagging works well for unstable base models and can reduce variance in predictions. Boosting can be used with any type of model and can reduce variance and bias in predictions.

Notation

The following notation is used for bagging and boosting unless otherwise stated:

K	The number of distinct records in the training set.
X_k	Predictor values for the *k*th record.
y_k	Target value for the *k*th record.
f_k	Frequency weight for the *k*th record.
w_k	Analysis weight for the *k*th record.
N	The total number of records; $N = \Sigma_{k=1}^{K} f_k$.
M	The number of base models to build; for bagging, this is the number of bootstrap samples.
$T^m\left(\cdot\right)$	The model built on the *m*th bootstrap sample.
f_k^m	Simulated frequency weight for the *k*th record of the *m*th bootstrap sample.
w_k^m	Updated analysis weight for the *k*th record of the *m*th bootstrap sample.
$\hat{y}_k^m = T^m\left(X_k\right)$	Predicted target value of the *k*th record by the *m*th model.
$P_{l_i}^m\left(X_k\right)$	For a categorical target, the probability that the kth record belongs to category l_i, *i*=1, ..., *C*, in model *m*.
$II\left(\pi\right)$	For any condition π, $II\left(\pi\right)$ is 1 if π holds and 0 otherwise.

Bootstrap Aggregation

Bootstrap aggregation (bagging) produces replicates of the training dataset by sampling with replacement from the original dataset. This creates bootstrap samples of equal size to the original dataset. The algorithm is performed iteratively over *k*=1,..,*K* and *m*=1,...,*M* to generate frequency weights:

$$f_{mk}^{*}=\begin{cases} rv.binom\left(N,\frac{f_k}{N}\right) & k=1 \\ rv.binom\left(N-\Sigma_{i=1}^{k-1}f_{mi}^{*},\dfrac{f_k}{N-\Sigma_{i=1}^{k-1}f_i}\right) & \text{otherwise} \end{cases}$$

Then a model is built on each replicate. Together these models form an ensemble model. The ensemble model scores new records using one of the following methods; the available methods depend upon the measurement level of the target.

Scoring a Continuous Target

- Mean

  $\hat{y}_k=\frac{1}{M}\sum_{m=1}^{M}\hat{y}_k^m$

- Median

  Sort $\hat{y}_k^m$ and relabel them $\hat{y}_{(1)}\leq ...\leq\hat{y}_{(M)}$

  $$\hat{y}_k=\left\{\begin{array}{cc} \hat{y}_{\left(\frac{M+1}{2}\right)} & \text{if } M \text{ is odd} \\ \frac{1}{2}\left(\hat{y}_{\left(\frac{M}{2}\right)}+\hat{y}_{\left(\frac{M}{2}+1\right)}\right) & \text{if } M \text{ is even} \end{array}\right\}$$

Scoring a Categorical Target

- Voting

  $\hat{y}_k=arg\max_{l_i\in\Omega}\frac{1}{|M_{l_i}|}\sum_{m\in M_{l_i}}P_{l_i}^m\left(X_k\right)$

  $\hat{p}_{\hat{y}_k}=\frac{1}{|M_{\hat{y}_k}|}\sum_{m\in M_{\hat{y}_k}}P_{\hat{y}_k}^m\left(X_k\right)$

  where $\Omega=\{arg\max_{l_i}|M_{l_i}|\}$

- Highest probability

  $\hat{y}_k=arg\max_{l_i}\left(\max_m\left(P_{l_i}^m\left(X_k\right)\right)\right)$

  $\hat{p}_{\hat{y}_k}=\max_m\left(P_{\hat{y}_k}^m\left(X_k\right)\right)$

- Highest mean probability

  $\hat{y}_k=arg\max_{l_i}\frac{1}{M}\sum_{m=1}^{M}P_{l_i}^m\left(X_k\right)$

  $\hat{p}_{\hat{y}_k}=\frac{1}{M}\sum_{m=1}^{M}P_{\hat{y}_k}^m\left(X_k\right)$

Bagging Model Measures

Accuracy

Accuracy is computed for the naive model, reference (simple) model, ensemble model (associated with each ensemble method), and base models.

For categorical targets, the classification accuracy is

$$\frac{1}{N}\sum_{k=1}^{K} f_k II\left(y_k == \hat{y}_k\right)$$

For continuous targets, it is

$$R^2 = 1 - \frac{\Sigma_{k=1}^{K} f_k {(y_k - \hat{y}_k)}^2}{\Sigma_{k=1}^{K} f_k {(y_k - \overline{y})}^2}$$

where $\overline{y} = \frac{1}{N}\Sigma_{k=1}^{K} f_k y_k$

Note that R^2 can never be greater than one, but can be less than zero.

For the naïve model, $\hat{y}_k$ is the modal category for categorical targets and the mean for continuous targets.

Diversity

Diversity is a range measure between 0 and 1 in the larger-is-more-diverse form. It shows how much predictions vary across base models.

For categorical targets, diversity is

$$\frac{1}{N \cdot M^2}\sum_{k=1}^{K} f_k L\left(y_k\right)\left[M - L\left(y_k\right)\right]$$

where $L\left(y_k\right) = \sum_{m=1}^{M} II\left(y_k = \hat{y}_k^m\right)$.

For continuous targets, diversity is

$$\mathrm{D} = \frac{\sum_{k=1}^{K} f_k \left[\frac{1}{M\left(M-1\right)}\sum_{m=1}^{M}\sum_{n=1,n\neq m}^{M}\left(y_k - \hat{y}_k^m\right)\left(\hat{y}_k^n - y_k\right)\right]}{\Sigma_{k=1}^{K} f_k {(y_k - \overline{y}_k)}^2}$$

Adaptive Boosting

Adaptive boosting (AdaBoost) is an algorithm used to boost models with continuous targets (Freund and Schapire 1996, Drucker 1997).

1. Initialize values.

   Set $w_k = \begin{cases} \frac{w_k}{\Sigma_{i=1}^{K} w_i f_i} & \text{if analysis weights specified} \\ 1/N & \text{otherwise} \end{cases}$

   Set *m*=1, $w_k^m = w_k$, and $f_k^m = f_k$. Note that analysis weights are initialized even if the method used to build base models does not support analysis weights.

2. Build base model *m*, $T^m(\cdot)$, using the training set and score the training set.

   Set the model weight for base model *m*, $\omega^m = \log\left(\frac{1 - \sum_{k=1}^{K} L_k w_k^m f_k}{\sum_{k=1}^{K} L_k w_k^m f_k}\right)$

   where $L_k = \frac{abs(\hat{y}_k^m - y_k)}{\max_k\left(abs\left(\hat{y}_k^m - y_k\right)\right)}$.

3. Set weights for the next base model .

   $w_k^{m+1} = \frac{a_k^{m+1}}{\Sigma_{i=1}^{K} a_i^{m+1} f_i}$

   where $a_k^{m+1} = w_k^m \left(\frac{\sum_{k=1}^{K} L_k w_k^m f_k}{1 - \sum_{k=1}^{K} L_k w_k^m f_k}\right)^{1-L_k}$. Note that analysis weights are always updated. If the method used to build base models does not support analysis weights, the frequency weights are updated for the next base model as follows:

   $$f_k^{m+1} = \begin{cases} rv.binom\left(N, w_k^{m+1} f_k\right) & k = 1 \\ rv.binom\left(N - \Sigma_{i=1}^{k-1} f_k^{m+1}, \frac{w_k^{m+1} f_k}{1 - \Sigma_{i=1}^{k-1} w_k^{m+1} f_i}\right) & \text{otherwise} \end{cases}$$

   If *m*<*M*, set *m*=*m*+1 and go to step 2. Otherwise, the ensemble model is complete.

Note: base models where $\sum_{k=1}^{K} L_k w_k^m f_k \geq 0.5$ or $\max_k\left(abs(\hat{y}_k^m - y_k)\right)$ are removed from the ensemble.

Scoring

AdaBoost uses the weighted median method to score the ensemble model.

Sort $\hat{y}_k^m$ and relabel them $\hat{y}_{(1)} \leq ... \leq \hat{y}_{(M)}$, retaining the association of the model weights, ω^m, and relabeling them $\omega_{(1)}, ..., \omega_{(M)}$

The ensemble predicted value is then $\hat{y}_k = \hat{y}_{(i)}$, where i is the value such that $\sum_{m=1}^{i-1} \omega^m < \frac{1}{2}\sum_{m=1}^{M} \omega^m \leq \sum_{m=1}^{i} \omega^m$

Stagewise Additive Modeling using Multiclass Exponential loss

Stagewise Additive Modeling using a Multiclass Exponential loss function (SAMME) is an algorithm that extends the original AdaBoost algorithm to categorical targets.

1. Initialize values.

   Set $w_k = \begin{cases} \frac{w_k}{\Sigma_{i=1}^{K} w_i f_i} & \text{if analysis weights specified} \\ 1/N & \text{otherwise} \end{cases}$

   Set *m*=1, $w_k^m = w_k$, and $f_k^m = f_k$. Note that analysis weights are initialized even if the method used to build base models does not support analysis weights.

2. Build base model *m*, $T^m(\cdot)$, using the training set and score the training set.

   Set the model weight for base model *m*, $\omega^m = \log \frac{1-err_m}{err^m} + \log(C-1)$

   where $err^m = \sum_{k=1}^{K} w_k^m f_k II\left(y_k \neq \hat{y}_k^m\right)$.

3. Set weights for the next base model.

   $w_k^{m+1} = \frac{a_k^{m+1}}{\Sigma_{i=1}^{K} a_i^{m+1} f_i}$

   where $a_k^{m+1} = w_k^m \exp\left(\omega^m II\left(y_k \neq \hat{y}_k^m\right)\right)$. Note that analysis weights are always updated. If the method used to build base models does not support analysis weights, the frequency weights are updated for the next base model as follows:

   $$f_k^{m+1} = \begin{cases} rv.binom\left(N, w_k^{m+1} f_k\right) & k = 1 \\ rv.binom\left(N - \Sigma_{i=1}^{k-1} f_k^{m+1}, \frac{w_k^{m+1} f_k}{1 - \Sigma_{i=1}^{k-1} w_k^{m+1} f_i}\right) & \text{otherwise} \end{cases}$$

   If *m*<*M*, set *m*=*m*+1 and go to step 2. Otherwise, the ensemble model is complete.

   Note: base models where $err_m = 0$ or $\omega^m <= 0$ are removed from the ensemble.

Scoring

SAMME uses the weighted majority vote method to score the ensemble model.

The predicted value of the *k*th record for the *m*th base model is $\hat{y}_k^m = arg\max_{l_i} P_{l_i}^m\left(X_k\right)$.

The ensemble predicted value is then $\hat{y}_k = arg\max_{l_i} \sum_{m=1}^{M} \omega^m II\left(\hat{y}_k^m == l_i\right)$. Ties are resolved at random.

The ensemble predicted probability is $\hat{p}_{\hat{y}_k} = \sum_{m \in M_{\hat{y}_k}} \frac{\omega^m}{\sum_{i \in M_{\hat{y}_k}} \omega^i} P^m_{\hat{y}_k}(X_k)$.

Boosting Model Measures

Accuracy

Accuracy is computed for the naive model, reference (simple) model, ensemble model (associated with each ensemble method), and base models.

For categorical targets, the classification accuracy is

$$\frac{1}{N}\sum_{k=1}^{K} f_k II\left(y_k == \hat{y}_k\right)$$

For continuous targets, it is

$$R^2 = 1 - \frac{\Sigma_{k=1}^{K} f_k (y_k - \hat{y}_k)^2}{\Sigma_{k=1}^{K} f_k (y_k - \overline{y})^2}$$

where $\overline{y} = \frac{1}{N}\Sigma_{k=1}^{K} f_k y_k$

Note that R^2 can never be greater than one, but can be less than zero.

For the naïve model, $\hat{y}_k$ is the modal category for categorical targets and the mean for continuous targets.

References

Drucker, H. 1997. Improving regressor using boosting techniques. In: *Proceedings of the 14th International Conferences on Machine Learning ,* D. H. Fisher,Jr., ed. San Mateo, CA: Morgan Kaufmann, 107–115.

Freund, Y., and R. E. Schapire. 1995. A decision theoretic generalization of on-line learning and an application to boosting. In: *Computational Learning Theory: 7 Second European Conference, EuroCOLT '95,* , 23–37.

Very large datasets (pass, stream, merge) algorithms

We implement the PSM features PASS, STREAM, and MERGE through ensemble modeling. PASS builds models on very large data sets with only one data pass; STREAM updates the existing model with new cases without the need to store or recall the old training data; MERGE builds models in a distributed environment and merges the built models into one model.

In an ensemble model, the training set will be divided into subsets called blocks, and a model will be built on each block. Because the blocks may be dispatched to different threads (here one process contains one thread) and even different machines, models in different processes can be built at the same time. As new data blocks arrive, the algorithm simply repeats this procedure. Therefore it can easily handle the data stream and perform incremental learning for ensemble modeling.

Pass

The PASS operation includes following steps:

1. Split the data into training blocks, a testing set and a holdout set. Note that the frequency weight, if specified, is ignored when splitting the training set into blocks (to prevent blocks from being entirely represented by a single case) but is accounted for when creating the testing and holdout sets.

2. Build base models on training blocks and build a reference model on the testing set. A single model is built on the testing set and each training block.

3. Evaluate each base model by computing the accuracy based on the testing set. Select a subset of base models as ensemble elements according to accuracy.

4. Evaluate the ensemble model and the reference model by computing the accuracy based on the holdout set. If the ensemble model's performance is not better than the reference model's performance on the holdout set, we use the reference model to score the new cases.

Computing Model Accuracy

The accuracy of a base model is assessed on the testing set. For each vector of predictors x_i and the corresponding label c_i observed in the testing set *T*, let $\hat{c}\left(x_i\right)$ be the label predicted by the given model. Then the testing error is estimated as:

Categorical target.$E = \frac{1}{\sum\limits_{i=1}^{|T|} f_i} \sum\limits_{i=1}^{|T|} \left(f_i \cdot I\left(c_i \neq \hat{c}\left(x_i\right)\right)\right)$

Continuous target.$E = \frac{1}{\sum\limits_{i=1}^{|T|} f_i} \sum\limits_{i=1}^{|T|} \left(f_i \cdot \left|y_i - \hat{y}_i\right|\right)$

Where $I\left(c_i \neq \hat{c}\left(x_i\right)\right)$ is 1 if $c_i \neq \hat{c}\left(x_i\right)$ and 0 otherwise.

The accuracy for the given model is computed by *A*=1−*E*. The accuracy for the whole ensemble model and the reference model is assessed on the holdout set.

Stream

When new cases arrive and the user wants to update the existing ensemble model with these cases, the algorithm will:

1. Start a PASS operation to build an ensemble model on the new data, then
2. MERGE the newly created ensemble model and the existing ensemble model.

Merge

The MERGE operation has the following steps:

1. Merge the holdout sets into a single holdout set and, if necessary, reduce this set to a reasonable size.
2. Merge the testing sets into a single testing set and, if necessary, reduce this set to a reasonable size.
3. Build a merged reference model on the merged testing set.
4. Evaluate every base model by computing the accuracy based on the merged testing set. Select a subset of base models as elements of the merged ensemble model according to accuracy.
5. Evaluate the merged ensemble model and the merged reference model by computing the accuracy based on the merged holdout set.

Adaptive Predictor Selection

There are two methods, depending upon whether the method used to build base models has an internal predictor selection algorithm.

Method has predictor selection algorithm

The first base model is built with all predictors available to the method's predictor selection algorithm. Base model *j* (*j* > 1) makes the *i*th predictor available with probability

$$p_i = \max\left(\frac{n'_i + C}{n_i + C}, \beta\right)$$

where n'_i is the number of times the *i*th predictor was selected by the method's predictor selection algorithm in the previous *j*−1 base models, n_i is the number of times the *i*th predictor was made available to the method's predictor selection algorithm in the previous *j*−1 base models, *C* is a constant to smooth the value of p_i, and β is a lower limit on p_i.

Method does not have predictor selection algorithm

Each base model makes the *i*th predictor available with probability

$$p_i = \begin{cases} (1-\rho_i)^2 & \text{if } \rho_i < 0.05 \\ \beta & \text{otherwise} \end{cases}$$

where ρ_i is the *p*-value of a test for the *i*th predictor, as defined below.

- For a categorical target and categorical predictor, ρ is a chi-square test of $G^2 = 2\sum_{i=1}^{I}\sum_{j=1}^{J} G_{ij}^2$ where $G_{ij}^2 = \begin{cases} N_{ij}\ln\left(N_{ij}/\hat{N}_{ij}\right) & N_{ij} > 0 \\ 0 & \text{else} \end{cases}$ and with degrees of freedom $(I-1)(J-1)$. N_{ij} is the number of cases with *X*=*i* and *Y*=*j*, $N_{i\cdot} = \sum_{j=1}^{J} N_{ij}$, $N_{\cdot j} = \sum_{i=1}^{I} N_{ij}$, and $\hat{N}_{ij} = N_{i\cdot}N_{\cdot j}/N$.
- For a categorical target and continuous predictor, ρ is an F test of $F = \frac{\sum_{j=1}^{J} N_j\left(\overline{x}_j - \overline{\overline{x}}\right)^2/(J-1)}{\sum_{j=1}^{J}(N_j-1)s_j/(N-J)}$ with degrees of freedom $J-1, N-J$. N_j is the number of cases with *Y*=*j*, $\overline{x}_j$ and s_j^2 are the sample mean and sample variance of *X* given *Y*=*j*, and $\overline{\overline{x}} = \sum_{j=1}^{J} N_j\overline{x}_j/N$
- For a continuous target and categorical predictor, ρ is an F test of $F = \frac{\sum_{i=1}^{I} N_i\left(\overline{y}_i - \overline{\overline{y}}\right)^2/(I-1)}{\sum_{i=1}^{I}(N_i-1)s(y)_i/(N-I)}$ with degrees of freedom $I-1, N-I$. N_i is the number of cases with *X*=*i*, $\overline{y}_i$ and $s(y)_i^2$ are the sample mean and sample variance of *Y* given *X*=*i*, and $\overline{\overline{y}} = \Sigma_{i=1}^{I} N_i\overline{y}_i/N$.
- For a continuous target and continuous predictor, ρ is a two-sided *t* test of $t = r\sqrt{\frac{N-2}{1-r^2}}$ where $r = \frac{\Sigma_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})/(N-1)}{\sqrt{s(x)^2 s(y)^2}}$ and with degrees of freedom $N-2$. $s(x)^2$ is the sample variance of *X* and $s(y)^2$ is the sample variance of *Y*.

Automatic Category Balancing

When a target category occurs relatively infrequently, many models do a poor job of predicting members of that rarely occurring category, even if the overall prediction rate of the model is fairly good. Automatic category balancing should improves the model's accuracy when predicting infrequently occurring values.

As records arrive, they are added to a training block until it is full. Then the proportion of records in each category is computed: $C_i = \frac{w_i}{w}$, where w_i is the weighted number of records taking category *i* and *w* is the total weighted number of records.

- ▶ If there is any category such that $C_i < \alpha/(10 \cdot |C|)$, where $|C|$ is the number of target categories and α= 0.3, then randomly remove each record from the training block with probability

  $$Min\left\{(1 - Min(C)/C_i), \left(1 - \frac{\alpha}{|C|}\right)\right\}$$

  This operation will tend to remove records from frequently-occurring categories. Add new records to the training block until it is full again, and repeat this step until the condition is not satisfied.

- ▶ If there is any category such that $C_i < \alpha/|C|$, then recompute the frequency weight for record *k* as $f_k = f_k \max\left(10, \alpha\max(C)/C_{i(k)}\right)$, where $i\,(k)$ is the category of the *k*th record. This operation gives greater weight to infrequently occurring categories.

Model Measures

The following notation applies.

N	Total number of records
M	Total number of base models
f_k	The frequency weight of record *k*
y_k	The observed target value of record *k*
$\hat{y}_k$	The predicted target value of record *k* by the ensemble model
$\hat{y}_k^m$	The predicted target value of record *k* by base model *m*

Accuracy

Accuracy is computed for the naive model, reference (simple) model, ensemble model (associated with each ensemble method), and base models.

For categorical targets, the classification accuracy is

$$\frac{1}{N}\sum_{k=1}^{K} f_k II\left(y_k == \hat{y}_k\right)$$

where

$$II\left(y_k = \hat{y}_k\right) = \begin{cases} 1, \text{if}\left(y_k = \hat{y}_k\right) \\ 0, \text{otherwise} \end{cases}$$

For continuous targets, it is

$$R^2 = 1 - \frac{\Sigma_{k=1}^{K} f_k \left(y_k - \hat{y}_k\right)^2}{\Sigma_{k=1}^{K} f_k \left(y_k - \overline{y}\right)^2}$$

where $\overline{y} = \frac{1}{N}\Sigma_{k=1}^{K} f_k y_k$

Note that R^2 can never be greater than one, but can be less than zero.

For the naïve model, $\hat{y}_k$ is the modal category for categorical targets and the mean for continuous targets.

Diversity

Diversity is a range measure between 0 and 1 in the larger-is-more-diverse form. It shows how much predictions vary across base models.

For categorical targets, diversity is

$$\frac{1}{N \cdot M^2} \sum_{k=1}^{K} f_k L\left(y_k\right)\left[M - L\left(y_k\right)\right]$$

where $L\left(y_k\right) = \sum_{m=1}^{M} II\left(y_k = \hat{y}_k^m\right)$ and $II\left(y_k = \hat{y}_k^m\right)$ is defined as above.

Diversity is not available for continuous targets.

Scoring

There are several strategies for scoring using the ensemble models.

Continuous Target

Mean.$\hat{y}_{i,PSM} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_{i,m}$

Median.$\hat{y}_{i,PSM} = Median_1^M\left(\hat{y}_{i,m}\right)$

where $\hat{y}_{i,PSM}$ is the final predicted value of case *i*, and $\hat{y}_{i,m}$ is the *m*th base model's predicted value of case *i*.

Categorical Target

Voting. Assume that $d_{m,k}$ represents the label output of the *m*th base model for a given vector of predictor values. $d_{m,k} = 1$ if the label assigned by the *m*th base model is the *k*th target category and 0 otherwise. There are total of *M* base models and *K* target categories. The majority vote method selects the *j*th category if it is assigned by the plurality of base models. It satisfies the following equation:

$$\sum_{m=1}^{M} d_{m,j} = \max_{k=1}^{K}\left(\sum_{m=1}^{M} d_{m,k}\right)$$

Let E_m be the testing error estimated for the *m*th base model. Weights for the weighted majority vote are then computed according to the following expression:

$$w_m = \max\left(\log\frac{1-E_m}{E_m}, 0\right) / \sum_{i=1}^{M} \max\left(\log\frac{1-E_i}{E_i}, 0\right)$$

Probability voting. Assume that $p_{m,k}$ is the posterior probability estimated for the *k*th target category by the *m*th base model for a given vector of predictor values. The following rules combine the probabilities computed by the base models. The *j*th category is selected such that it satisfies the corresponding equation.

- Highest probability. $\max_{\mathrm{m}=1}^{\mathrm{M}}\left(p_{m,j}\right) = \max_{k=1}^{K}(\max_{\mathrm{m}=1}^{\mathrm{M}}\left(p_{m,k}\right)\Big)$
- Highest mean probability. $\frac{1}{M}\sum_{m=1}^{M} p_{m,j} = \max_{k=1}^{K}\left(\frac{1}{M}\sum_{m=1}^{M} p_{m,k}\right)$

Ties are resolved at random.

Softmax smoothing. The softmax function can be used for smoothing the probabilities:

$$p_i^S = \frac{Exp\left(p_i\right)}{\sum_{i=1}^{K} Exp\left(p_i\right)}$$

where p_i is the rule-based confidence for category *i* and p_i^S is the smoothed value.

Ensembling model scores algorithms

Ensembling scores from individual models can give more accurate predictions. By combining scores from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy. Models combined in this manner typically perform at least as well as the best of the individual models and often better.

Note that while the options for general ensembling of scores are similar to those for boosting, bagging, and very large datasets, the specific options for combining scoring are slightly different.

Notation

The following notation applies.

N	Total number of records
M	Total number of base models
y_i	The observed target value of record *i*
$\hat{y}_i$	The predicted target value of record *i* by the ensemble model
$\hat{y}_i^m$	The predicted target value of record *i* by base model *m*

Scoring

There are several strategies for scoring using the ensemble models.

Continuous Target

Mean.$\hat{y}_{i,M} = \frac{1}{M}\sum_{m=1}^{M}\hat{y}_{i,m}$

where $\hat{y}_{i,M}$ is the final predicted value of case *i*, and $\hat{y}_{i,m}$ is the *m*th base model's predicted value of case *i*.

Standard error.$\hat{SE}_{i,M} = \frac{1}{\sqrt{M}}\sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(\hat{y}_{i,m} - \hat{y}_{i,M}\right)^2}$

Categorical Target

Voting. Assume that $d_{m,k}$ represents the label output of the *m*th base model for a given vector of predictor values. $d_{m,k} = 1$ if the label assigned by the *m*th base model is the *k*th target category and 0 otherwise. There are total of *M* base models and *K* target categories. The majority vote method selects the *j*th category if it is assigned by the plurality of base models. It satisfies the following equation:

$$\sum_{m=1}^{M} d_{m,j} = \max_{k=1}^{K} \left(\sum_{m=1}^{M} d_{m,k} \right)$$

Confidence-weighted (probability) voting. Assume that $p_{m,k}$ is the posterior probability estimated for the *k*th target category by the *m*th base model for a given vector of predictor values. The following rules combine the probabilities computed by the base models. The *j*th category is selected such that it satisfies the corresponding equation.

$$\sum_{m=1}^{M} p_{m,j} d_{m,j} = \max_{k=1}^{K} p_{m,k} \left(\sum_{m=1}^{M} d_{m,k} \right)$$

Highest confidence (probability) wins.

$$\max_{\mathrm{m}=1}^{\mathrm{M}} \left(p_{m,j} \right) = \max_{k=1}^{K} (\max_{\mathrm{m}=1}^{\mathrm{M}} \left(p_{m,k} \right) \big)$$

Raw propensity-weighted voting. This is equivalent to confidence-weighted voting for a flag target, where the weights for *true* are the propensities and the weights for *false* are 1−propensity.

Adjusted propensity-weighted voting. This is similar to raw propensity-weighted voting for a flag target, where the weights for *true* are the adjusted propensities and the weights for *false* are 1−adjusted propensity.

Average raw propensity. The raw propensities scores are averaged across the base models. If the average is > 0.5, then the record is scored as *true*.

Average adjusted propensity. The adjusted propensities scores are averaged across the base models. If the average is > 0.5, then the record is scored as *true*.

Factor Analysis/PCA Algorithms

Overview

The Factor/PCA node performs principal components analysis and six types of factor analysis.

Primary Calculations

Factor Extraction

Principal Components Analysis

The matrix of factor loadings based on factor *m* is

$$\Lambda_m = \Omega_m \Gamma_m^{\frac{1}{2}}$$

where

$$\Omega_m = (\omega_1, \omega_2, ..., \omega_m)$$

$$\Gamma_m = diag\left(|\gamma_1|, |\gamma_2|, ..., |\gamma_m|\right)$$

The communality of variable *i* is given by

$$h_i = \sum_{j=1}^{m} |\gamma_j| \omega_{ij}^2$$

Analyzing a Correlation Matrix

$\gamma_1 \geq \gamma_2 \geq ... \geq \gamma_m$ are the eigenvalues and ω_i are the corresponding eigenvectors of $\mathbf{R}$, where $\mathbf{R}$ is the correlation matrix.

Analyzing a Covariance Matrix

$\gamma_1 \geq \gamma_2 \geq ... \geq \gamma_m$ are the eigenvalues and ω_i are the corresponding eigenvectors of $\mathbf{\Sigma}$, where $\mathbf{\Sigma} = (\sigma_{ij})_{n \times n}$ is the covariance matrix.

The rescaled loadings matrix is $\Lambda_{mR} = [diag(\Sigma)]^{-\frac{1}{2}} \Lambda_m$.

The rescaled communality of variable *i* is $h_{iR} = \sigma_{ii}^{-1} h_i$.

Principal Axis Factoring

Analyzing a Correlation Matrix

An iterative solution for communalities and factor loadings is sought. At iteration *i*, the communalities from the preceding iteration are placed on the diagonal of $\mathbf{R}$, and the resulting $\mathbf{R}$ is denoted by $\mathbf{R}_i$. The eigenanalysis is performed on $\mathbf{R}_i$, and the new communality of variable *j* is estimated by

$$h_{j(i)} = \sum_{j=1}^{m} \left|\gamma_{k(i)}\right| \omega_{jk(i)}^{2}$$

The factor loadings are obtained by

$$\Lambda_{m(i)} = \Omega_{m(i)} \Gamma_{m(i)}^{\frac{1}{2}}$$

Iterations continue until the maximum number (default 25) is reached or until the maximum change in the communality estimates is less than the convergence criterion (default 0.001).

Analyzing a Covariance Matrix

This analysis is the same as analyzing a correlation matrix, except Σ is used instead of the correlation matrix $\mathbf{R}$. Convergence is dependent on the maximum change of rescaled communality estimates.

At iteration *i*, the rescaled loadings matrix is $\Lambda_{m(i)R} = \left[diag(\Sigma)\right]^{-\frac{1}{2}} \Lambda_{m(i)}$. The rescaled communality of variable *i* is $h_{j(i)R} = \sigma_{ii}^{-1} h_{j(i)}$.

Maximum Likelihood

The maximum likelihood solutions of Λ and ψ^2 are obtained by minimizing

$$F = tr\left[\left(\Lambda\Lambda' + \psi^2\right)^{-1}\mathbf{R}\right] - \log\left|\left(\Lambda\Lambda' + \psi^2\right)^{-1}\mathbf{R}\right| - p$$

with respect to Λ and ψ, where *p* is the number of variables, Λ is the factor loading matrix, and ψ^2 is the diagonal matrix of unique variances.

The minimization of *F* is performed by way of a two-step algorithm. First, the conditional minimum of *F* for a given *y* is found. This gives the function $f(\psi)$, which is minimized numerically using the Newton-Raphson procedure. Let $\mathbf{x}^{(s)}$ be the column vector containing the logarithm of the diagonal elements of *y* at the *s*th iteration. Then

$$\mathbf{x}^{(s+1)} = \mathbf{x}^{(s)} - \mathbf{d}^{(s)}$$

where $\mathbf{d}^{(s)}$ is the solution to the system of linear equations

$$\mathbf{H}^{(s)}\mathbf{d}^{(s)} = \mathbf{h}^{(s)}$$

and where

$$\mathbf{H}^{(s)} = \frac{\partial^2 f(\psi)}{\partial x_i \partial x_j}$$

and $\mathbf{h}^{(s)}$ is the column vector containing $\frac{\partial f(\psi)}{\partial x_i}$. The starting point $\mathbf{x}^{(1)}$ is

$$\mathbf{x}_i^1 = \begin{cases} \log\left[\left(1 - \frac{m}{2p}\right)/r^{ii}\right] & \text{for ML and GLS} \\ \left[\left(1 - \frac{m}{2p}\right)/r^{ii}\right]^{\frac{1}{2}} & \text{for ULS} \end{cases}$$

where *m* is the number of factors and r^{ii} is the *i*th diagonal element of $\mathbf{R}^{-1}$.

The values of $f(\psi)$, $\frac{\partial f}{\partial x_i}$, and $\frac{\partial^2 f}{\partial x_i \partial x_j}$ can be expressed in terms of the eigenvalues $\gamma_1 \le \gamma_2 \le ... \le \gamma_p$ and corresponding eigenvectors ω_1, ω_2,...,ω_p of matrix $\psi\mathbf{R}^{-1}\psi$. That is,

$$f(\psi) = \sum_{k=m+1}^{p} \left(\log\gamma_k + \gamma_k^{-1} - 1\right)$$

$$\frac{\partial f}{\partial x_i} = \sum_{k=m+1}^{p} \left(1 - \gamma_k^{-1}\right)\omega_{ik}^2$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = -\delta_{ij}\frac{\partial f}{\partial x_i} + \sum_{k=m+1}^{p} \omega_{ik}\omega_{jk}\left(\sum_{n=1}^{m} \frac{\gamma_k + \gamma_n - 2}{\gamma_k - \gamma_n}\omega_{in}\omega_{jn} + \delta_{ij}\right)$$

where

$$\delta_{ij} = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \ne j \end{cases}$$

The approximate second-order derivatives

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \cong \left(\sum_{k=m+1}^{p} \omega_{ik}\omega_{jk}\right)^2$$

are used in the initial step and when the matrix of the exact second-order derivatives is not positive definite or when all elements of the vector $\mathbf{d}$ are greater than 0.1. If $\frac{\partial^2 f}{\partial x_i^2} < 0.05$ (Heywood variables), the diagonal element is replaced by 1 and the rest of the elements of that column and row are set to 0. If the value of $f(\psi)$ is not decreased by step $\mathbf{d}$, the step is halved and halved again until the value of $f(\psi)$ decreases or 25 halvings fail to produce a decrease. (In this case, the computations are terminated.) Stepping continues until the largest absolute value of the elements of $\mathbf{d}$ is less than the criterion value (default 0.001) or until the maximum number of iterations (default 25) is reached. Using the converged value of ψ (denoted by $\hat{\psi}$), the eigenanalysis is performed on the matrix $\hat{\psi}\mathbf{R}^{-1}\hat{\psi}$. The factor loadings are computed as

$$\hat{\Lambda}_m = \hat{\psi}\Omega_m\left(\Gamma_m^{-1} - \mathbf{I}_m\right)^{\frac{1}{2}}$$

where

$$\Gamma_m = diag\left(\gamma_1, \gamma_2, ..., \gamma_m\right)$$

$$\Omega_m = \left(\omega_1, \omega_2, ..., \omega_m\right)$$

Unweighted and Generalized Least Squares

The same basic algorithm is used in ULS and GLS as in maximum likelihood, except that

$$f(\psi) = \begin{cases} \sum_{k=m+1}^{p} \frac{\gamma_k^2}{2} & \text{for ULS} \\ \sum_{k=m+1}^{p} \frac{(\gamma_k - 1)^2}{2} & \text{for GLS} \end{cases}$$

for the ULS method, the eigenanalysis is performed on the matrix $\mathbf{R} - \psi^2$, where $\gamma_1 \geq \gamma_2 \geq ... \geq \gamma_p$ are the eigenvalues. In terms of the derivatives, for ULS,

$$\frac{\partial f}{\partial x_i} = 2x_i \sum_{k=m+1}^{p} \gamma_k \omega_{ik}^2$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = 4\left[x_i x_j \sum_{k=m+1}^{p} \omega_{ik}\omega_{jk} \sum_{n=1}^{m} \frac{\gamma_k + \gamma_n}{\gamma_k - \gamma_n} \omega_{ik}\omega_{jk} + \delta_{ij} \sum_{k=m+1}^{p} \left(x_i^2 - \frac{\gamma_k}{2}\right)\omega_{ik}^2\right]$$

and

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \cong 4x_i x_j \left(\sum_{k=m+1}^{p} \omega_{ik}\omega_{jk}\right)^2$$

For GLS,

$$\frac{\partial f}{\partial x_i} = \sum_{k=m+1}^{p} (\gamma_k^2 - \gamma_k)\omega_{ik}^2$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \delta_{ij} \frac{\partial f}{\partial x_i} + \sum_{k=m+1}^{p} \gamma_k \omega_{ik}\omega_{jk} \left(\sum_{n=1}^{m} \gamma_n \frac{\gamma_k + \gamma_n - 2}{\gamma_k - \gamma_n} \omega_{in}\omega_{jn} + r^{ii} \exp\left(\frac{x_i + x_j}{2}\right)\right)$$

and

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \cong \left(\sum_{k=m+1}^{p} \omega_{ik}\omega_{jk}\right)^2$$

Also, the factor loadings of the ULS method are obtained by

$$\hat{\Lambda}_m = \Omega_m \Gamma_m^{\frac{1}{2}}$$

The chi-square statistic for *m* factors for the ML and GLS methods is given by

$$\chi_m^2 = \left(W - 1 - \frac{2p+5}{6} - \frac{2m}{3}\right) f\left(\hat{\psi}\right)$$

with $\left((p-m)^2 - p - m\right)/2$ degrees of freedom.

Alpha Factoring

Alpha factoring involves an iterative procedure, where at each iteration *i*:

The eigenvalues ($\gamma_{(i)}$) and eigenvectors ($\omega_{(i)}$) of

$$\mathbf{H}_{(i-1)}^{\frac{1}{2}}(\mathbf{R}-\mathbf{I})\mathbf{H}_{(i-1)}^{\frac{1}{2}} + \mathbf{I}$$

are computed.

The new communalities are

$$h_{k(i)} \left(\sum_{j=1}^{m} \left|\gamma_{j(i)}\right| \omega_{kj(i)}^2 \right) h_{k(i-1)}$$

The initial values of the communalities, $\mathbf{H}_0$, are

$$h_{io} = \begin{cases} 1 - \frac{1}{r^{ii}} & |\mathbf{R}| \geq 10^{-8} \text{and all} 0 \leq h_{io} \leq 1 \\ \max_j \left|r_{ij}\right| & \text{otherwise} \end{cases}$$

where r^{ii} is the *i*th diagonal entry of $\mathbf{R}^{-1}$.

If $|R| \geq 10^{-8}$ and all r^{ii} are equal to one, the procedure is terminated. If for some *i*, $\max_j |r_{ij}| > 1$, the procedure is terminated.

Iteration stops if any of the following are true:

$$\max_k \left|h_{k(i)} - h_{k(i-1)}\right| < \epsilon$$

$$i = MAX$$

$$h_{k(i)} = 0 \text{ for any} k$$

The communalities are the values when iteration stops, unless the last termination criterion is true, in which case the procedure terminates. The factor pattern matrix is

$$F_m = H_{(f)}^{\frac{1}{2}} \Omega_{m(f)} \Gamma_{m(f)}^{\frac{1}{2}}$$

where *f* is the final iteration.

Image Factoring

Analyzing a Correlation Matrix

Eigenvalues and eigenvectors of $\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1}$ are found.

$$S^2 = diag\left(\frac{1}{r^{11}}, ..., \frac{1}{r^{nn}}\right)$$

where r^{11} is the *i*th diagonal element of $\mathbf{R}^{-1}$

The factor pattern matrix is

$$\mathbf{F}_m = \mathbf{S}\Omega_m(\Lambda_m - \mathbf{I}_m)\Lambda_m^{-\frac{1}{2}}$$

where Λ_m and Ω_m correspond to the *m* eigenvalues greater than 1 (and the associated eigenvectors). If $m = 0$, the procedure is terminated.

The communalities are

$$h_i = \sum_{j=1}^{m} \frac{(\gamma_j - 1)^2 \omega_{ij}^2}{(\gamma_j r^{ii})}$$

The image covariance matrix is

$$\mathbf{R} + \mathbf{S}^2\mathbf{R}^{-1}\mathbf{S}^2 - 2\mathbf{S}^2$$

The anti-image covariance matrix is

$$\mathbf{S}^2\mathbf{R}^{-1}\mathbf{S}^2$$

Analyzing a Covariance Matrix

When analyzing a covariance matrix, the covariance matrix Σ is used instead of the correlation matrix $\mathbf{R}$. The calculation is similar to the correlation matrix case.

The rescaled factor pattern matrix is

$$\mathbf{F}_{mR} = [diag(\Sigma)]^{\frac{1}{2}}\mathbf{F}_m$$

and the rescaled communality of variable *i* is $h_{iR} = \sigma_{ii}^{-1} h_i$.

Factor Rotation

Orthogonal Rotations

Rotations are done cyclically on pairs of factors until the maximum number of iterations is reached or the convergence criterion is met. The algorithm is the same for all orthogonal rotations, differing only in computations of the tangent values of the rotation angles.

The factor pattern matrix is normalized by the square root of communalities:

$$\Lambda_m^* = \mathbf{H}^{\frac{1}{2}}\Lambda_m$$

where

$\Lambda_m = (\underline{\lambda}_1, ..., \underline{\lambda}_m)$ is the factor pattern matrix

$\mathbf{H} = diag\,(h_1, ..., h_n)$

The tranformation matrix $\mathbf{T}$ is initialized to $\mathbf{I}_m$.

At each iteration *i*:

- The convergence criterion is

$$SV_{(i)} = \sum_{j=1}^{m}\left(n\sum_{k=1}^{n}\lambda_{kj(i)}^{*4} - \left(\sum_{k=1}^{n}\lambda_{kj(i)}^{*2}\right)^2\right)/n^2$$

where the initial value of $\Lambda_{m(1)}^*$ is the original factor pattern matrix. For subsequent iterations, the initial value is the final value of $\Lambda_{m(i-1)}^*$ when all factor pairs have been rotated.

For all pairs of factors (λ_j,λ_k) where $k > j$, the following are computed:

- The angle of rotation is

$$P = \frac{1}{4}\tan^{-1}\left(\frac{X}{Y}\right)$$

where

$$X = \begin{cases} D - \frac{2AB}{n} & \text{Varimax} \\ D - \frac{mAB}{n} & \text{Equamax} \\ D & \text{Quartimax} \end{cases}$$

$$Y = \begin{cases} C - \left(\frac{A^2-B^2}{n}\right) & \text{Varimax} \\ C - \frac{m(A^2-B^2)}{2n} & \text{Equamax} \\ C & \text{Quartimax} \end{cases}$$

$$u_{p(i)} = f_{pj(i)}^{*2} - f_{pk(i)}^{*2} \quad v_{p(i)} = 2f_{pj(i)}^{*}f_{pk(i)}^{*} \quad p = 1, ..., n$$

$$A = \sum_{p=1}^{n} u_{p(i)} \qquad B = \sum_{p=1}^{n} v_{p(i)}$$
$$C = \sum_{p=1}^{n} \left[u_{p(i)}^2 - v_{p(i)}^2\right] \quad D = \sum_{p=1}^{n} 2u_{p(i)}v_{p(i)}$$

If $|\sin(P)| \leq 10^{-15}$, no rotation is done on the pair of factors.

- The new rotated factors are

$$\left(\underline{\tilde{\lambda}}_{j(i)}, \underline{\tilde{\lambda}}_{k(i)}\right) = \left(\underline{\lambda}_{j(i)}^*, \underline{\lambda}_{k(i)}^*\right) \begin{vmatrix} \cos(P) & -\sin(P) \\ \sin(P) & \cos(P) \end{vmatrix}$$

  where $\underline{\lambda}_{j(i)}^*$ are the last values for factor *j* calculated in this iteration.

- The accrued rotation transformation matrix is

$$\left(\tilde{t}_j, \tilde{t}_k\right) = \left(t_j, t_k\right) \begin{vmatrix} \cos(P) & -\sin(P) \\ \sin(P) & \cos(P) \end{vmatrix}$$

  where t_j and t_k are the last calculated values of the *j*th and *k*th columns of $\mathbf{T}$.

- Iteration is terminated when

$$\left|SV_{(i)} - SV_{(i-1)}\right| \leq 10^{-5}$$

  or the maximum number of iterations is reached.

Final rotated factor pattern matrix

$$\hat{\Lambda}_m = H^{\frac{1}{2}} \Lambda_{m(f)}^*$$

where $\Lambda_{m(f)}^*$ is the value of the last iteration.

Reflect factors with negative sums. If

$$\sum_{i=1}^{n} \tilde{\lambda}_{ij(f)} < 0$$

then

$$\underline{\tilde{\lambda}}_j = -\underline{\tilde{\lambda}}_{j(f)}$$

Rearrange the rotated factors such that

$$\sum_{j=1}^{n} \tilde{\lambda}_{j1}^2 \geq \ldots \geq \sum_{j=1}^{n} \tilde{\lambda}_{jm}^2$$

The communalities are

$$h_j = \sum_{i=1}^{m} \tilde{\lambda}_{ji}^2$$

Direct Oblimin Rotation

The direct oblimin method (Jennrich and Sampson, 1966) is used for oblique rotation. The user can choose the parameter δ. The default value is $\delta = 0$.

The factor pattern matrix is normalized by the square root of the communalities

$$\Omega_m^* = H^{\frac{1}{2}} \Lambda_m$$

where

$$h_j = \sum_{k=1}^{m} \lambda_{jk}^2$$

If no Kaiser is specified, this normalization is not done.

Initializations

The factor correlation matrix $\mathbf{C}$ is initialized to $\mathbf{I}_m$. The following are also computed:

$$s_k = \begin{cases} 1 & \text{if Kaiser} \\ h_k & \text{if no Kaiser} \end{cases} \quad k = 1, ..., n$$

$$u_i = \sum_{j=1}^{n} \lambda_{ji}^{*2} \; i = 1, ..., m$$

$$v_i = \sum_{j=1}^{n} \lambda_{ji}^{*4}$$

$$x_i = v_i - \left(\frac{\delta}{n}\right) u_i^2$$

$$D = \sum_{i=1}^{m} u_i$$

$$G = \sum_{i=1}^{m} x_i$$

$$H = \sum_{k=1}^{n} s_i^2 - \left(\frac{\delta}{n}\right) D^2$$

$$FO = H - G$$

At each iteration, all possible factor pairs are rotated. For a pair of factors $\underline{\lambda}_p^*$ and $\underline{\lambda}_p^*$ $(p \neq q)$, the following are computed:

$$D_{pq} = D - u_p - u_q$$

$$G_{pq} = G - x_p - x_q$$

$$s_{pq,i} = s_i - \lambda_{ip}^{*2} - \lambda_{iq}^{*2}$$

$$y_{pq} = \sum_{i=1}^{n} \lambda_{ip}^* \lambda_{iq}^*$$

$$z_{pq} = \sum_{i=1}^{n} \lambda_{ip}^{*2} \lambda_{iq}^{*2}$$

$$T = \sum_{i=1}^{n} s_{pq,i} \lambda_{ip}^{*2} - \left(\frac{\delta}{n}\right) u_p D_{pq}$$

$$Z = \sum_{i=1}^{n} s_{pq,i} \lambda_{ip}^* \lambda_{iq}^* - \left(\frac{\delta}{n}\right) y_{pq} D_{pq}$$

$$P = \sum_{i=1}^{n} \lambda_{ip}^{*3} \lambda_{iq}^* - \left(\frac{\delta}{n}\right) u_p y_{pq}$$

$$R = z_{pq} - \left(\frac{\delta}{n}\right) u_p u_q$$

$$P' = \frac{3}{2}\left(c_{pq} - \frac{P}{x_p}\right)$$

$$Q' = \frac{1}{2}(x_p - 4c_{pq}P + R + 2T)/x_p$$

$$R' = \frac{1}{2}(c_{pq}(T + R) - P - Z)/x_p$$

A root a of the equation $b^3 + P'b^2 + Q'b + R = 0$ is computed, as well as

$$A = 1 + 2c_{pq}a + a^2$$

$$t_1 = |A|^{\frac{1}{2}}$$

$$t_2 = \frac{a}{t_1}$$

The rotated pair of factors is

$$\left(\tilde{\underline{\lambda}}_p^*, \tilde{\underline{\lambda}}_q^*\right) = \left(\underline{\lambda}_p^*, \underline{\lambda}_q^*\right) \begin{vmatrix} t_1 & -a \\ 0 & 1 \end{vmatrix}$$

These replace the previous factor values.

New values are computed for

$$\tilde{u}_p = |A| u_p$$

$$\tilde{x}_p = A^2 x_p$$

$$\tilde{v}_q = \sum_{i=1}^{n} \tilde{\lambda}_{iq}^{*4}$$

$$\tilde{u}_q = \sum_{i=1}^{n} \tilde{\lambda}_{iq}^{*2}$$

$$\tilde{x}_q = \tilde{v}_q - \left(\frac{\delta}{n}\right) \tilde{u}_q^2$$

$$\tilde{S}_k = S_{pq,k} + \tilde{\lambda}_{kp}^{*2} + \tilde{\lambda}_{kq}^{*2}$$

$$\tilde{D} = D_{pq} + \tilde{u}_p + \tilde{u}_q$$

$$\tilde{G} = G_{pq} + \tilde{x}_p + \tilde{x}_q$$

All values designated with a tilde (~) replace the original values and are used in subsequent calculations.

The new factor correlations with factor *p* are

$$\tilde{c}_{ip} = t_1^{-1} c_{ip} + t_2 c_{iq} \ (i \neq p)$$

$$\tilde{c}_{pi} = \tilde{c}_{ip}$$

$$\tilde{c}_{pp} = 1$$

After all factor pairs have been rotated, iteration is terminated if:

MAX iterations have been done, *or*

$$\left| F1_{(i)} - F1_{(i-1)} \right| < (FO)(EPS)$$

where

$$F1_{(i)} = \tilde{H} - \tilde{G}$$

$$\tilde{H} = \sum_{i=1}^{n} \tilde{s}_k^2 - \left(\frac{\delta}{n}\right)\tilde{D}^2$$

$$F1_{(0)} = FO$$

Otherwise, the factor pairs are rotated again.

The final rotated factor pattern matrix is

$$\tilde{\lambda}_m = \mathbf{H}^{\frac{1}{2}}\tilde{\lambda}_m^*$$

where $\tilde{\lambda}_m$ is the value in the final iteration.

The factor structure matrix is

$$\mathbf{S} = \tilde{\Lambda}_m\tilde{\mathbf{C}}_m$$

where $\tilde{\mathbf{C}}_m$ is the factor correlation matrix in the final iteration.

Promax Rotation

The promax rotation is a computationally fast rotation (Hendrickson and White, 1964). The speed is achieved by first rotating to an orthogonal varimax solution and then relaxing the orthogonality of the factors to better fit the simple structure.

Varimax rotation is used to get an orthogonal rotated matrix $\Lambda_R = \{\lambda_{ij}\}$.

The matrix $\mathbf{P} = (p_{ij})_{p\times m}$ is calculated, where

$$p_{ij} = \left|\frac{\lambda_{ij}}{\left(\sum_{j=1}^{m}\lambda_{ij}^2\right)^{\frac{1}{2}}}\right|^{k+1} \frac{\left(\sum_{j=1}^{m}\lambda_{ij}^2\right)^{\frac{1}{2}}}{\lambda_{ij}}$$

Here, k is the power of promax rotation $(k > 1)$.

The matrix $\mathbf{L}$ is calculated.

$$\mathbf{L} = \left(\Lambda'_R\Lambda_R\right)^{-1}\Lambda'_R\mathbf{P}$$

The matrix $\mathbf{L}$ is normalized by column to a transformation matrix

$$\mathbf{Q} = \mathbf{LD}$$

where $\mathbf{D} = (diag(\mathbf{L}'\mathbf{L}))^{\frac{1}{2}}$ is the diagonal matrix that normalizes the columns of $\mathbf{L}$.

At this stage, the rotated factors are

$$f_{promax\_temp} = \mathbf{Q}^{-1}f_{varimax}$$

Because $var(f_{promax\_temp}) = (\mathbf{Q}'\mathbf{Q})^{-1}$, and the diagonal elements do not equal 1, we must modify the rotated factor to

$$f_{promax} = \mathbf{C} f_{promax\_temp}$$

where $\mathbf{C} = \left\{ diag\left((\mathbf{Q}'\mathbf{Q})^{-1} \right) \right\}^{-\frac{1}{2}}$

The rotated factor pattern is

$$\Lambda_{promax} = \Lambda_{varimax} \mathbf{Q} \mathbf{C}^{-1}$$

The correlation matrix of the factors is

$$\mathbf{R}_{ff} = \mathbf{C} \big(\mathbf{Q}'\mathbf{Q} \big)^{-1} \mathbf{C}'$$

The factor structure matrix is

$$\Lambda_S = \Lambda_{promax} \mathbf{R}_{ff}$$

Factor Score Coefficients

IBM® SPSS® Modeler uses the regression method of computing factor score coefficients (Harman, 1976).

$$\mathbf{W} = \begin{cases} \Lambda_m \Gamma_m^{-1} & \text{PCA without rotation} \\ \Lambda_m (\Lambda'_m \Lambda_m)^{-1} & \text{PCA with rotation} \\ \mathbf{R}^{-1} \mathbf{S}_m & \text{otherwise} \end{cases}$$

where $\mathbf{S}_m$ is the factor structure matrix. For orthogonal rotations $\mathbf{S}_m = \Lambda_m$.

For principal components analysis without rotation, if any $|\gamma_i| \leq 10^{-8}$, factor score coefficients are not computed. For principal components with rotation, if the determinant of $\Lambda'_m \Lambda_m$ is less than 10^{-8}, the coefficients are not computed. Otherwise, if $\mathbf{R}$ is singular, factor score coefficients are not computed.

Blank Handling

By default, a case that has a missing value for any input or output field is deleted from the computation of the correlation matrix on which all consequent computations are based. If the Only use complete records option is deselected, each correlation in the correlation matrix $\mathbf{R}$ is computed based on records with complete data for the two fields associated with the correlation, regardless of missing values on other fields. For some datasets, this approach can lead to a nonpositive definite $\mathbf{R}$ matrix, so that the model cannot be estimated.

Secondary Calculations

Field Statistics and Other Calculations

The statistics shown in the advanced output for the regression equation node are calculated in the same manner as in the FACTOR procedure in IBM® SPSS® Statistics. For more details, see the SPSS Statistics Factor algorithm document, available at *http://www.ibm.com/support*.

Generated Model/Scoring

Factor Scores

Factor scores are assigned to scored records by applying the factor score coefficients to the input field value for the record,

$$fs_k = \sum_{i=1}^{n} w_{ki} f_i$$

where fs_k is the factor score for the *k*th factor, w_{ki} is the factor score coefficient for the *i*th input field (from the **W** matrix) and the *k*th factor, and f_i is the value of the *i*th input field for the record being scored.For more information, see the topic "Factor Score Coefficients" on p. 151.

Blank Handling

Records with missing values for any input field in the final model cannot be scored and are assigned factor/component score values of $null$.

Feature Selection Algorithm

Introduction

Data mining problems often involve hundreds, or even thousands, of variables. As a result, the majority of time and effort spent in the model-building process involves examining which variables to include in the model. Fitting a neural network or a decision tree to a set of variables this large may require more time than is practical.

Feature selection allows the variable set to be reduced in size, creating a more manageable set of attributes for modeling. Adding feature selection to the analytical process has several benefits:

- Simplifies and narrows the scope of the features that is essential in building a predictive model.
- Minimizes the computational time and memory requirements for building a predictive model because focus can be directed to the subset of predictors that is most essential.
- Leads to more accurate and/or more parsimonious models.
- Reduces the time for generating scores because the predictive model is based upon only a subset of predictors.

Primary Calculations

Feature selection consists of three steps:

- **Screening.** Removes unimportant and problematic predictors and cases.
- **Ranking.** Sorts remaining predictors and assigns ranks.
- **Selecting.** Identifies the important subset of features to use in subsequent models.

The algorithm described here is limited to the supervised learning situation in which a set of predictor variables is used to predict a target variable. Any variables in the analysis can be either categorical or continuous. Common target variables include whether or not a customer churns, whether or not a person will buy, and whether or not a disease is present.

The terms **features**, **variables**, and **attributes** are often used interchangeably. Within this document, we use variables and predictors when discussing input to the feature selection algorithm, with features referring to the predictors that actually get selected by the algorithm for use in a subsequent modeling process.

Screening

This step removes variables and cases that do not provide useful information for prediction and issues warnings about variables that may not be useful.

The following variables are removed:

- Variables that have all missing values.
- Variables that have all constant values.
- Variables that represent case ID.

The following cases are removed:

- Cases that have missing target values.
- Cases that have missing values in all its predictors.

The following variables are removed based on user settings:

- Variables that have more than m_1% missing values.
- Categorical variables that have a single category counting for more than m_2% cases.
- Continuous variables that have standard deviation < m_3%.
- Continuous variables that have a coefficient of variation |CV| < m_4%. CV = standard deviation / mean.
- Categorical variables that have a number of categories greater than m_5% of the cases.

Values m_1, m_2, m_3, m_4, and m_5 are user-controlled parameters.

Ranking Predictors

This step considers one predictor at a time to see how well each predictor alone predicts the target variable. The predictors are ranked according to a user-specified criterion. Available criteria depend on the measurement levels of the target and predictor.

The **importance** value of each variable is calculated as $(1 - p)$, where p is the p value of the appropriate statistical test of association between the candidate predictor and the target variable, as described below.

Categorical Target

This section describes ranking of predictors for a categorical target under the following scenarios:

- All predictors categorical
- All predictors continuous
- Some predictors categorical, some continuous

All Categorical Predictors

The following notation applies:

Table 17-1
Notation

Notation	Description
X	The predictor under consideration with I categories.
Y	Target variable with J categories.
N	Total number of cases.
N_{ij}	The number of cases with $X = i$ and $Y = j$.

Notation	**Description**
$N_{i\cdot}$	The number of cases with $X = i$. $N_{i\cdot} = \sum_{j=1}^{J} N_{ij}$
$N_{\cdot j}$	The number of cases with $Y = j$. $N_{\cdot j} = \sum_{i=1}^{I} N_{ij}$

The above notations are based on nonmissing pairs of (*X*, *Y*). Hence *J*, *N*, and $N{\cdot}j$ may be different for different predictors.

P Value Based on Pearson's Chi-square

Pearson's chi-square is a test of independence between *X* and *Y* that involves the difference between the observed and expected frequencies. The expected cell frequencies under the null hypothesis of independence are estimated by $\hat{N}_{ij} = N_i.N._j/N$. Under the null hypothesis, Pearson's chi-square converges asymptotically to a chi-square distribution χ_d^2 with degrees of freedom $d = (I–1)(J–1)$.

The *p* value based on Pearson's chi-square X^2 is calculated by *p* value = Prob($\chi_d^2 > X^2$), where

$$X^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \left(N_{ij} - \hat{N}_{ij}\right)^2 / \hat{N}_{ij}.$$

Predictors are ranked by the following rules.

1. Sort the predictors by *p* value in the ascending order
2. If ties occur, sort by chi-square in descending order.
3. If ties still occur, sort by degree of freedom *d* in ascending order.
4. If ties still occur, sort by the data file order.

P Value Based on Likelihood Ratio Chi-square

The likelihood ratio chi-square is a test of independence between *X* and *Y* that involves the ratio between the observed and expected frequencies. The expected cell frequencies under the null hypothesis of independence are estimated by $\hat{N}_{ij} = N_i.N._j/N$. Under the null hypothesis, the likelihood ratio chi-square converges asymptotically to a chi-square distribution χ_d^2 with degrees of freedom $d = (I–1)(J–1)$.

The *p* value based on likelihood ratio chi-square G^2 is calculated by *p* value = Prob($\chi_d^2 > G^2$), where

$$G^2 = 2\sum_{i=1}^{I} \sum_{j=1}^{J} G_{ij}^2, \text{ with } G_{ij}^2 = \begin{cases} N_{ij} \ln\left(N_{ij}/\hat{N}_{ij}\right) & N_{ij} > 0, \\ 0 & \text{else.} \end{cases}$$

Predictors are ranked according to the same rules as those for the *p* value based on Pearson's chi-square.

Cramer's V

Cramer's *V* is a measure of association, between 0 and 1, based upon Pearson's chi-square. It is defined as

$$V = \left(\frac{X^2}{N(\min\{I,J\}-1)}\right)^{1/2}.$$

Predictors are ranked by the following rules:

1. Sort predictors by Cramer's *V* in descending order.
2. If ties occur, sort by chi-square in descending order.
3. If ties still occur, sort by data file order.

Lambda

Lambda is a measure of association that reflects the proportional reduction in error when values of the independent variable are used to predict values of the dependent variable. A value of 1 means that the independent variable perfectly predicts the dependent variable. A value of 0 means that the independent variable is no help in predicting the dependent variable. It is computed as

$$\lambda(Y|X) = \frac{\sum_i \max_j (N_{ij}) - \max_j (N_{\cdot j})}{N - \max_j (N_{\cdot j})}.$$

Predictors are ranked by the following rules:

1. Sort predictors by lambda in descending order.
2. If ties occur, sort by *I* in ascending order.
3. If ties still occur, sort by data file order.

All Continuous Predictors

If all predictors are continuous, *p* values based on the *F* statistic are used. The idea is to perform a one-way ANOVA *F* test for each continuous predictor; this tests if all the different classes of *Y* have the same mean as *X*.

The following notation applies:

Table 17-2
Notation

Notation	Description
N_j	The number of cases with *Y* = *j*.
$\overline{x}_j$	The sample mean of predictor *X* for target class *Y* = *j*.
s_j^2	The sample variance of predictor *X* for target class *Y* = *j*. $s_j^2 = \sum_{i=1}^{N_j} (x_{ij} - \overline{x}_j)^2/(N_j - 1)$
$\overline{\overline{x}}$	The grand mean of predictor *X*. $\overline{\overline{x}} = \sum_{j=1}^{J} N_j \overline{x}_j / N$

The above notations are based on nonmissing pairs of (*X*, *Y*).

P Value Based on the F Statistic

The *p* value based on the *F* statistic is calculated by *p* value = Prob{*F*(*J*–1, *N*–*J*) > *F*}, where

$$F = \frac{\sum_{j=1}^{J} N_j \left(\overline{x}_j - \overline{\overline{x}}\right)^2 / (J-1)}{\sum_{j=1}^{J} \left(N_j - 1\right) s_j^2 / (N-J)},$$

and *F*(*J*–1, *N*–*J*) is a random variable that follows an *F* distribution with degrees of freedom *J*–1 and *N*–*J*. If the denominator for a predictor is zero, set the *p* value = 0 for the predictor.

Predictors are ranked by the following rules:

1. Sort predictors by *p* value in ascending order.
2. If ties occur, sort by *F* in descending order.
3. If ties still occur, sort by *N* in descending order.
4. If ties still occur, sort by the data file order.

Mixed Type Predictors

If some predictors are continuous and some are categorical, the criterion for continuous predictors is still the *p* value based on the *F* statistic, while the available criteria for categorical predictors are restricted to the *p* value based on Pearson's chi-square or the *p* value based on the likelihood ratio chi-square. These *p* values are comparable and therefore can be used to rank the predictors.

Predictors are ranked by the following rules:

1. Sort predictors by *p* value in ascending order.
2. If ties occur, follow the rules for breaking ties among all categorical and all continuous predictors separately, then sort these two groups (categorical predictor group and continuous predictor group) by the data file order of their first predictors.

Continuous Target

This section describes ranking of predictors for a continuous target under the following scenarios:

- All predictors categorical
- All predictors continuous
- Some predictors categorical, some continuous

All Categorical Predictors

If all predictors are categorical and the target is continuous, *p* values based on the *F* statistic are used. The idea is to perform a one-way ANOVA *F* test for the continuous target using each categorical predictor as a factor; this tests if all different classes of *X* have the same mean as *Y*.

The following notation applies:

Table 17-3
Notation

Notation	Description
X	The categorical predictor under consideration with *I* categories.
Y	The continuous target variable. y_{ij} represents the value of the continuous target for the j^{th} case with $X = i$.
Ni	The number of cases with $X = i$.
$\overline{y}_i$	The sample mean of target *Y* in predictor category $X = i$.
$s(y)_i^2$	The sample variance of target *Y* for predictor category $X = i$. $s(y)i2 = \sum_{j=1}^{N_i} (y_{ij} - \overline{y}_i)^2/(N_i - 1)$
$\overline{\overline{y}}$	The grand mean of target *Y*. $\overline{\overline{y}} = \Sigma_{i=1}^{I} N_i \overline{y}_i / N$

The above notations are based on nonmissing pairs of (*X*, *Y*).

The *p* value based on the *F* statistic is *p* value = Prob$\{F(I-1, N-I) > F\}$, where

$$F = \frac{\sum_{i=1}^{I} N_i \left(\overline{y}_i - \overline{\overline{y}}\right)^2 /(I-1)}{\sum_{i=1}^{I} (N_i - 1)\, s(y)_i^2 /(N-I)},$$

in which $F(I-1, N-I)$ is a random variable that follows a *F* distribution with degrees of freedom $I-1$ and $N-I$. When the denominator of the above formula is zero for a given categorical predictor *X*, set the *p* value = 0 for that predictor.

Predictors are ranked by the following rules:

1. Sort predictors by *p* value in ascending order.
2. If ties occur, sort by *F* in descending order.
3. If ties still occur, sort by *N* in descending order.
4. If ties still occur, sort by the data file order.

All Continuous Predictors

If all predictors are continuous and the target is continuous, the *p* value is based on the asymptotic *t* distribution of a transformation *t* on the Pearson correlation coefficient *r*.

The following notation applies:

Table 17-4
Notation

Notation	Description
X	The continuous predictor under consideration.
Y	The continuous target variable.
$\overline{x} = \Sigma_{i=1}^{N} x_i / N$	The sample mean of predictor variable *X*.
$\overline{y} = \Sigma_{i=1}^{N} y_i / N$	The sample mean of target *Y*.
$s(x)^2$	The sample variance of predictor variable *X*.
$s(y)^2$	The sample variance of target variable *Y*.

The above notations are based on nonmissing pairs of (*X*, *Y*).

The Pearson correlation coefficient *r* is

$$r = \frac{\Sigma_{i=1}^{N} (x_i - \overline{x}\big)(y_i - \overline{y})/(N-1)}{\sqrt{s(x)^2 s(y)^2}}.$$

The transformation *t* on *r* is given by

$$t = r\sqrt{\frac{N-2}{1-r^2}}.$$

Under the null hypothesis that the population Pearson correlation coefficient $\rho = 0$, the *p* value is calculated as

$$p\ value = \begin{cases} 0 & \text{if } r^2 = 1, \\ 2\ \text{Prob}\{T > |t|\} & \text{else.} \end{cases}$$

T is a random variable that follows a *t* distribution with *N*–2 degrees of freedom. The *p* value based on the Pearson correlation coefficient is a test of a linear relationship between *X* and *Y*. If there is some nonlinear relationship between *X* and *Y*, the test may fail to catch it.

Predictors are ranked by the following rules:

1. Sort predictors by *p* value in ascending order.
2. If ties occur in, sort by r^2 in descending order.
3. If ties still occur, sort by *N* in descending order.
4. If ties still occur, sort by the data file order.

Mixed Type Predictors

If some predictors are continuous and some are categorical in the dataset, the criterion for continuous predictors is still based on the *p* value from a transformation and that for categorical predictors from the *F* statistic.

Predictors are ranked by the following rules:

1. Sort predictors by p value in ascending order.
2. If ties occur, follow the rules for breaking ties among all categorical and all continuous predictors separately, then sort these two groups (categorical predictor group and continuous predictor group) by the data file order of their first predictors.

Selecting Predictors

If the length of the predictor list has not been prespecified, the following formula provides an automatic approach to determine the length of the list.

Let L_0 be the total number of predictors under study. The length of the list L may be determined by

$$L = \left[\min\left(\max\left(30, 2\sqrt{L_0}\right), L_0\right)\right],$$

where $[x]$ is the closest integer of x. The following table illustrates the length L of the list for different values of the total number of predictors L_0.

L_0	L	L/L_0(%)
10	10	100.00%
15	15	100.00%
20	20	100.00%
25	25	100.00%
30	30	100.00%
40	30	75.00%
50	30	60.00%
60	30	50.00%
100	30	30.00%
500	45	9.00%
1000	63	6.30%
1500	77	5.13%
2000	89	4.45%
5000	141	2.82%
10,000	200	2.00%
20,000	283	1.42%
50,000	447	0.89%

Generated Model

The feature selection generated model is different from most other generated models in that it does not add predictors or other derived fields to the data stream. Instead, it acts as a filter, removing unwanted fields from the data stream based on generated model settings.

The set of fields filtered from the stream is controlled by one of the following criteria:

- Field importance categories (**Important**, **Marginal**, or **Unimportant**). Fields assigned to any of the selected categories are preserved; others are filtered.
- Top *k* fields. The *k* fields with the highest importance values are preserved; others are filtered.
- Importance value. Fields with importance value greater than the specified value are preserved; others are filtered.
- Manual selection. The user can select specific fields to be preserved or filtered.

GENLIN Algorithms

Generalized linear models (GZLM) are commonly used analytical tools for different types of data. Generalized linear models cover not only widely used statistical models, such as linear regression for normally distributed responses, logistic models for binary data, and log linear model for count data, but also many useful statistical models via its very general model formulation.

Generalized Linear Models

Generalized linear models were first introduced by Nelder and Wedderburn (1972) and later expanded by McCullagh and Nelder (1989). The following discussion is based on their works.

Notation

The following notation is used throughout this section unless otherwise stated:

Table 18-1
Notation

Notation	Description
n	Number of complete cases in the dataset. It is an integer and $n \geq 1$.
p	Number of parameters (including the intercept, if exists) in the model. It is an integer and $p \geq 1$.
p_X	Number of non-redundant columns in the design matrix. It is an integer and $p_X \geq 1$.
$\mathbf{y}$	$n \times 1$ dependent variable vector. The rows are the cases.
$\mathbf{r}$	$n \times 1$ vector of events for the binomial distribution; it usually represents the number of "successes." All elements are non-negative integers.
$\mathbf{m}$	$n \times 1$ vector of trials for the binomial distribution. All elements are positive integers and $m_i \geq r_i$, i=1,...,n.
$\boldsymbol{\mu}$	$n \times 1$ vector of expectations of the dependent variable.
$\boldsymbol{\eta}$	$n \times 1$ vector of linear predictors.
$\mathbf{X}$	$n \times p$ design matrix. The rows represent the cases and the columns represent the parameters. The ith row is $x_i = (x_{i1}, \ldots, x_{ip})^T$,$i$=1,...,$n$ with $x_{i1} = 1$ if the model has an intercept.
$\mathbf{O}$	$n \times 1$ vector of scale offsets. This variable can't be the dependent variable (**y**) or one of the predictor variables (**X**).
β	$p \times 1$ vector of unknown parameters. The first element in β is the intercept, if there is one.
$\boldsymbol{\omega}$	$n \times 1$ vector of scale weights. If an element is less than or equal to 0 or missing, the corresponding case is not used.
$\mathbf{f}$	$n \times 1$ vector of frequency counts. Non-integer elements are treated by rounding the value to the nearest integer. For values less than 0.5 or missing, the corresponding cases are not used.
N	Effective sample size. $N = \sum_{i=1}^{n} f_i$. If frequency count variable **f** is not used, $N = n$.

Model

A GZLM of **y** with predictor variables **X** has the form

$$\eta = g\left(\mathrm{E}\left(\mathbf{y}\right)\right) = \mathbf{X}\beta + \mathbf{O}, \quad \mathbf{y} \sim F$$

where $\boldsymbol{\eta}$ is the linear predictor; **O** is an offset variable with a constant coefficient of 1 for each observation; $g(.)$ is the monotonic differentiable link function which states how the mean of **y**, $E\left(\mathbf{y}\right) = \mu$, is related to the linear predictor $\boldsymbol{\eta}$; F is the response probability distribution. Choosing different combinations of a proper probability distribution and a link function can result in different models.

In addition, GZLM also assumes y_{i} are independent for i=1,….,n. Then for each observation, the model becomes

$$\eta_i = g\left(\mu_i\right) = x_i^{\mathrm{T}}\beta + o_i, \quad y_i \sim F$$

Notes

- **X** can be any combination of scale variables (covariates), categorical variables (factors), and interactions. The parameterization of **X** is the same as in the GLM procedure. Due to use of the over-parameterized model where there is a separate parameter for every factor effect level occurring in the data, the columns of the design matrix **X** are often dependent. Collinearity between scale variables in the data can also occur. To establish the dependencies in the design matrix, columns of $\mathbf{X}^{\mathrm{T}}\mathbf{\Psi}\mathbf{X}$, where $\mathbf{\Psi} =$ diag$(f_1\omega_1, \ldots f_n\omega_n)$,, are examined by using the sweep operator. When a column is found to be dependent on previous columns, the corresponding parameter is treated as redundant. The solution for redundant parameters is fixed at zero.
- When **y** is a binary dependent variable which can be character or numeric, such as "male"/"female" or 1/2, its values will be transformed to 0 and 1 with 1 typically representing a success or some other positive result. In this document, we assume to be modeling the probability of success. In this document, we assume that **y** has been transformed to 0/1 values and we always model the probability of success; that is, Prob(**y** = 1). Which original value should be transformed to 0 or 1 depends on what the reference category is. If the reference category is the last value, then the first category represents a success and we are modeling the probability of it. For example, if the reference category is the last value, "male" in "male"/"female" and 2 in 1/2 are the last values (since "male" comes later in the dictionary than "female") and would be transformed to 0, and "female" and 1 would be transformed to 1 as we model the probability of them, respectively. However, one way to change to model the probability of "male" and 2 instead is to specify the reference category as the first value. Note if original binary format is 0/1 and the reference category is the last value, then 0 would be transformed to 1 and 1 to 0.
- When **r**, representing the number of successes (or number of 1s) and **m**, representing the number of trials, are used for the binomial distribution, the response is the binomial proportion **y** = **r**/**m**.

Probability Distribution

GZLMs are usually formulated within the framework of the exponential family of distributions. The probability density function of the response Y for the exponential family can be presented as

$$f\left(y\right)=\exp\left\{\frac{y\theta-b\left(\theta\right)}{\phi/\omega}+c\left(y,\phi/\omega\right)\right\}$$

where θ is the canonical (natural) parameter, ϕ is the scale parameter related to the variance of y and ω is a known prior weight which varies from case to case. Different forms of b(θ) and c(y, ϕ/ω) will give specific distributions. In fact, the exponential family provides a notation that allows us to model both continuous and discrete (count, binary, and proportional) outcomes. Several are available including continuous ones: normal, inverse Gaussian, gamma; discrete ones: negative binomial, Poisson, binomial.

The mean and variance of y can be expressed as follows

$$E\left(y\right)=b^{'}\left(\theta\right)=\mu$$

$$Var\left(y\right)=b^{''}\left(\theta\right)\frac{\phi}{\omega}=V\left(\mu\right)\frac{\phi}{\omega}$$

where $b^{'}\left(\theta\right)$ and $b^{''}\left(\theta\right)$ denote the first and second derivatives of b with respect to θ, respectively; $V\left(\mu\right)$ is the variance function which is a function of μ.

In GZLM, the distribution of y is parameterized in terms of the mean (μ) and a scale parameter (ϕ) instead of the canonical parameter (θ). The following table lists the distribution of y, corresponding range of y, variance function (V(μ)), the variance of y (Var(y)), and the first derivative of the variance function ($V^{'}\left(\mu\right)$), which will be used later.

Table 18-2
Distribution, range and variance of the response, variance function, and its first derivative

Distribution	**Range of *y***	***V*(μ)**	**Var(*y*)**	***V*'(μ)**
Normal	(−∞,∞)	1	ϕ	0
Inverse Gaussian	(0,∞)	μ^3	$\phi\mu^3$	$3\mu^2$
Gamma	(0,∞)	μ^2	$\phi\mu^2$	2μ
Negative binomial	0(1)∞	$\mu+k\mu^2$	$\mu+k\mu^2$	1+2kμ
Poisson	0(1)∞	μ	μ	1
Binomial(m)	0(1)m/m	μ(1−μ)	μ(1−μ)/m	1−2μ

Notes

- 0(1)z means the range is from 0 to z with increments of 1; that is, 0, 1, 2, …, z.
- For the binomial distribution, the binomial trial variable m is considered as a part of the weight variable ω.
- If a weight variable ω is presented, ϕ is replaced by ϕ/ω.
- For the negative binomial distribution, the ancillary parameter (k) can be user-specified. When k = 0, the negative binomial distribution reduces to the Poisson distribution. When k = 1, the negative binomial is the geometric distribution.

Scale parameter handling. The expressions for $V(\mu)$ and Var(y) for continuous distributions include the scale parameter ϕ which can be used to scale the relationship of the variance and mean (Var(y) and μ). Since it is usually unknown, there are three ways to fit the scale parameter:

1. It can be estimated with β jointly by maximum likelihood method.
2. It can be set to a fixed positive value.
3. It can be specified by the deviance or Pearson chi-square. For more information, see the topic "Goodness-of-Fit Statistics " on p. 178.

On the other hand, discrete distributions do not have this extra parameter (it is theoretically equal to one). Because of it, the variance of y might not be equal to the nominal variance in practice (especially for Poisson and binomial because the negative binomial has an ancillary parameter k). A simple way to adjust this situation is to allow the variance of y for discrete distributions to have the scale parameter as well, but unlike continuous distributions, it can't be estimated by the ML method. So for discrete distributions, there are two ways to obtain the value of ϕ:

1. It can be specified by the deviance or Pearson chi-square.
2. It can be set to a fixed positive value.

To ensure the data fit the range of response for the specified distribution, we follow the rules:

- For the gamma or inverse Gaussian distributions, values of **y** must be real and greater than zero. If a value of **y** is less than or equal to 0 or missing, the corresponding case is not used.
- For the negative binomial and Poisson distributions, values of **y** must be integer and non-negative. If a value of **y** is non-integer, less than 0 or missing, the corresponding case is not used.
- For the binomial distribution and if the response is in the form of a single variable, **y** must have only two distinct values. If **y** has more than two distinct values, the algorithm terminates in an error.
- For the binomial distribution and the response is in the form of ratio of two variables denoted events/trials, values of **r** (the number of events) must be nonnegative integers, values of **m** (the number of trials) must be positive integers and $m_i \geq r_i$, $\forall\ i$. If a value of **r** is not integer, less than 0, or missing, the corresponding case is not used. If a value of **m** is not integer, less than or equal to 0, less than the corresponding value of **r**, or missing, the corresponding case is not used.

The ML method will be used to estimate β and possibly ϕ. The kernels of the log-likelihood function (ℓ_k) and the full log-likelihood function (ℓ), which will be used as the objective function for parameter estimation, are listed for each distribution in the following table. Using ℓ or ℓ_k won't affect the parameter estimation, but the selection will affect the calculation of information criteria. For more information, see the topic "Goodness-of-Fit Statistics " on p. 178.

Table 18-3
The log-likelihood function for probability distribution

Distribution	$\boldsymbol{\ell}_{\mathbf{k}}$ **and** $\boldsymbol{\ell}$
Normal	$\ell_k = \sum_{i=1}^{n} -\frac{f_i}{2}\left\{\frac{\omega_i(y_i-\mu_i)^2}{\phi} + \ln\left(\frac{\phi}{\omega_i}\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
Inverse Gaussian	$\ell_k = \sum_{i=1}^{n} -\frac{f_i}{2}\left\{\frac{\omega_i(y_i-\mu_i)^2}{\phi y_i \mu_i^2} + \ln\left(\frac{\phi y_i^3}{\omega_i}\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
Gamma	$\ell_k = \sum_{i=1}^{n} f_i\left\{\frac{\omega_i}{\phi}\ln\left(\frac{\omega_i y_i}{\phi\mu_i}\right) - \frac{\omega_i y_i}{\phi\mu_i} - \ln\left(\Gamma\left(\frac{\omega_i}{\phi}\right)\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\{-\ln(y_i)\}$
Negative binomial	$\ell_k = \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\{y_i\ln(k\mu_i) - (y_i+1/k)\ln(1+k\mu_i) + \ln(\Gamma(y_i+1/k)) - \ln(\Gamma(1/k))\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\{-\ln(\Gamma(y_i+1))\}$
Poisson	$\ell_k = \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\{y_i\ln(\mu_i) - \mu_i\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\{-\ln(y_i!)\}$
Binomial(*m*)	$\ell_k = \sum_{i=1}^{n} f_i\frac{\omega_i^*}{\phi}\{y_i\ln(\mu_i) + (1-y_i)\ln(1-\mu_i)\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\left\{\ln\binom{m_i}{r_i}\right\}$,where $\binom{m_i}{r_i} = \frac{m_i!}{r_i!(m_i-r_i)!}$

When an individual $y = 0$ for the negative binomial or Poissondistributions and $y = 0$ or 1 for the binomial distribution, a separate value of the log-likelihood is given. Let $\ell_{\mathrm{k,i}}$ be the log-likelihood value for individual case *i* when $y_{\mathrm{i}} = 0$ for the negative binomial and Poisson and 0/1 for the binomial. The full log-likelihood for *i* is equal to the kernel of the log-likelihood for *i*; that is, $\ell_{\mathrm{i}}=\ell_{\mathrm{k,i}}$.

Table 18-4
Log-likelihood

Distribution	$\boldsymbol{\ell}_{\mathbf{k,i}}$
Negative binomial	$\ell_{k,i} = -f_i\frac{\omega_i}{\phi}\frac{\ln(1+k\mu_i)}{k}$ if $y_i = 0$

Distribution	$\ell_{k,i}$
Poisson	$\ell_{k,i} = -f_i \frac{\omega_i}{\phi} \mu_i \;\text{ if }\; y_i = 0$
Binomial(m)	$\ell_{k,i} = \begin{cases} f_i \frac{\omega_i}{\phi} \ln(1-\mu_i) & \text{if}\, y_i = 0 \\ f_i \frac{\omega_i}{\phi} \ln(\mu_i) & \text{if}\, y_i = 1 \end{cases}$

- $\Gamma(z)$ is the gamma function and $\ln(\Gamma(z))$ is the log-gamma function (the logarithm of the gamma function), evaluated at z.
- For the negative binomial distribution, the scale parameter is still included in ℓ_k for flexibility, although it is usually set to 1.
- For the binomial distribution (**r**/**m**), the scale weight variable becomes $\omega_i^* = \omega_i m_i$ in ℓ_k; that is, the binomial trials variable **m** is regarded as a part of the weight. However, the scale weight in the extra term of ℓ is still ω_i.

Link Function

The following tables list the form, inverse form, range of $\hat{\mu}$, and first and second derivatives for each link function.

Table 18-5
Link function name, form, inverse of link function, and range of the predicted mean

Link function	$\eta = g(\mu)$	Inverse $\mu = g^{-1}(\eta)$	Range of $\hat{\mu}$
Identity	μ	η	$\hat{\mu} \in R$
Log	$\ln(\mu)$	$\exp(\eta)$	$\hat{\mu} \geq 0$
Logit	$\ln\left(\frac{\mu}{1-\mu}\right)$	$\frac{\exp(\eta)}{1+\exp(\eta)}$	$\hat{\mu} \in [0,1]$
Probit	$\Phi^{-1}(\mu),\ \text{where}$ $\Phi(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\xi} e^{-z^2/2} dz$	$\Phi(\eta)$	$\hat{\mu} \in [0,1]$
Complementary log-log	$\ln(-(\ln(1-\mu))$	$1-\exp(-\exp(\eta))$	$\hat{\mu} \in [0,1]$
Power(α)$\begin{cases} \alpha \neq 0 \\ \alpha = 0 \end{cases}$	$\begin{cases} \mu^\alpha \\ \ln(\mu) \end{cases}$	$\begin{cases} \eta^{1/\alpha} \\ \exp(\eta) \end{cases}$	$\begin{cases} \hat{\mu} \in R \text{ if } \alpha \text{ or} 1/\alpha \text{ is odd integer} \\ \hat{\mu} \geq 0 \text{ otherwise} \end{cases}$
Log-complement	$\ln(1-\mu)$	$1-\exp(\eta)$	$\hat{\mu} \leq 1$
Negative log-log	$-\ln(-\ln(\mu))$	$\exp(-\exp(-\eta))$	$\hat{\mu} \in [0,1]$
Negative binomial	$\ln\left(\frac{\mu}{\mu+\frac{1}{k}}\right)$	$\frac{\exp(\eta)}{k(1-\exp(\eta))}$	$\hat{\mu} \geq 0$
Odds power(α)$\begin{cases} \alpha \neq 0 \\ \alpha = 0 \end{cases}$	$\begin{cases} \frac{(\mu/(1-\mu))^\alpha - 1}{\alpha} \\ \ln\left(\frac{\mu}{1-\mu}\right) \end{cases}$	$\begin{cases} \frac{(1+\alpha\eta)^{1/\alpha}}{1+(1+\alpha\eta)^{1/\alpha}} \\ \frac{\exp(\eta)}{1+\exp(\eta)} \end{cases}$	$\hat{\mu} \in [0,1]$

Note: In the power link function, if |α| < 2.2e-16, α is treated as 0.

Table 18-6
The first and second derivatives of link function

Link function	First derivative $g'(\mu) = \frac{\partial\eta}{\partial\mu} = \Delta$	Second derivative $g''(\mu) = \frac{\partial^2\eta}{\partial\mu^2}$
Identity	1	0

Link function	**First derivative** $g^{'}(\mu) = \frac{\partial\eta}{\partial\mu} = \Delta$	**Second derivative** $g^{''}(\mu) = \frac{\partial^2\eta}{\partial\mu^2}$
Log	$\frac{1}{\mu}$	$-\Delta^2$
Logit	$\frac{1}{\mu(1-\mu)}$	$\Delta^2(2\mu-1)$
Probit	$\frac{1}{\phi\left(\Phi^{-1}(\mu)\right)}$, where $\phi(z) = \frac{1}{\sqrt{2\pi}}e^{-z^2/2}$	$\Delta^2\Phi^{-1}(\mu)$
Complementary log-log	$\frac{1}{(\mu-1)\ln(1-\mu)}$	$-\Delta^2(1+\ln(1-\mu))$
Power(α)$\begin{cases}\alpha\neq 0\\ \alpha=0\end{cases}$	$\begin{cases}\alpha\mu^{\alpha-1}\\ \frac{1}{\mu}\end{cases}$	$\begin{cases}\Delta\frac{\alpha-1}{\mu}\\ -\Delta^2\end{cases}$
Log-complement	$\frac{-1}{1-\mu}$	$-\Delta^2$
Negative log-log	$\frac{-1}{\mu\ln(\mu)}$	$\Delta^2(1+\ln(\mu))$
Negative binomial	$\frac{1}{\mu+k\mu^2}$	$-\Delta^2(1+2k\mu)$
Odds power(α)$\begin{cases}\alpha\neq 0\\ \alpha=0\end{cases}$	$\begin{cases}\frac{\mu^{\alpha-1}}{(1-\mu)^{\alpha+1}}\\ \frac{1}{\mu(1-\mu)}\end{cases}$	$\begin{cases}\Delta\left(\frac{\alpha-1}{\mu}+\frac{\alpha+1}{1-\mu}\right)\\ \Delta^2(2\mu-1)\end{cases}$

When the canonical parameter is equal to the linear predictor, $\theta = \eta$, then the link function is called the **canonical link function**. Although the canonical links lead to desirable statistical properties of the model, particularly in small samples, there is in general no a priori reason why the systematic effects in a model should be additive on the scale given by that link. The canonical link functions for probability distributions are given in the following table.

Table 18-7
Canonical and default link functions for probability distributions

Distribution	**Canonical link function**
Normal	Identity
Inverse Gaussian	Power(−2)
Gamma	Power(−1)
Negative binomial	Negative binomial
Poisson	Log
Binomial	Logit

Estimation

Having selected a particular model, it is required to estimate the parameters and to assess the precision of the estimates.

Parameter estimation

The parameters are estimated by maximizing the log-likelihood function (or the kernel of the log-likelihood function) from the observed data. Let **s** be the first derivative (gradient) vector of the log-likelihood with respect to each parameter, then we wish to solve

$$\mathbf{s} = \left[\frac{\partial \ell}{\partial \beta}\right]_{p\times 1} = 0$$

In general, there is no closed form solution except for a normal distribution with identity link function, so estimates are obtained numerically via an iterative process. A Newton-Raphson and/or Fisher scoring algorithm is used and it is based on a linear Taylor series approximation of the first derivative of the log-likelihood.

First Derivatives

If the scale parameter ϕ is not estimated by the ML method, **s** is a p×1 vector with the form:

$$\mathbf{s} = \sum_{i=1}^{n} \frac{f_i \omega_i (y_i - \mu_i)}{\phi V(\mu_i)\, g'(\mu_i)} \cdot x_i = \frac{1}{\phi}\sum_{i=1}^{n} \frac{f_i \omega_i (y_i - \mu_i)}{V(\mu_i)\, g'(\mu_i)} \cdot x_i$$

where $\mu_i, V(\mu_i)$ and $g'(\mu_i)$ are defined in Table 18-5“Link function name, form, inverse of link function, and range of the predicted mean” on p. 168, Table 18-2“Distribution, range and variance of the response, variance function, and its first derivative” on p. 165 and Table 18-6“The first and second derivatives of link function” on p. 168, respectively.

If the scale parameter ϕ is estimated by the ML method, it is handled by searching for ln(ϕ) since ϕ is required to be greater than zero.

Let τ = ln(ϕ) so ϕ = exp(τ) , then **s** is a (p+1)×1 vector with the following form

$$\mathbf{s} = \begin{bmatrix} \frac{\partial \ell}{\partial \beta} \\ \frac{\partial \ell}{\partial \tau} \end{bmatrix}_{(p+1)\times 1} = \begin{bmatrix} \frac{1}{\exp(\tau)} \sum_{i=1}^{n} \frac{f_i \omega_i (y_i - \mu_i)}{V(\mu_i)\, g'(\mu_i)} \cdot x_i \\ \frac{\partial \ell}{\partial \tau} \end{bmatrix}$$

where $\partial\ell/\partial\beta$ is the same as the above with ϕ is replaced with exp(τ), $\partial\ell/\partial\tau$ has a different form depending on the distribution as follows:

Table 18-8
The 1st derivative functions w.r.t. the scale parameter for probability distributions

Distribution	$\frac{\partial \ell}{\partial \tau}$
Normal	$\sum_{i=1}^{n} \frac{f_i}{2}\left\{\frac{\omega_i (y_i - \mu_i)^2}{\exp(\tau)} - 1\right\}$
Inverse Gaussian	$\sum_{i=1}^{n} \frac{f_i}{2}\left\{\frac{\omega_i (y_i - \mu_i)^2}{\exp(\tau)\, y_i \mu_i^2} - 1\right\}$
Gamma	$\sum_{i=1}^{n} -\frac{f_i \omega_i}{\exp(\tau)}\left\{\ln\left(\frac{\omega_i y_i}{\exp(\tau)\,\mu_i}\right) + \left(1 - \frac{y_i}{\mu_i}\right) - \psi\left(\frac{\omega_i}{\exp(\tau)}\right)\right\}$

Note: $\psi(z)$ is a digamma function, which is the derivative of logarithm of a gamma function, evaluated at z; that is, $\psi(z) = \frac{\partial \ln(\Gamma(z))}{\partial z} = \frac{\Gamma'(z)}{\Gamma(z)}$.

As mentioned above, for normal distribution with identity link function which is a classical linear regression model, there is a closed form solution for both β and τ, so no iterative process is needed. The solution for β, after applying the SWEEP operation in GLM procedure, is

$$\hat{\beta}=\left(\sum_{i=1}^{n} f_i\omega_i \mathrm{x}_i^{\mathrm{T}}\mathrm{x}_i\right)^{-}\left(\sum_{i=1}^{n} f_i\omega_i \mathrm{x}_i^{\mathrm{T}}(y_i-o_i)\right)=\left(\mathrm{X}^{\mathrm{T}}\Psi\mathrm{X}\right)^{-}\left(\mathrm{X}^{\mathrm{T}}\Psi(\mathbf{y}-\mathbf{o})\right),$$

where $\Psi=\mathrm{diag}(f_1\omega_1,\ldots f_n\omega_n)$ and $(\mathrm{Z})^{-}$ is the generalized inverse of a matrix **Z**. If the scale parameter ϕ is also estimated by the ML method, the estimate of τ is

$$\hat{\tau}=\ln\left(\hat{\phi}\right)=\ln\left(\frac{1}{N}\sum_{i=1}^{n} f_i\omega_i\left(y_i-\mathrm{x}_i^{\mathrm{T}}\hat{\beta}-o_i\right)^2\right).$$

Second Derivatives

Let **H** be the second derivative (Hessian) matrix. If the scale parameter is not estimated by the ML method, **H** is a *p*×*p* matrix with the following form

$$\mathbf{H}=\left[\frac{\partial^2\ell}{\partial\beta\partial\beta^{\mathrm{T}}}\right]_{p\times p}=-\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$$

where **W** is an *n*×*n* diagonal matrix. There are two definitions for **W** depending on which algorithm is used: $\mathbf{W}_{\mathrm{e}}$ for Fisher scoring and $\mathbf{W}_{\mathrm{o}}$ for Newton-Raphson. The *i*th diagonal element for $\mathbf{W}_{\mathrm{e}}$ is

$$w_{e,i}=\frac{f_i\omega_i}{\phi}\cdot\frac{1}{V\left(\mu_i\right)\left(g^{'}\left(\mu_i\right)\right)^2},$$

and the *i*th diagonal element for $\mathbf{W}_{\mathrm{o}}$ is

$$w_{o,i}=w_{e,i}+\frac{f_i\omega_i}{\phi}(y_i-\mu_i)\cdot\frac{V\left(\mu_i\right)g^{''}\left(\mu_i\right)+V^{'}\left(\mu_i\right)g^{'}\left(\mu_i\right)}{\left(V\left(\mu_i\right)\right)^2\left(g^{'}\left(\mu_i\right)\right)^3},$$

where $V^{'}\left(\mu_i\right)$ and $g^{''}\left(\mu_i\right)$ are defined in Table 18-2“Distribution, range and variance of the response, variance function, and its first derivative” on p. 165 and Table 18-6“The first and second derivatives of link function” on p. 168, respectively. Note the expected value of $\mathbf{W}_{\mathrm{o}}$ is $\mathbf{W}_{\mathrm{e}}$ and when the canonical link is used for the specified distribution, then $\mathbf{W}_{\mathrm{o}}=\mathbf{W}_{\mathrm{e}}$.

If the scale parameter is estimated by the ML method, **H** becomes a (*p*+1)×(*p*+1) matrix with the form

$$\mathbf{H}=\begin{bmatrix}\frac{\partial^2\ell}{\partial\beta\partial\beta^{\mathrm{T}}} & \frac{\partial^2\ell}{\partial\beta\partial\tau}\\ \frac{\partial^2\ell}{\partial\tau\partial\beta^{\mathrm{T}}} & \frac{\partial^2\ell}{\partial\tau^2}\end{bmatrix}_{(p+1)\times(p+1)}$$

where $\partial^2\ell/\partial\beta\partial\tau$ is a p×1 vector and $\partial^2\ell/\partial\tau\partial\beta^{\mathrm{T}}$ is a 1×p vector and the transpose of $\partial^2\ell/\partial\beta\partial\tau$. For all three continuous distributions:

$$\frac{\partial^2\ell}{\partial\beta\partial\tau}=\sum_{i=1}^{n}-\frac{f_i\omega_i(y_i-\mu_i)}{\exp\left(\tau\right)V\left(\mu_i\right)g'\left(\mu_i\right)}\cdot\mathrm{x}_i=-\frac{\partial\ell}{\partial\beta}.$$

The forms of $\partial^2\ell/\partial\tau^2$ are listed in the following table.

Table 18-9
The second derivative functions w.r.t. the scale parameter for probability distributions

Distribution	$\frac{\boldsymbol{\partial^2\ell}}{\boldsymbol{\partial\tau^2}}$
Normal	$\sum_{i=1}^{n}-\frac{f_i\omega_i}{2\exp\left(\tau\right)}\left(y_i-\mu_i\right)^2$
Inverse Gaussian	$\sum_{i=1}^{n}-\frac{f_i\omega_i}{2\exp\left(\tau\right)y_i\mu_i^2}\left(y_i-\mu_i\right)^2$
Gamma	$\sum_{i=1}^{n}\frac{f_i\omega_i}{\exp\left(\tau\right)}\left\{\ln\left(\frac{\omega_i y_i}{\exp\left(\tau\right)\mu_i}\right)+\left(2-\frac{y_i}{\mu_i}\right)-\psi\left(\frac{\omega_i}{\exp\left(\tau\right)}\right)-\frac{\omega_i}{\exp\left(\tau\right)}\psi'\left(\frac{\omega_i}{\exp\left(\tau\right)}\right)\right\}$

Note: $\psi'(z)$ is a trigamma function, which is the derivative of $\psi(z)$, evaluated at z.

Iterations

An iterative process to find the solution for β (which might include ϕ) is based on Newton-Raphson (for all iterations), Fisher scoring (for all iterations) or a hybrid method. The hybrid method consists of applying Fisher scoring steps for a specified number of iterations before switching to Newton-Raphson steps. Newton-Raphson performs well if the initial values are close to the solution, but the hybrid method can be used to improve the algorithm's robustness from bad initial values. Apart from improved robustness, Fisher scoring is faster due to the simpler form of the Hessian matrix.

The following notation applies to the iterative process:

Table 18-10
Notation

Notation	**Description**
I	Starting iteration for checking complete separation and quasi-complete separation. It must be 0 or a positive integer. This criterion is not used if the value is 0.
J	The maximum number of steps in step-halving method. It must be a positive integer.
K	The first number of iterations using Fisher scoring, then switching to Newton-Raphson. It must be 0 or a positive integer. A value of 0 means using Newton-Raphson for all iterations and a value greater or equal to M means using Fisher scoring for all iterations.
M	The maximum number of iterations. It must be a non-negative integer. If the value is 0, then initial parameter values become final estimates.
ϵ_ℓ, ϵ_{p},$\epsilon_{\mathbf{H}}$	Tolerance levels for three types of convergence criteria.
Abs	A 0/1 binary variable; *Abs* = 1 if absolute change is used for convergence criteria and *Abs* = 0 if relative change is used.

And the iterative process is outlined as follows:

1. Input values for *I*, *J*, *K*, *M*, ϵ_ℓ, $\epsilon_{\mathbf{p}}$,$\epsilon_{\mathbf{H}}$ and *Abs* for each type of three convergence criteria.
2. For $\beta^{(0)}$ compute initial values (see below), then calculate log-likelihood $\ell^{(0)}$, gradient vector $\mathbf{s}^{(0)}$ and Hessian matrix $\mathbf{H}^{(0)}$ based on $\beta^{(0)}$.
3. Let ξ=1.
4. Compute estimates of *i*th iteration:

   $\beta^{(i)} = \beta^{(i-1)} - \xi\left(\mathbf{H}^{(i-1)}\right)^{-}\mathbf{s}^{(i-1)}$, where $(\mathbf{H})^{-}$ is a generalized inverse of **H**. Then compute the log-likelihood based on $\beta^{(i)}$.
5. Use step-halving method if $\ell^{(i)} < \ell^{(i-1)}$: reduce ξ by half and repeat step (4). The set of values of ξ is $\{0.5^{\mathrm{j}} : \mathrm{j} = 0, \ldots, J - 1\}$. If *J* is reached but the log-likelihood is not improved, issue a warning message, then stop.
6. Compute gradient vector $\mathbf{s}^{(i)}$ and Hessian matrix $\mathbf{H}^{(i)}$ based on $\beta^{(i)}$. Note that $\mathbf{W}_{\mathrm{e}}$ is used to calculate $\mathbf{H}^{(i)}$ if $i \le K$; $\mathbf{W}_{\mathrm{o}}$ is used to calculate $\mathbf{H}^{(i)}$ if $i > K$.
7. Check if complete or quasi-complete separation of the data is established (see below) if distribution is binomial and the current iteration $i \ge I$. If either complete or quasi-complete separation is detected, issue a warning message, then stop.
8. Check if all three convergence criteria (see below) are met. If they are not but *M* is reached, issue a warning message, then stop.
9. If all three convergence criteria are met, check if complete or quasi-complete separation of the data is established if distribution is binomial and $i < I$ (because checking for complete or quasi-complete separation has not started yet). If complete or quasi-complete separation is detected, issue a warning message, then stop, otherwise, stop (the process converges for binomial successfully). If all three convergence criteria are met for the distributions other than binomial, stop (the process converges for other distributions successfully). The final vector of estimates is denoted by $\hat{\beta}$ (and $\hat{\tau}$). Otherwise, go back to step (3).

Initial Values

Initial values are calculated as follows:

1. Set the initial fitted values $\tilde{\mu}_{\mathrm{i}} = (y_i m_i + 0.5)/(m_i + 1)$ for a binomial distribution (y_{i} can be a proportion or 0/1 value) and $\tilde{\mu}_{\mathrm{i}} = y_i$ for a non-binomial distribution. From these derive $\tilde{\eta}_i$=$g\left(\tilde{\mu}_i\right)$,$g^{'}\left(\tilde{\mu}_i\right)$ and$V\left(\tilde{\mu}_i\right)$. If $\tilde{\eta}_i$ becomes undefined, set $\tilde{\eta}_i = 1$.
2. Calculate the weight matrix $\tilde{W}_e$ with the diagonal element $\tilde{w}_{ei} = \frac{f_i \omega_i}{\phi} \cdot \frac{1}{V(\tilde{\mu}_i)\left(g'(\tilde{\mu}_i)\right)^2}$, where ϕ is set to 1 or a fixed positive value. If the denominator of $\tilde{w}_{ei}$ becomes 0, set $\tilde{w}_{ei}$= 0.
3. Assign the adjusted dependent variable *z* with the *i*th observation $z_i = (\tilde{\eta}_i - o_i) + (y_i - \tilde{\mu}_i)\, g^{'}\left(\tilde{\mu}_i\right)$ for a binomial distribution and $z_i = (\tilde{\eta}_i - o_i)$ for a non-binomial distribution.

4. Calculate the initial parameter values

$$\beta^{(0)} = \left(\mathrm{X}^{\mathrm{T}}\tilde{W}_e\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{T}}\tilde{W}_e\mathrm{z}$$

and

$$\phi^{(0)}=\left(\mathrm{z}-\mathrm{X}\beta^{(0)}\right)^{\mathrm{T}}\tilde{W}_e\left(\mathrm{z}-\mathrm{X}\beta^{(0)}\right).$$

if the scale parameter is estimated by the ML method.

Scale Parameter Handling

1. For normal, inverse Gaussian, and gamma response, if the scale parameter is estimated by the ML method, then it will be estimated jointly with the regression parameters; that is, the last element of the gradient vector **s** is with respect to τ.
2. If the scale parameter is set to be a fixed positive value, then it will be held fixed at that value for in each iteration of the above process.
3. If the scale parameter is specified by the deviance or Pearson chi-square divided by degrees of freedom, then it will be fixed at 1 to obtain the regression estimates through the whole iterative process. Based on the regression estimates, calculate the deviance and Pearson chi-square values and obtain the scale parameter estimate.

Checking for Separation

For each iteration after the user-specified number of iterations; that is, if *i* > *I*, calculate (note here *v* refers to cases in the dataset)

$$p_{\min} = \min_v p_v$$

$$p_{\max} = \max_v p_v,$$

$$p^*_{\min} = \min_v\left(\min\left(\mu_v, 1-\mu_v\right)\right),$$

where

$$p_v = \begin{cases} \mu_v & \text{if } y_v = \text{success } (=1) \\ 1-\mu_v & \text{if } y_v = \text{ failure } (=0) \end{cases}$$

(p_v is the probability of the observed response for case *v*) and $\mu_v = g^{-1}\left(\mathrm{x}_v^{\mathrm{T}}\beta + o_v\right)$.

If $\min\left(p_{\min}, p_{\max}\right) = p_{\min} > 0.99$ we consider there to be complete separation. Otherwise, if $p_{\max} > 0.99$ or $p^*_{\min} < 0.001$ and if there are very small diagonal elements (absolute value $< \sqrt{10^{-7}} \approx 3.16 \times 10^{-4}$) in the non-redundant parameter locations in the lower triangular matrix in Cholesky decomposition of –**H**, where **H** is the Hessian matrix, then there is a quasi-complete separation.

Convergence Criteria

The following convergence criteria are considered:

$$\text{Log-likelihood convergence: } \begin{cases} \frac{\left|\ell^{(i)}-\ell^{(i-1)}\right|}{\left|\ell^{(i-1)}\right|+10^{-6}} < \epsilon_\ell \text{ if relative change} \\ \left|\ell^{(i)} - \ell^{(i-1)}\right| < \epsilon_\ell \text{ if absolute change} \end{cases}.$$

$$\text{Parameter convergence: } \begin{cases} \max_j \left(\frac{\left|\beta_j^{(i)}-\beta_j^{(i-1)}\right|}{\left|\beta_j^{(i-1)}\right|+10^{-6}} \right) < \epsilon_{\text{P}} \text{ if relative change} \\ \max_j \left(\left|\beta_j^{(i)} - \beta_j^{(i-1)}\right| \right) < \epsilon_{\text{P}} \text{ if absolute change} \end{cases}.$$

$$\text{Hessian convergence: } \begin{cases} \frac{\left(\mathbf{s}^{(i)}\right)^{\text{T}}\left(\mathbf{H}^{(i)}\right)^{-}\left(\mathbf{s}^{(i)}\right)}{\left|\ell^{(v)}\right|+10^{-6}} < \epsilon_{\mathbf{H}} \text{ if relative change} \\ \left(\mathbf{s}^{(i)}\right)^{\text{T}}\left(\mathbf{H}^{(i)}\right)^{-}\left(\mathbf{s}^{(i)}\right) < \epsilon_{\mathbf{H}} \text{ if absolute change} \end{cases}.$$

where $\epsilon_\ell, \epsilon_{\text{P}}$ and $\epsilon_{\mathbf{H}}$ are the given tolerance levels for each type.

If the Hessian convergence criterion is not user-specified, it is checked based on absolute change with ϵ_{H} = 1E-4 after the log-likelihood or parameter convergence criterion has been satisfied. If Hessian convergence is not met, a warning is displayed.

Parameter Estimate Covariance Matrix, Correlation Matrix and Standard Errors

The parameter estimate covariance matrix, correlation matrix and standard errors can be obtained easily with parameter estimates. Whether or not the scale parameter is estimated by ML, parameter estimate covariance and correlation matrices are listed for $\hat{\beta}$ only because the covariance between $\hat{\beta}$ and $\hat{\tau}$ should be zero.

Model-Based Parameter Estimate Covariance

The model-based parameter estimate covariance matrix is given by

$$\Sigma_{\text{m}} = -\text{H}^{-} = -(-\mathbf{XWX})^{-}$$

where $\mathbf{H}^{-}$ is the generalized inverse of the Hessian matrix evaluated at the parameter estimates. The corresponding rows and columns for redundant parameter estimates should be set to zero.

Robust Parameter Estimate Covariance

The validity of the parameter estimate covariance matrix based on the Hessian depends on the correct specification of the variance function of the response in addition to the correct specification of the mean regression function of the response. The robust parameter estimate covariance provides a consistent estimate even when the specification of the variance function of the response is incorrect. The robust estimator is also called Huber's estimator because Huber (1967) was the first to describe this variance estimate; White's estimator or HCCM (heteroskedasticity consistent covariance matrix) estimator because White (1980) independently showed that this variance estimate is consistent under a linear regression model including heteroskedasticity; or the sandwich estimator because it includes three terms. The robust (or Huber/White/sandwich) estimator is defined as follows

$$\Sigma_{\mathrm{r}} = \Sigma_{\mathrm{m}}\left(\sum_{i=1}^{n}\left[\frac{\partial \ell_i}{\partial\beta}\right]\left[\frac{\partial \ell_i}{\partial\beta}\right]^{\mathrm{T}}\right)\Sigma_{\mathrm{m}} = \Sigma_{\mathrm{m}}\left(\sum_{i=1}^{n} f_i\left(\frac{\omega_i(y_i-\mu_i)}{\phi V\left(\mu_i\right) g'\left(\mu_i\right)}\right)^2\cdot x_i\cdot x_i^{\mathrm{T}}\right)\Sigma_{\mathrm{m}}$$

Parameter Estimate Correlation

The correlation matrix is calculated from the covariance matrix as usual. Let σ_{ij} be an element of Σ_{m} or Σ_{r}, then the corresponding element of the correlation matrix is $\frac{\sigma_{ij}}{\sqrt{\sigma_{ii}}\sqrt{\sigma_{jj}}}$. The corresponding rows and columns for redundant parameter estimates should be set to system missing values.

Parameter Estimate Standard Error

Let $\hat{\beta}_i$ denote a non-redundant parameter estimate. Its standard error is the square root of the *i*th diagonal element of Σ_{m} or Σ_{r}:

$$\hat{\sigma}_{\beta_i} = \sqrt{\sigma_{ii}}$$

The standard error for redundant parameter estimates is set to a system missing value. If the scale parameter is estimated by the ML method, we obtain $\hat{\tau}$ and its standard error estimate $\hat{\sigma}_{\tau} = \sqrt{-\frac{1}{\left(\frac{\partial^2\ell}{\partial\tau^2}\right)}}$, where $\frac{\partial^2\ell}{\partial\tau^2}$ can be found in Table 18-9“The second derivative functions w.r.t. the scale parameter for probability distributions” on p. 172. Then the estimate of the scale parameter is $\exp\left(\hat{\tau}\right)$ and the standard error estimate is $\left(\exp\left(\hat{\tau}\right)\cdot\hat{\sigma}_{\tau}\right)$

Wald Confidence Intervals

Wald confidence intervals are based on the asymptotic normal distribution of the parameter estimates. The 100(1 – α)% Wald confidence interval for β_{j} is given by

$$\left(\hat{\beta}_j - z_{1-\alpha/2}\hat{\sigma}_{\beta_j}, \hat{\beta}_j + z_{1-\alpha/2}\hat{\sigma}_{\beta_j}\right),$$

where z_p is the 100*p*th percentile of the standard normal distribution.

If exponentiated parameter estimates are requested for logistic regression or log-linear models, then using the delta method, the estimate of $\exp\left(\beta_j\right)$ is $\exp\left(\hat{\beta}_j\right)$, the standard error estimate of $\exp\left(\hat{\beta}_j\right)$ is $\left(\exp\left(\hat{\beta}_j\right)\cdot\hat{\sigma}_{\beta_j}\right)$ and the corresponding 100(1 – α)% Wald confidence interval for $\exp\left(\beta_j\right)$ is

$$\left(\exp\left(\hat{\beta}_j - z_{1-\alpha/2}\hat{\sigma}_{\beta_j}\right), \exp\left(\hat{\beta}_j + z_{1-\alpha/2}\hat{\sigma}_{\beta_j}\right)\right).$$

Wald confidence intervals for redundant parameter estimates are set to system missing values.

Similarly, the 100(1 – α)% Wald confidence interval for $\phi$is

$$\left(\exp\left(\hat{\tau} - z_{1-\alpha/2}\hat{\sigma}_{\tau}\right), \exp\left(\hat{\tau} + z_{1-\alpha/2}\hat{\sigma}_{\tau}\right)\right)$$

Chi-Square Statistics

The hypothesis $H_{0i} : \beta_i = 0$ is tested for each non-redundant parameter using the chi-square statistic:

$$c_i = \left(\frac{\hat{\beta}_i}{\hat{\sigma}_{\beta_j}}\right)^2$$

which has an asymptotic chi-square distribution with 1 degree of freedom.

Chi-square statistics and their corresponding *p*-values are set to system missing values for redundant parameter estimates.

The chi-square statistic is not calculated for the scale parameter, even if it is estimated by ML method.

P Values

Given a test statistic *T* and a corresponding cumulative distribution function *G* as specified above, the *p*-value is defined as $p = 1 - G\left(T\right)$. For example, the *p*-value for the chi-square test of $H_{0i} : \beta_i = 0$ is $p_i = 1 - prob\left(\chi_1^2 \leq c_i\right)$.

Model Testing

After estimating parameters and calculating relevant statistics, several tests for the given model are performed.

Lagrange Multiplier Test

If the scale parameter for normal, inverse Gaussian and gamma distributions is set to a fixed value or specified by the deviance or Pearson chi-square divided by the degrees of freedom (when the scale parameter is specified by the deviance or Pearson chi-square divided by the degrees of freedom, it can be considered as a fixed value), or an ancillary parameter *k* for the negative binomial is set to a fixed value other than 0, the Lagrange Multiplier (LM) test assesses the validity of the value. For a fixed ϕ or *k*, the test statistic is defined as

$$T_{LM} = \frac{s^2}{A}$$

where $s = \partial\ell/\partial\tau$ and $A = -\left(\frac{\partial^2\ell}{\partial\tau^2}\right) - \left(-\frac{\partial^2\ell}{\partial\tau\partial\beta^{\mathrm{T}}}\right)\left(-\frac{\partial^2\ell}{\partial\beta\partial\beta^{\mathrm{T}}}\right)^{-}\left(-\frac{\partial^2\ell}{\partial\beta\partial\tau}\right)$ evaluated at the parameter estimates and fixed ϕ or k value. T_{LM} has an asymptotic chi-square distribution with 1 degree of freedom, and the p-values are calculated accordingly.

For testing ϕ, see Table 18-8"The 1st derivative functions w.r.t. the scale parameter for probability distributions" on p. 170 and see Table 18-9"The second derivative functions w.r.t. the scale parameter for probability distributions" on p. 172 for the elements of s and A, respectively.

If k is set to 0, then the above statistic can't be applied. According to Cameron and Trivedi (1998), the LM test statistic should now be based on the following auxiliary OLS regression (without constant)

$$\frac{(y_i-\hat{\mu}_i)^2-y_i}{\hat{\mu}_i}= \alpha\hat{\mu}_i + \epsilon_i$$

where $\hat{\mu}_i= g^{-1}\left(x_i^T\hat{\beta}\right)$ and ϵ_i is an error term. Let the response of the above OLS regression $\left[(y_i - \hat{\mu}_i)^2 - y_i\right]/\hat{\mu}_i$ be z_i and the explanatory variable $\hat{\mu}_i$ be w_i. The estimate of the above regression parameter α and the standard error of the estimate of α are

$$\hat{\alpha} = \frac{\sum_{i=1}^{n} f_i w_i z_i}{\sum_{i=1}^{n} f_i w_i^2} \text{ and } \hat{\sigma}_\alpha = \sqrt{\frac{s_e^2}{\sum_{i=1}^{n} f_i w_i^2}},$$

where $s_e^2 = \frac{1}{N-1}\sum_{i=1}^{n} f_i e_i^2$ and $e_i = z_i - \hat{\alpha}w_i$. Then the LM test statistic is a z statistic

$$z = \frac{\hat{\alpha}}{\hat{\sigma}_\alpha},$$

and it has an asymptotic standard normal distribution under the null hypothesis of equidispersion in a Poisson model ($H_0 : k = 0$). Three p-values are provided. The alternative hypothesis can be one-sided overdispersion ($H_a : k > 0$), underdispersion ($H_a : k < 0$) or two-sided non-directional ($H_a : k \neq 0$) with the variance function of $V(\mu) = \mu + k\mu^2$. The calculation of p-values depends on the alternative. For $H_a : k > 0$,p-value $= 1 - \Phi(z)$, where $\Phi(\cdot)$ is the cumulative probability of a standard normal distribution; for $H_a : k < 0$,p-value $= \Phi(z)$; and for $H_a : k \neq 0$,p-value $= 2(1 - \Phi(|z|))$.

Goodness-of-Fit Statistics

Several statistics are calculated to assess goodness of fit of a given generalized linear model.

Deviance

The theoretical definition of deviance is:

$$D = 2\phi(\ell(\mathrm{y};\mathrm{y}) - \ell(\hat{\mu};\mathrm{y})),$$

where $\ell\left(\hat{\mu};\mathrm{y}\right)$ is the log-likelihood function expressed as the function of the predicted mean values $\hat{\mu}$ (calculated based on the parameter estimates) given the response variable, and $\ell\left(\mathrm{y};\mathrm{y}\right)$ is the log-likelihood function computed by replacing $\hat{\mu}$ with **y**. The formula used for the deviance is $\sum_{i=1}^{n} f_i d_i$, where the form of di for the distributions are given in the following table:

Table 18-11
Deviance for individual case

Distribution	***di***
Normal	$\omega_i(y_i-\mu_i)^2$
Inverse Gaussian	$\frac{\omega_i}{y_i\mu_i^2}(y_i-\mu_i)^2$
Gamma	$2\omega_i\left\{-\ln\left(\frac{y_i}{\mu_i}\right)+\frac{y_i-\mu_i}{\mu_i}\right\}$
Negative Binomial	$2\omega_i\left\{y_i\ln\left(\frac{y_i}{\mu_i}\right)-(y_i+1/k)\ln\left(\frac{y_i+1/k}{\mu_i+1/k}\right)\right\}$
Poisson	$2\omega_i\left\{y_i\ln\left(\frac{y_i}{\mu_i}\right)-(y_i-\mu_i)\right\}$
Binomial(*m*)	$2\omega_i^*\left\{y_i\ln\left(\frac{y_i}{\mu_i}\right)+(1-y_i)\ln\left(\frac{1-y_i}{1-\mu_i}\right)\right\}$

Note

- When **y** is a binary dependent variable with 0/1 values (binomial distribution), the deviance and Pearson chi-square are calculated based on the subpopulations; see below.
- When y = 0 for negative binomial and Poisson distributions and y = 0 (for r = 0) or 1 (for r = m) for binomial distribution with **r**/**m** format, separate values are given for the deviance. Let d_i be the deviance value for individual case i when y_i = 0 for negative binomial and Poisson and 0/1 for binomial.

Table 18-12
Deviance for individual case

Distribution	$\boldsymbol{d_i}$
Negative Binomial	$2\omega_i\frac{\ln(1+k\mu_i)}{k}$ if$y_i=0$
Poisson	$2\omega_i\mu_i$ if$y_i=0$
Binomial(*m*)	$\begin{cases}-2\omega_i^*\ln(1-\mu_i)\text{ if } y_i=0 \text{ or } r_i=0\\ -2\omega_i^*\ln(\mu_i)\ \text{ if } y_i=1 \text{ or } r_i=m_i\end{cases}$

Pearson Chi-Square

$$\chi^2=\sum_{i=1}^{n} f_i\gamma_i$$

where $\gamma_i=\frac{\omega_i^*(y_i-\mu_i)^2}{V(\mu_i)}$ for the binomial distribution and $\gamma_i=\frac{\omega_i(y_i-\mu_i)^2}{V(\mu_i)}$ for other distributions.

Scaled Deviance and Scaled Pearson Chi-Square

The scaled deviance is $D^*=D/\phi$ and the scaled Pearson chi-square is $\chi^{2*}=\chi^2/\phi$.

Since the scaled deviance and Pearson chi-square statistics have a limiting chi-square distribution with $N - p_X$ degrees of freedom, the deviance or Pearson chi-square divided by its degrees of freedom can be used as an estimate of the scale parameter for both continuous and discrete distributions.

$\hat{\phi} = \frac{D}{N - p_x}$ or $\hat{\phi} = \frac{\chi^2}{N - p_x}$.

If the scale parameter is measured by the deviance or Pearson chi-square, first we assume $\phi = 1$, then estimate the regression parameters, calculate the deviance and Pearson chi-square values and obtain the scale parameter estimate from the above formula. Then the scaled version of both statistics is obtained by dividing the deviance and Pearson chi-square by $\hat{\phi}$. In the meantime, some statistics need to be revised. The gradient vector and the Hessian matrix are divided by $\hat{\phi}$ and the covariance matrix is multiplied by $\hat{\phi}$. Accordingly the estimated standard errors are also adjusted, the Wald confidence intervals and significance tests will be affected even the parameter estimates are not affected by $\hat{\phi}$.

Note that the log-likelihood is not revised; that is, the log-likelihood is based on $\phi = 1$ because the scale parameter should be kept the same in the log-likelihood for fair comparison in information criteria and model fitting omnibus test.

Overdispersion

For the Poisson and binomial distributions, if the estimated scale parameter is not near the assumed value of one, then the data may be overdispersed if the value is greater than one or underdispersed if the value is less than one. Overdispersion is more common in practice. The problem with overdispersion is that it may cause standard errors of the estimated parameters to be underestimated. A variable may appear to be a significant predictor, when in fact it is not.

Deviance and Pearson Chi-Square for Binomial Distribution with 0/1 Binary Response Variable

When **r** and **m** (event/trial) variables are used for the binomial distribution, each case represents m Bernoulli trials. When **y** is a binary dependent variable with 0/1 values, each case represents a single trial. The trial can be repeated for several times with the same setting (i.e. the same values for all predictor variables). For example, suppose the first 10 *y* values are 2 1s and 8 0s and **x** values are the same (if recorded in events/trials format, these 10 cases is recorded as 1 case with r = 2 and m = 10), then these 10 cases should be considered from the same subpopulation. Cases with common values in the variable list that includes all predictor variables are regarded as coming from the same subpopulation. When the binomial distribution with binary response is used, we should calculate the deviance and Pearson chi-square based on the subpopulations. If we calculate them based on the cases, the results might not be useful.

If subpopulations are specified for the binomial distribution with 0/1 binary response variable, the data should be reconstructed from the single trial format to the events/trials format. Assume the following notation for formatted data:

Table 18-13
Notation

Notation	**Description**
n_S	Number of subpopulations.

Notation	Description
r_{j1}	Sum of the product of the frequencies and the scale weights associated with $y = 1$ in the jth subpopulation. So r_{j0} is that with $y = 0$ in the jth subpopulation.
m_{j}	Total weighted observations; $m_{\mathrm{j}} = r_{\mathrm{j1}} + r_{\mathrm{j0}}$.
y_{j1}	The proportion of 1s in the jth subpopulation; $y_{\mathrm{j1}} = r_{\mathrm{j1}}/\, m_{\mathrm{j}}$.
μ_j	The fitted probability in the jth subpopulation ($\hat{\mu}_j$ would be the same for each case in the jth subpopulation because values for all predictor variables are the same for each case.)

The deviance and Pearson chi-square are defined as follows:

$$D = 2\sum_{j=1}^{n_s} m_j \left\{ y_{j1} \ln\left(\frac{y_{j1}}{\mu_j}\right) + (1-y_{j1}) \ln\left(\frac{1-y_{j1}}{1-\mu_j}\right) \right\} \text{ and } \chi^2 = \sum_{j=1}^{n_s} \frac{m_j (y_{j1}-\mu_j)^2}{\mu_j(1-\mu_j)},$$

then the corresponding estimate of the scale parameter will be

$\hat{\phi} = \frac{D}{n_s - p_x}$ and $\hat{\phi} = \frac{\chi^2}{n_s - p_x}$.

The full log likelihood, based on subpopulations, is defined as follows:

$$\ell = \ell_k + \sum_{j=1}^{n_s} \frac{1}{\phi} \left\{ \ln\binom{m_j}{r_{j1}} \right\} = \ell_k + \sum_{j=1}^{n_s} \frac{1}{\phi} \left\{ \ln \frac{m_j!}{r_{j1}! r_{j0}!} \right\},$$

where ℓ_k is the kernel log likelihood; it should be the same as the kernel log-likelihood computed based on cases before, there is no need to compute again.

Information Criteria

Information criteria are used when comparing different models for the same data. The formulas for various criteria are as follows.

Akaike information criteria (AIC). $-2\ell + 2d$

Finite sample corrected (AICC). $-2\ell + \frac{2d \cdot N}{(N-d-1)}$

Bayesian information criteria (BIC). $-2\ell + d\ln(N)$

Consistent AIC (CAIC). $-2\ell + d(\ln(N) + 1)$.

where ℓ is the log-likelihood evaluated at the parameter estimates. Notice that $d = p_{\mathrm{X}}$ if only β is included; $d = p_{\mathrm{X}} + 1$ if the scale parameter is included for normal, inverse Gaussian, or gamma.

Notes

- ℓ (the full log-likelihood) can be replaced with ℓ_{k} (the kernel of the log-likelihood) depending on the user's choice.
- When **r** and **m** (event/trial) variables are used for the binomial distribution, then the N used here would be the sum of the trials frequencies; $N = \sum_{i=1}^{n} f_i m_i$. In this way, the same value results whether the data are in raw, binary form or in summarized, binomial form.

Test of Model Fit

The model fitting omnibus test is based on –2 log-likelihood values for the model under consideration and the initial model. For the model under consideration, the value of the –2 log-likelihood is

$$-2\ell\left(\hat{\beta}\right)$$

Let the initial model be the intercept-only model if intercept is in the considered model or the empty model otherwise. For the intercept-only model, the value of the –2 log-likelihood is

$$-2\ell\left(\hat{\beta}_0\right)$$

For the empty model, the value of the –2 log-likelihood is

$$-2\ell(0)$$

Then the omnibus (or global) test statistic is

$S = 2\left(\ell\left(\hat{\beta}\right) - \ell(\beta_0)\right)$ for the intercept-only model or

$S = 2\left(\ell\left(\hat{\beta}\right) - \ell(0)\right)$ for the empty model.

S has an asymptotic chi-square distribution with *r* degrees of freedom, equal to the difference in the number of valid parameters between the model under consideration and the initial model. $r = p_x - 1$ for the intercept-only model,; $r = p_x$ for the empty model. The *p*-values then can be calculated accordingly.

Note if the scale parameter is estimated by the ML method in the model under consideration, then it will also be estimated by the ML method in the initial model.

Default Tests of Model Effects

For each regression effect specified in the model, type I and III analyses can be conducted.

Type I Analysis

Type I analysis consists of fitting a sequence of models, starting with a model with only an intercept term (if there is one), and adding one additional effect, which can be covariates, factors and interactions, of the model on each step. So it depends on the order of effects specified in the model. On the other hand, type III analysis won't depend on the order of effects.

Wald Statistics. For each effect specified in the model, type I test matrix $\mathbf{L}_\mathrm{i}$ is constructed and H_0: $\mathbf{L}_\mathrm{i}\beta = \mathbf{0}$ is tested. Construction of matrix $\mathbf{L}_\mathrm{i}$ is based on the generating matrix $\mathbf{H}_\omega = \left(\mathbf{X}^\mathrm{T}\mathbf{\Omega}\mathbf{X}\right)^{-}\mathbf{X}^\mathrm{T}\mathbf{\Omega}\mathbf{X}$, where $\mathbf{\Omega}$ is the scale weight matrix with *i*th diagonal element ω_i and such that $\mathbf{L}_\mathrm{i}\beta$ is estimable. It involves parameters only for the given effect and the effects containing the given effect. If such a matrix cannot be constructed, the effect is not testable.

Since Wald statistics can be applied to type I and III analysis and custom tests, we express Wald statistics in a more general form. The Wald statistic for testing $\mathbf{L}_i\beta = \mathbf{K}$, where $\mathbf{L}_\mathrm{i}$ is a r×p full row rank hypothesis matrix and **K** is a r×1 resulting vector, is defined by

$$S = \left(\mathbf{L}_i\hat{\beta} - \mathbf{K}\right)^{\mathrm{T}} \left(\mathbf{L}_i\mathbf{\Sigma}\mathbf{L}_i{}^{\mathrm{T}}\right)^{-} \left(\mathbf{L}_i\hat{\beta} - \mathbf{K}\right)$$

where $\hat{\beta}$ is the maximum likelihood estimate and $\mathbf{\Sigma}$ is the parameter estimates covariance matrix. S has an asymptotic chi-square distribution with r_C degrees of freedom, where $r_C = rank\left(\mathrm{L\Sigma L}^{\mathrm{T}}\right)$. If $r_C < r$, then $\left(\mathrm{L\Sigma L}^{\mathrm{T}}\right)^{-}$ is a generalized inverse such that Wald tests are effective for a restricted set of hypotheses $\mathbf{L}_{iC}\beta - \mathbf{K}_C$ containing a particular subset C of independent rows from H_0.

For type I and III analysis, calculate the Wald statistic for each effect i according to the corresponding hypothesis matrix $\mathbf{L}_\mathrm{i}$ and **K**=**0**.

Type III Analysis

Wald statistics. See the discussion of Wald statistics for Type I analysis above. L_i is the type III test matrix for the ith effect.

Blank handling

All records with missing values for any input or output field are excluded from the estimation of the model.

Scoring

Scoring is defined as assigning one or more values to a case in a data set.

Predicted Values

Due to the non-linear link functions, the predicted values will be computed for the linear predictor and the mean of the response separately. Also, since estimated standard errors of predicted values of linear predictor are calculated, the confidence intervals for the mean are obtained easily.

Predicted values are still computed as long all the predictor variables have non-missing values in the given model.

Predicted Values of the Linear Predictors

$$\hat{\eta}_i = x_i^{\mathrm{T}}\hat{\beta} + \mathrm{o}_i$$

Estimated Standard Errors of Predicted Values of the Linear Predictors

$$\hat{\sigma}_\eta = \sqrt{x_i^{\mathrm{T}}\Sigma x_i}$$

Predicted Values of the Means

$\hat{\mu}_i = g^{-1}\left(x_i^{\mathrm{T}}\hat{\beta} + o_i\right)$

where g^{-1} is the inverse of the link function. For binomial response with 0/1 binary response variable, this the predicted probability of category 1.

Confidence Intervals for the Means

Approximate 100(1−α)% confidence intervals for the mean can be computed as follows

$g^{-1}\left(x_i^{\mathrm{T}}\hat{\beta} + \mathrm{o}_i \pm z_{1-\alpha/2}\hat{\sigma}_\eta\right)$

If either endpoint in the argument is outside the valid range for he inverse link function, the corresponding confidence interval endpoint is set to a system missing value.

Blank handling

Records with missing values for any input field in the final model cannot be scored, and are assigned a predicted value of $null$.

References

Aitkin, M., D. Anderson, B. Francis, and J. Hinde. 1989. *Statistical Modelling in GLIM*. Oxford: Oxford Science Publications.

Albert, A., and J. A. Anderson. 1984. On the Existence of Maximum Likelihood Estimates in Logistic Regression Models. *Biometrika*, 71, 1–10.

Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Diggle, P. J., P. Heagerty, K. Y. Liang, and S. L. Zeger. 2002. *The analysis of Longitudinal Data*, 2 ed. Oxford: Oxford University Press.

Dobson, A. J. 2002. *An Introduction to Generalized Linear Models*, 2 ed. Boca Raton, FL: Chapman & Hall/CRC.

Dunn, P. K., and G. K. Smyth. 2005. Series Evaluation of Tweedie Exponential Dispersion Model Densities. *Statistics and Computing*, 15, 267–280.

Dunn, P. K., and G. K. Smyth. 2001. Tweedie Family Densities: Methods of Evaluation. In: *Proceedings of the 16th International Workshop on Statistical Modelling,* Odense, Denmark: .

Gill, J. 2000. *Generalized Linear Models: A Unified Approach*. Thousand Oaks, CA: Sage Publications.

Hardin, J. W., and J. M. Hilbe. 2001. *Generalized Estimating Equations*. Boca Raton, FL: Chapman & Hall/CRC.

Hardin, J. W., and J. M. Hilbe. 2003. *Generalized Linear Models and Extension*. Station, TX: Stata Press.

Horton, N. J., and S. R. Lipsitz. 1999. Review of Software to Fit Generalized Estimating Equation Regression Models. *The American Statistician*, 53, 160–169.

Huber, P. J. 1967. The Behavior of Maximum Likelihood Estimates under Nonstandard Conditions. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* Berkeley, CA: University of California Press, 221–233.

Lane, P. W., and J. A. Nelder. 1982. Analysis of Covariance and Standardization as Instances of Prediction. *Biometrics*, 38, 613–621.

Lawless, J. E. 1984. Negative Binomial and Mixed Poisson Regression. *The Canadian Journal of Statistics*, 15, 209–225.

Liang, K. Y., and S. L. Zeger. 1986. Longitudinal Data Analysis Using Generalized Linear Models. *Biometrika*, 73, 13–22.

Lipsitz, S. H., K. Kim, and L. Zhao. 1994. Analysis of Repeated Categorical Data Using Generalized Estimating Equations. *Statistics in Medicine*, 13, 1149–1163.

McCullagh, P. 1983. Quasi-Likelihood Functions. *Annals of Statistics*, 11, 59–67.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Miller, M. E., C. S. Davis, and J. R. Landis. 1993. The Analysis of Longitudinal Polytomous Data: Generalized Estimating Equations and Connections with Weighted Least Squares. *Biometrics*, 49, 1033–1044.

Nelder, J. A., and R. W. M. Wedderburn. 1972. Generalized Linear Models. *Journal of the Royal Statistical Society Series A*, 135, 370–384.

Pan, W. 2001. Akaike's Information Criterion in Generalized Estimating Equations. *Biometrics*, 57, 120–125.

Pregibon, D. 1981. Logistic Regression Diagnostics. *Annals of Statistics*, 9, 705–724.

Smyth, G. K., and B. Jorgensen. 2002. Fitting Tweedie's Compound Poisson Model to Insurance Claims Data: Dispersion Modelling. *ASTIN Bulletin*, 32, 143–157.

White, H. 1980. A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica*, 48, 817–836.

Williams, D. A. 1987. Generalized Linear Models Diagnostics Using the Deviance and Single Case Deletions. *Applied Statistics*, 36, 181–191.

Zeger, S. L., and K. Y. Liang. 1986. Longitudinal Data Analysis for Discrete and Continuous Outcomes. *Biometrics*, 42, 121–130.

Generalized linear mixed models algorithms

Generalized linear mixed models extend the linear model so that:

- The target is linearly related to the factors and covariates via a specified link function.
- The target can have a non-normal distribution.
- The observations can be correlated.

Generalized linear mixed models cover a wide variety of models, from simple linear regression to complex multilevel models for non-normal longitudinal data.

Notation

The following notation is used throughout this chapter unless otherwise stated:

n	Number of complete cases in the dataset. It is an integer and $n \geq 1$.
p	Number of parameters (including the constant, if it exists) in the model. It is an integer and $p \geq 1$.
p_X	Number of non-redundant columns in the design matrix of fixed effects. It is an integer and $p_X \geq 1$.
K	Number of random effects.
$\mathbf{y}$	$n\times$ 1 target vector. The rows are records.
$\mathbf{r}$	$n\times$ 1 events vector for the binomial distribution representing the number of "successes" within a number of trials. All elements are non-negative integers.
$\mathbf{m}$	$n\times$ 1 trials vector for the binomial distribution. All elements are positive integers and $m_i \geq r_i$, i=1,...,n.
$\boldsymbol{\mu}$	$n\times$ 1 expected target value vector.
$\boldsymbol{\eta}$	$n\times$ 1 linear predictor vector.
$\mathbf{X}$	$n\times$ p design matrix. The rows represent the records and the columns represent the parameters. The ith row is $x_i^T = (x_{i1}, \ldots, x_{ip})$, where the superscript T means transpose of a matrix or vector, $i = 1, \ldots, n$ with $x_{i1} = 1$ if the model has an intercept.
$\mathbf{Z}$	$n\times$ r design matrix of random effects.
$\mathbf{O}$	$n\times$ 1 offset vector. This can't be the target or one of the predictors. Also this can't be a categorical field.
$\boldsymbol{\beta}$	$p\times$ 1 parameter vector. The first element is the intercept, if there is one.
$\boldsymbol{\gamma}$	$r\times$ 1 random effect vector.
$\boldsymbol{\omega}$	$n\times$ 1 scale weight vector. If an element is less than or equal to 0 or missing, the corresponding record is not used.
$\mathbf{f}$	$n\times$ 1 frequency weight vector. Non-integer elements are treated by rounding the value to the nearest integer. For values less than 0.5 or missing, the corresponding records are not used.
N	Effective sample size, $N = \sum_{i=1}^{n} f_i$.. If frequency weights are not used, $N = n$.
θ_k	covariance parameters of the kth random effect
θ_G	covariance parameters of the random effects, $\theta_G = \left[\theta_1^T, \ldots, \theta_K^T\right]^T$

θ_R	covariance parameters of the residuals
θ	$\theta = \left[\theta_G^T, \theta_R^T\right]^T = \left[\theta_1^T, \ldots, \theta_K^T, \theta_R^T\right]^T$
$V_{Y\vert\gamma}$	Covariance matrix of **y**, conditional on the random effects

Model

The form of a generalized linear mixed model for the target **y** with the random effects **γ** is

$$\eta = g(\mathrm{E}(y|\gamma)) = \mathbf{X}\beta + \mathbf{Z}\gamma + \mathrm{O}, y|\gamma \sim F,$$

where η is the linear predictor; *g*(.) is the monotonic differentiable link function; **γ** is a (*r*× 1) vector of random effects which are assumed to be normally distributed with mean 0 and variance matrix **G**, **X** is a (*n*× *p*) design matrix for the fixed effects; **Z** is a (*n*× *r*) design matrix for the random effects; **O** is an offset with a constant coefficient of 1 for each observation; *F* is the conditional target probability distribution. Note that if there are no random effects, the model reduces to a generalized linear model (GZLM).

The probability distributions without random effects offered (except multinomial) are listed in Table 19-1 on p. 188. The link functions offered are listed in Table 19-3 on p. 189. Different combinations of probability distribution and link function can result in different models.

See "Nominal multinomial distribution " on p. 206 for more information on the nominal multinomial distribution.

See "Ordinal multinomial distribution " on p. 213 for more information on the ordinal multinomial distribution.

Note that the available distributions depend on the measurement level of the target:

- A continuous target can have any distribution except multinomial. The binomial distribution is allowed because the target could be an "events" field. The default distribution for a continuous target is the normal distribution.
- A nominal target can have the multinomial or binomial distribution. The default is multinomial.
- An ordinal target can have the multinomial or binomial distribution. The default is multinomial.

Table 19-1
Distribution, range and variance of the response, variance function, and its first derivative

Distribution	**Range of *y***	***V*(μ)**	**Var(*y*)**	***V'*(μ)**
Normal	(−∞,∞)	1	ϕ	0
Inverse Gaussian	(0,∞)	μ^3	$\phi\mu^3$	$3\mu^2$
Gamma	(0,∞)	μ^2	$\phi\mu^2$	2μ
Negative binomial	0(1)∞	$\mu+k\mu^2$	$\mu+k\mu^2$	$1+2k\mu$
Poisson	0(1)∞	μ	μ	1
Binomial(*m*)	0(1)*m*/*m*	$\mu(1-\mu)$	$\mu(1-\mu)/m$	$1-2\mu$

Notes

- 0(1)*z* means the range is from 0 to *z* with increments of 1; that is, 0, 1, 2, …, *z*.
- For the binomial distribution, the binomial trial variable *m* is considered as a part of the weight variable ω.
- If a scale weight variable ω is presented, ϕ is replaced by ϕ/ω.
- For the negative binomial distribution, the ancillary parameter (*k*) is estimated by the maximum likelihood (ML) method. When *k* = 0, the negative binomial distribution reduces to the Poisson distribution. When *k* = 1, the negative binomial is the geometric distribution.

The full log-likelihood function (ℓ), which will be used as the objective function for parameter estimation, is listed for each distribution in the following table.

Table 19-2
The log-likelihood function for probability distribution

Distribution	ℓ
Normal	$\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
Inverse Gaussian	$\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
Gamma	$\ell = \ell_k + \sum_{i=1}^{n} f_i\{-\ln(y_i)\}$
Negative binomial	$\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\{-\ln(\Gamma(y_i+1))\}$
Poisson	$\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\{-\ln(y_i!)\}$
Binomial(*m*)	$\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\left\{\ln\begin{pmatrix} m_i \\ r_i \end{pmatrix}\right\}$,where $\begin{pmatrix} m_i \\ r_i \end{pmatrix} = \frac{m_i!}{r_i!(m_i-r_i)!}$

The following tables list the form, inverse form, range of $\hat{\mu}$, and first and second derivatives for each link function.

Table 19-3
Link function name, form, inverse of link function, and range of the predicted mean

Link function	**$\eta=g(\mu)$**	**Inverse $\mu=g^{-1}(\eta)$**	**Range of $\hat{\mu}$**
Identity	μ	η	$\hat{\mu} \in R$
Log	$\ln(\mu)$	$\exp(\eta)$	$\hat{\mu} \geq 0$
Logit	$\ln\left(\frac{\mu}{1-\mu}\right)$	$\frac{\exp(\eta)}{1+\exp(\eta)}$	$\hat{\mu} \in [0,1]$
Probit	$\Phi^{-1}(\mu)$, where $\Phi(\xi) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\xi} e^{-z^2/2}dz$	$\Phi(\eta)$	$\hat{\mu} \in [0,1]$
Complementary log-log	$\ln(-(\ln(1-\mu))$	$1-\exp(-\exp(\eta))$	$\hat{\mu} \in [0,1]$

Link function	**η=*g*(μ)**	**Inverse μ=*g*$^{-1}$(η)**	**Range of $\hat{\mu}$**
Power(α)$\begin{cases} \alpha \neq 0 \\ \alpha = 0 \end{cases}$	$\begin{cases} \mu^{\alpha} \\ \ln(\mu) \end{cases}$	$\begin{cases} \eta^{1/\alpha} \\ \exp(\eta) \end{cases}$	$\begin{cases} \hat{\mu} \in R \text{ if } \alpha \text{ or} 1/\alpha \text{ is odd integer} \\ \hat{\mu} \geq 0 \text{ otherwise} \end{cases}$
Log-complement	ln(1−μ)	1−exp(η)	$\hat{\mu} \leq 1$
Negative log-log	−ln(−ln(μ))	exp(−exp(−η))	$\hat{\mu} \in [0, 1]$

Note: In the power link function, if |α| < 2.2e-16, α is treated as 0.

Table 19-4
The first and second derivatives of link function

Link function	**First derivative** $g'(\mu) = \frac{\partial \eta}{\partial \mu} = \Delta$	**Second derivative** $g''(\mu) = \frac{\partial^2 \eta}{\partial \mu^2}$
Identity	1	0
Log	$\frac{1}{\mu}$	$-\Delta^2$
Logit	$\frac{1}{\mu(1-\mu)}$	$\Delta^2(2\mu - 1)$
Probit	$\frac{1}{\phi\left(\Phi^{-1}(\mu)\right)}$, where $\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$	$\Delta^2 \Phi^{-1}(\mu)$
Complementary log-log	$\frac{1}{(\mu-1)\ln(1-\mu)}$	$-\Delta^2(1 + \ln(1 - \mu))$
Power(α)$\begin{cases} \alpha \neq 0 \\ \alpha = 0 \end{cases}$	$\begin{cases} \alpha \mu^{\alpha - 1} \\ \frac{1}{\mu} \end{cases}$	$\begin{cases} \Delta \frac{\alpha - 1}{\mu} \\ -\Delta^2 \end{cases}$
Log-complement	$\frac{-1}{1-\mu}$	$-\Delta^2$
Negative log-log	$\frac{-1}{\mu \ln(\mu)}$	$\Delta^2(1 + \ln(\mu))$

When the canonical parameter is equal to the linear predictor, $\theta = \eta$, then the link function is called the **canonical link function**. Although the canonical links lead to desirable statistical properties of the model, particularly in small samples, there is in general no a priori reason why the systematic effects in a model should be additive on the scale given by that link. The canonical link functions for probability distributions are given in the following table.

Table 19-5
Canonical and default link functions for probability distributions

Distribution	**Canonical link function**
Normal	Identity
Inverse Gaussian	Power(−2)
Gamma	Power(−1)
Negative binomial	Negative binomial
Poisson	Log
Binomial	Logit

The variance of **y**, conditional on the random effects, is

$$var\,(\mathrm{y}|\gamma) = \mathrm{A}^{1/2}\mathrm{R}\mathrm{A}^{1/2}$$

The matrix **A** is a diagonal matrix and contains the variance function of the model, which is the function of the mean **μ**, divided by the corresponding scale weight variable; that is, A $= \text{diag}(V(\mu_i)/\omega_i), i = 1, \ldots, n.$ The variance functions, *V*(μ), are different for different distributions. The matrix **R** is the variance matrix for repeated measures.

Generalized linear mixed models allow correlation and/or heterogeneity from random effects (**G**-side) and/or heterogeneity from residual effects (**R**-side). resulting in 4 types of models:

1. If a GLMM has no **G**-side or **R**-side effects, then it reduces to a GZLM; **G**=**0** and R $= \phi$I, where **I** is the identity matrix and ϕ is the scale parameter. For continuous distributions (normal, inverse Gauss and gamma), ϕ is an unknown parameter and is estimated jointly with the regression parameters by the maximum likelihood (ML) method. For discrete distributions (negative binomial, Poisson, binomial and multinomial), ϕ is estimated by Pearson chi-square as follows:

   $$\hat{\phi} = \frac{1}{N^*}\sum_{i=1}^{n} f_i\omega_i \frac{(y_i - \mu_i)^2}{V(\mu_i)},$$

   where $N^* = N - p_x$ for the restricted maximum pseudo-likelihood (REPL) method.

2. If a model only has **G**-side random effects, then the **G** matrix is user-specified and R $= \phi$I. ϕ is estimated jointly with the covariance parameters in **G** for continuous distributions and $\phi = 1$ for discrete distributions..

3. If a model only has **R**-side residual effects, then **G** = **0** and the **R** matrix is user-specified. All covariance parameters in **R** are estimated using the REPL method, defined in “Estimation ” on p. 192.

4. If a model has both **G**-side and **R**-side effects, all covariance parameters in **G** and **R** are jointly estimated using the REPL method.

For the negative binomial distribution, there is the ancillary parameter *k*, which is first estimated by the ML method, ignoring random and residual effects, then fixed to that estimate while other regression and covariance parameters are estimated.

Fixed effects transformation

To improve numerical stability, the **X** matrix is transformed according to the following rules.

The *i*th row of **X** is x$_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}}$, *i*=1,...,*n* with $x_{i1} = 1$ if the model has an intercept. Suppose x$_i^*$ is the transformation of x$_i$ then the *j*th entry of x$_i^*$ is defined as

$$\mathrm{x}_{ij}^* = \frac{x_{ij} - c_j}{s_j}$$

where c_{j} and s_{j} are centering and scaling values for x_{ij}, respectively, for j=1,...,p and choices of c_{j} and s_{j} , are listed as follows:

- For a non-constant continuous predictor or a derived predictor which includes a continuous predictor, if the model has an intercept, $c_1 = 0$ and $c_j = \overline{x}_j, j \neq 1,$ where $\overline{x}_j$ is the sample mean of the jth predictor, $\overline{x}_j = \frac{1}{N}\sum\limits_{i=1}^{n} f_i x_{ij}$ and $s_1 = 1$ and $s_j = \sqrt{s_{x_j}^2}, j \neq 1$, where $\sqrt{s_{x_j}^2}$ is the sample standard deviation of the jth predictor and $s_{x_j}^2 = \frac{1}{N-1}\sum\limits_{i=1}^{n} f_i (x_{ij} - \overline{x}_j)^2$. Note that the intercept column is not transformed. If the model has no intercept, $c_j = 0$ and $s_j = \sqrt{s_{x_j}^2 + \overline{x}_j^2}$.
- For a constant predictor $x_{ij} = a \neq 0, \forall i$, $c_j = 0$ and $s_j = a$, that is, scale it to 1.
- For a dummy predictor that is derived from a factor or a factor interaction, $c_j = 0$and$s_j = 1$; that is, leave it unchanged.

Estimation

We estimate GLMMs using linearization-based methods, also called the pseudo likelihood approach (PL; Wolfinger and O'Connell (1994)), penalized quasi-likelihood (PQL; Breslow and Clayton (1993)), marginal quasi-likelihood (MQL; Goldstein (1991)). They are based on the similar principle that the GLMMs are approximated by an LMM so that well-established estimation methods for LMMs can be applied. More specifically, the mean target function; that is, the inverse link function is approximated by a linear Taylor series expansion around the current estimates of the fixed-effect regression coefficients and different solutions of random effects (0 is used for MQL and the empirical Bayes estimates are used for PQL). Applying this linear approximation of the mean target leads to a linear mixed model for a transformation of the original target. The parameters of this LMM can be estimated by Newton-Raphson or Fisher scoring technique and the estimates then are used to update the linear approximation. The algorithm iterates between two steps until convergence. In general, the method is a doubly iterative process. The outer iterations are to update the transformed target for an LMM and the inner iterations are to estimate parameters of the LMM.

It is well known that parameter estimation for an LMM can be based on maximum likelihood (ML) or restricted (or residual) maximum likelihood (REML). Similarly, parameter estimation for a GLMM in the inner iterations can based on maximum pseudo-likelihood (PL) or restricted maximum pseudo-likelihood (REPL).

Linear mixed pseudo model

Following Wolfinger and O'Connell (1993), a first-order Taylor series of $\boldsymbol{\mu}$ in (1) about $\tilde{\beta}$ and $\tilde{\gamma}$ yields

$$\mu \approx \tilde{\mu} + \left(g^{-1}\right)' \left(\mathrm{X}\tilde{\beta} + \mathrm{Z}\tilde{\gamma} + \mathrm{O}\right) \left[\mathrm{X}\left(\beta - \tilde{\beta}\right) + \mathrm{Z}(\gamma - \tilde{\gamma})\right]$$

where $\left(g^{-1}\right)'\left(\mathbf{X}\tilde{\beta}+\mathrm{Z}\tilde{\gamma}+\mathrm{O}\right)$ is a diagonal matrix with elements consisting of evaluations of the 1st derivative of g^{-1}. Since $\left(g^{-1}\right)'\left(\mathbf{X}\tilde{\beta}+\mathrm{Z}\tilde{\gamma}+\mathrm{O}\right)=\left(g'(\tilde{\mu})\right)^{-1}$, this equation can be rearranged as

$$g'(\tilde{\mu})(\mu-\tilde{\mu})+\mathbf{X}\tilde{\beta}+\mathrm{Z}\tilde{\gamma}\approx\mathbf{X}\beta+\mathrm{Z}\gamma$$

If we define a pseudo target variable as

$$\mathrm{v}\equiv g'(\tilde{\mu})(\mathrm{y}-\tilde{\mu})+\mathbf{X}\tilde{\beta}+\mathrm{Z}\tilde{\gamma}=g'(\tilde{\mu})(\mathrm{y}-\tilde{\mu})+g(\tilde{\mu})-\mathrm{O},$$

then the conditional expectation and variance of v, based on $\mathrm{E}\,(\mathrm{y}|\gamma)$ and $var\,(\mathrm{y}|\gamma)=\mathrm{A}^{1/2}\mathrm{R}\mathrm{A}^{1/2}$, are

$$\mathrm{E}(\mathrm{v}|\gamma)=g'(\tilde{\mu})(\mu-\tilde{\mu})+\mathbf{X}\tilde{\beta}+\mathrm{Z}\tilde{\gamma}$$

$$var(\mathrm{v}|\gamma)=g'(\tilde{\mu})\mathrm{A}_{\tilde{\mu}}^{1/2}\mathrm{R}\mathrm{A}_{\tilde{\mu}}^{1/2}g'(\tilde{\mu})$$

where $\mathrm{A}_{\tilde{\mu}}^{1/2}=\mathrm{diag}\left[\left(V(\tilde{\mu}_i)/\omega_i\right)^{1/2}\right],i=1,\ldots,n.$

Furthermore, we also assume $\mathrm{v}|\gamma$ is normally distributed. Then we consider the model of v

$$\mathrm{v}=\mathbf{X}\beta+\mathrm{Z}\gamma+\varepsilon$$

as a weighted linear mixed model with fixed effects β, random effects $\gamma\sim N(0,\mathrm{G})$, error terms $\varepsilon\sim N\left(0,g'(\tilde{\mu})\mathrm{A}_{\tilde{\mu}}^{1/2}\mathrm{R}\mathrm{A}_{\tilde{\mu}}^{1/2}g'(\tilde{\mu})\right)$, because $var(\varepsilon)=var(\mathrm{v}|\gamma)$, and diagonal weight matrix $\tilde{W}=\mathrm{A}_{\tilde{\mu}}\left[g'(\tilde{\mu})\right]^{-2}$. Note that the new target v (with **O** if an offset variable exists) is a Taylor series approximation of the linked target $g(\mathrm{y})$. The estimation method of unknown parameters of β and θ, which contains all unknowns in **G** and **R**, for traditional linear mixed models can be applied to this linear mixed pseudo model.

The Gaussian log pseudo-likelihood (PL) and restricted log pseudo-likelihood (REPL), which are expressed as the functions of covariance parameters in θ, corresponding to the linear mixed model for v are the following:

$$\ell(\theta;\mathrm{v})=-\frac{1}{2}\ln|\mathrm{V}(\theta)|-\frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta)-\frac{N}{2}\ln(2\pi)$$

$$\ell_R(\theta;\mathrm{v})=-\frac{1}{2}\ln|\mathrm{V}(\theta)|-\frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta)-\frac{1}{2}\ln\left|\mathrm{X}^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}\right|-\frac{N-p_x}{2}\ln(2\pi)$$

where
$\mathrm{V}(\theta)=\mathrm{Z}\mathrm{G}(\theta)\mathrm{Z}+\tilde{\mathbf{W}}^{-1/2}\mathrm{R}(\theta)\tilde{\mathbf{W}}^{-1/2}$,$\mathrm{r}(\theta)=\mathrm{v}-\mathrm{X}\left(\mathrm{X}^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}\right)^{-}\mathrm{X}^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{v}=\mathrm{v}-\mathrm{X}\hat{\beta}$,$N$ denotes the effective sample size, and p_{X} denotes the rank of the design matrix of **X** or the number of non-redundant parameters in **X**. Note that the regression parameters in β are profiled from the above equations because the estimation of β can be obtained analytically. The covariance

parameters in θ are estimated by Newton-Raphson or Fisher scoring algorithm. Following the tradition in linear mixed models, the objection functions of minimization for estimating θ would be $-2\ell(\theta;\mathrm{v})$ or $-2\ell_R(\theta;\mathrm{v})$. Upon obtaining $\hat{\theta}$, estimates for β and γ are computed as

$$\hat{\beta} = \left(\mathrm{X}^{\mathrm{T}}\mathrm{V}\left(\hat{\theta}\right)^{-1}\mathrm{X}\right)^{-}\mathrm{X}^{\mathrm{T}}\mathrm{V}\left(\hat{\theta}\right)^{-1}\mathrm{v}$$

$$\hat{\gamma} = \hat{G}\mathrm{Z}^{\mathrm{T}}\mathrm{V}\left(\hat{\theta}\right)^{-1}\hat{r}$$

where $\hat{\beta}$ is the best linear unbiased estimator (BLUE) of β and $\hat{\gamma}$ is the estimated best linear unbiased predictor (BLUP) of γ in the linear mixed pseudo model. With these statistics, v and $\tilde{W}$ are recomputed based on $\tilde{\mu}$ and the objective function is minimized again to obtain updated $\hat{\theta}$. Iteration between $-2\ell(\theta;\mathrm{v})$ and the above equation yields the PL estimation procedure and between $-2\ell_R(\theta;\mathrm{v})$ and the above equation the REPL procedure.

There are two choices for $\tilde{\gamma}$ (the current estimates of γ):

1. $\hat{\gamma}$ for PQL; and
2. 0 for MQL.

On the other hand, $\hat{\beta}$ is always used as the current estimate of the fixed effects. Based on the two objective functions (PL or REPL) and two choices of random effect estimates (PQL or MQL), 4 estimation methods can be implemented for GLMMs:

1. PL-PQL: pseudo-likelihood with $\tilde{\gamma}$=$\hat{\gamma}$;
2. PL-MQL: pseudo-likelihood with $\tilde{\gamma}$=0;
3. REPL-PQL: residual pseudo-likelihood with $\tilde{\gamma}$=$\hat{\gamma}$;
4. REPL-MQL: residual pseudo-likelihood with $\tilde{\gamma}$=0.

We use method 3, REPL-PQL.

Iterative process

The doubly iterative process for the estimation of θ is as follows:

1. Obtain an initial estimate of μ, $\mu^{(0)}$. Specifically, $\mu_i^0 = (y_i m_i + 0.5)/(m_i + 1)$ for a binomial distribution (y_i can be a proportion or 0/1 value) and $\mu_i^0 = y_i$ for a non-binomial distribution. Also set the outer iteration index *j* = 0.
2. Based on $\tilde{\mu}$, compute

   $\mathrm{v} = g(\tilde{\mu}) - \mathrm{O} + g'(\tilde{\mu})(\mathrm{y} - \tilde{\mu})$ and $\tilde{W} = \mathrm{A}_{\tilde{\mu}}^{-1}\left[g'(\tilde{\mu})\right]^{-2}$.

   Fit a weighted linear mixed model with pseudo target v, fixed effects design matrix **X**, random effects design matrix **Z**, and diagonal weight matrix $\tilde{W}$. The fitting procedure, which is called the inner iteration, yields the estimates of θ, and is denoted as $\theta^{(j)}$. The procedure uses the specified settings for parameter, log-likelihood, and Hessian convergence criteria for determining convergence of the linear mixed model. If j = 0, go to step 4; otherwise go to the next step.

3. Check if the following criterion with tolerance level ξ is satisfied:

   $\max_i \left(2 \times \frac{\left|\theta_i^{(j)} - \theta_i^{(j-1)}\right|}{\left|\theta_i^{(j-1)}\right| + \left|\theta_i^{(j-1)}\right|} \right) < \xi.$

   If it is met or maximum number of outer iterations is reached, stop. Otherwise, go to the next step.

4. Compute $\hat{\beta}$ by setting $\hat{\theta} = \theta^{(j)}$ then set $\tilde{\beta} = \hat{\beta}$. Depending on the choice of random effect estimates, set $\tilde{\gamma}$=$\hat{\gamma}$.

5. Compute the new estimate of μ by

   $\tilde{\mu} = g^{-1}\left(\mathbf{X}\tilde{\beta} + \mathrm{Z}\tilde{\gamma} + \mathrm{O}\right),$

   set $j = j + 1$ and go to step 2.

Wald confidence intervals for covariance parameter estimates

Here we assume that the estimated parameters of **G** and **R** are obtained through the above doubly iterative process. Then their asymptotic covariance matrix can be approximated by $2\mathrm{H}^{-1}$, where **H** is the Hessian matrix of the objective function ($-2\ell(\theta; \mathrm{v})$ or $-2\ell_R(\theta; \mathrm{v})$) evaluated at $\hat{\theta}$. The standard error for the ith covariance parameter estimate in the $\hat{\theta}$ vector, say $\hat{\theta}_i$, is the square root of the ith diagonal element of $2\mathrm{H}^{-1}$.

Thus, a simple Wald's type confidence interval or test statistic for any covariance parameter can be obtained by using the asymptotic normality. However, these can be unreliable in small samples, especially for variance and correlation parameters that have a range of $[0, \infty)$ and $[-1, 1]$ respectively. Therefore, following the same method used in linear mixed models, these parameters are transformed to parameters that have range $(-\infty, \infty)$. Using the delta method, these transformed estimates still have asymptotic normal distributions.

For variance type parameters in **G** and **R**, such as σ^2 in the autoregressive, autoregressive moving average, compound symmetry, diagonal, Toeplitz, and variance components, and θ_{ii} in the unstructured type, the 100(1 – α)% Wald confidence interval is given, assuming the variance parameter estimate is $\hat{\sigma}^2$ and its standard error is $\mathrm{se}\left(\hat{\sigma}^2\right)$ from the corresponding diagonal element of $2\mathrm{H}^{-1}$, by

$$\exp\left(\ln\left(\hat{\sigma}^2\right) \pm z_{1-\alpha/2} \cdot \hat{\sigma}^{-2} \cdot \mathrm{se}\left(\hat{\sigma}^2\right)\right)$$

For correlation type parameters in **G** and **R**, such as ρ in the autoregressive, autoregressive moving average, and Toeplitz types and φ in the autoregressive moving average type, which usually come with the constraint of $|\rho| \leq 1$, the 100(1 – α)% Wald confidence interval is given, assuming the correlation parameter estimate is $\hat{\rho}$ and its standard error is $\mathrm{se}(\hat{\rho})$ from the corresponding diagonal element of $2\mathrm{H}^{-1}$, by

$$\tanh\left(\tanh^{-1}(\hat{\rho}) \pm z_{1-\alpha/2} \cdot \left(1 - \hat{\rho}^2\right)^{-1} \cdot \mathrm{se}(\hat{\rho})\right)$$

where $\tanh x = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$ and $\tanh^{-1}x = \frac{1}{2}\ln\left[\frac{1+x}{1-x}\right]$ are hyperbolic tangent and inverse hyperbolic tangent, respectively.

For general type parameters, other than variance and correlation types, in **G** and **R**, such as σ_1 in the compound symmetry type and $\theta_{ij}, i \neq j$, (off-diagonal elements) in the unstructured type, no transformation is done. Then the 100(1 – α)% Wald confidence interval is simply, assuming the parameter estimate is $\hat{\sigma}_1$ and its standard error is se$(\hat{\sigma}_1)$ from the corresponding diagonal element of $2\mathrm{H}^{-1}$,

$$\left(\hat{\sigma}_1 - z_{1-\alpha/2} \cdot \mathrm{se}(\hat{\sigma}_1)\right), \hat{\sigma}_1 + z_{1-\alpha/2} \cdot \mathrm{se}(\hat{\sigma}_1)\right)$$

The 100(1 – α)% Wald confidence interval for ϕ is

$$\left(\exp\left(\hat{\tau} - z_{1-\alpha/2}\hat{\sigma}_\tau\right), \exp\left(\hat{\tau} + z_{1-\alpha/2}\hat{\sigma}_\tau\right)\right)$$

where $\tau = \ln(\phi)$.

Note that the *z*-statistics for the hypothesis $H_{0i} : \theta_i = 0$, where θ_i is a covariance parameter in θ vector, are calculated; however, the Wald tests should be considered as an approximation and used with caution because the test statistics might not have a standardized normal distribution.

Statistics for estimates of fixed and random effects

The approximate covariance matrix of $\left(\hat{\beta} - \beta, \hat{\gamma} - \gamma\right)$ is

$$\hat{C} = \begin{bmatrix} \mathrm{X}^\mathrm{T}\mathrm{R}^{*-1}\mathrm{X} & \mathrm{X}^\mathrm{T}\mathrm{R}^{*-1}\mathrm{Z} \\ \mathrm{Z}^\mathrm{T}\mathrm{R}^{*-1}\mathrm{X} & \mathrm{Z}^\mathrm{T}\mathrm{R}^{*-1}\mathrm{Z} + \mathrm{G}\left(\hat{\theta}\right)^{-1} \end{bmatrix}^{-} = \begin{bmatrix} \mathrm{C}_{11} & \mathrm{C}_{21}^\mathrm{T} \\ \mathrm{C}_{21} & \mathrm{C}_{22} \end{bmatrix}$$

where $\mathrm{R}^* \equiv v\hat{a}r(\mathrm{v}|\gamma) = g'(\hat{\mu})\mathrm{A}_{\hat{\mu}}^{1/2}\mathrm{R}\mathrm{A}_{\hat{\mu}}^{1/2}g'(\hat{\mu})$ is evaluated at the converged estimates and

$$\hat{C}_{11} = \left(\mathbf{X}^\mathrm{T}\hat{V}^{-1}\mathbf{X}\right)^{-}$$

$$\hat{C}_{21} = -\hat{G}\mathrm{Z}^\mathrm{T}\hat{V}^{-1}\mathbf{X}\hat{C}_{11}$$

$$\hat{C}_{22} = \left(\mathrm{Z}^\mathrm{T}\hat{R}^{-1}\mathrm{Z} + \hat{G}^{-1}\right)^{-1} - \hat{C}_{21}\mathbf{X}^\mathrm{T}\hat{V}^{-1}\mathrm{Z}\hat{G}$$

Statistics for estimates of fixed effects on original scale

If the **X** matrix is transformed, the restricted log pseudo-likelihood (REPL) would be different based on transformed and original scale, so the REPL on the transformed scale should be transformed back on the final iteration so that any post-estimation statistics based on REPL can be calculated correctly. Suppose the final objective function value based on the transformed and original scales are $-2\ell_R^*(\theta; \mathrm{v})$ and $-2\ell_R(\theta; \mathrm{v})$, respectively, then $-2\ell_R(\theta; \mathrm{v})$ can be obtained from $-2\ell_R^*(\theta; \mathrm{v})$ as follows:

$$-2\ell_R(\theta; \mathrm{v}) = -2\ell_R^*(\theta; \mathrm{v}) - 2\ln|\mathrm{A}|$$

Because REPL has the following extra term involved the X matrix

$$\begin{aligned} -\tfrac{1}{2}\ln\left|\mathrm{X}^{*\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}^*\right| &= -\tfrac{1}{2}\ln\left|(\mathrm{XA})^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{XA}\right| \\ &= -\tfrac{1}{2}\ln\left(\left|\mathrm{A}^{\mathrm{T}}\right| \times \left|\mathrm{XV}(\theta)^{-1}\mathrm{X}\right| \times |\mathrm{A}|\right) \\ &= -\tfrac{1}{2}\left(\ln\left|\mathrm{XV}(\theta)^{-1}\mathrm{X}\right| + \ln|\mathrm{A}| + \ln\left|\mathrm{A}^{\mathrm{T}}\right|\right) \\ &= -\tfrac{1}{2}\ln\left|\mathrm{XV}(\theta)^{-1}\mathrm{X}\right| - \ln|\mathrm{A}| \end{aligned}$$

then $-\frac{1}{2}\ln\left|\mathrm{XV}(\theta)^{-1}\mathrm{X}\right| = -\frac{1}{2}\ln\left|\mathrm{X}^{*\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}^*\right| + \ln|\mathrm{A}|$ and $\ell_R(\theta; \mathrm{v}) = \ell_R^*(\theta; \mathrm{v}) + \ln|\mathrm{A}|$. Please note that PL values are the same whether the X matrix is transformed or not.

In addition, the final estimates of $\boldsymbol{\beta}$, $\mathbf{C}_{11}$, $\mathbf{C}_{21}$ and $\mathbf{C}_{22}$ are based on the transformed scale, denoted as $\hat{\beta}^*$,$\hat{C}_{11}^*$,$\hat{C}_{21}^*$ and $\hat{C}_{22}^*$, respectively. They are transformed back to the original scale, denoted as $\hat{\beta}$,$\hat{C}_{11}$,$\hat{C}_{21}$ and $\hat{C}_{22}$, respectively, as follows:

$$\hat{\beta} = \mathrm{A}\hat{\beta}^*,$$

$$\hat{C}_{11} = A\hat{C}_{11}^* A^{\mathrm{T}},$$

$$\hat{C}_{21} = \hat{C}_{21}^*\mathrm{A}^{\mathrm{T}},$$

$$\hat{C}_{22} = \hat{C}_{22}^*.$$

Note that **A** could reduce to S^{-1}; hereafter, the superscript * denotes a quantity on the transformed scale.

Estimated covariance matrix of the fixed effects parameters

Two estimated covariance matrices of the fixed effects parameters can be calculated: model-based and robust.

The model-based estimated covariance matrix of the fixed effects parameters is given by

$$\Sigma_{\mathrm{m}} = \hat{C}_{11}$$

The robust estimated covariance matrix of the fixed effects parameters for a GLMM is defined as the classical sandwich estimator. It is similar to that for a generalized linear model or a generalized estimating equation (GEE). If the model is a generalized linear mixed model and it is processed by subjects, then the robust estimator is defined as follows

$$\Sigma\text{r}=\Sigma\text{m}\left(\sum_{j=1}^{S}\mathbf{X}_j^{\mathrm{T}}\hat{V}_j^{-1}\hat{r}_j\hat{r}_j^{\mathrm{T}}\hat{V}_j^{-1}\mathbf{X}_j\right)\Sigma\text{m}$$

where $\hat{r}_j = \mathrm{v}_j - \mathrm{X}_j\hat{\beta}$.

Standard errors for estimates in fixed effects and predictions in random effects

Let $\hat{\beta}_i$ denote a non-redundant parameter estimate in fixed effects. Its standard error is the square root of the *i*th diagonal element of Σm or Σr,

$$\hat{\sigma}_{\beta_i} = \sqrt{\sigma_{ii}}$$

The standard error for redundant parameter estimates is set to a system missing value.

Let $\hat{\gamma}_i$ denote a prediction in random effects. Its standard error is the square root of the *i*th diagonal element of $\hat{C}_{22}$:

$$\hat{\sigma}_{\gamma_i} = \sqrt{\hat{C}_{22,ii}}$$

Test statistics for estimates in fixed effects and predictions in random effects

The hypothesis $H_{0i} : \beta_i = 0$ is tested for each non-redundant parameter in fixed effects using the *t* statistic:

$$t_i = \frac{\hat{\beta}_i}{\hat{\sigma}_{\beta_i}}$$

which has an asymptotic *t* distribution with υ degrees of freedom. See "Method for computing degrees of freedom " on p. 203 for details on computing the degrees of freedom.

Wald confidence intervals for estimates in fixed effects and predictions in random effects

The 100(1 – α)% Wald confidence interval for β_i is given by

$$\left(\hat{\beta}_i - t_{\upsilon,\alpha/2}\hat{\sigma}_{\beta_i}, \hat{\beta}_i + t_{\upsilon,\alpha/2}\hat{\sigma}_{\beta_i}\right)$$

where $t_{\upsilon,\alpha/2}$ is the $(1 - \alpha/2)$ 100th percentile of the t_υ distribution.

For some models (see the list below), the exponentiated parameter estimates, their standard errors, and confidence intervals are computed. Using the delta method, the estimate of $\exp(\beta_i)$ is $\exp\left(\hat{\beta}_i\right)$, the standard error estimate is $\left(\exp\left(\hat{\beta}_i\right) \cdot \hat{\sigma}_{\beta_i}\right)$ and the corresponding 100(1 – α)% Wald confidence interval for $\exp(\beta_i)$ is

$$\left(\exp\left(\hat{\beta}_i - t_{\upsilon,\alpha/2}\hat{\sigma}_{\beta_i}\right), \exp\left(\hat{\beta}_i + t_{\upsilon,\alpha/2}\hat{\sigma}_{\beta_i}\right)\right).$$

The list of models is as follows:

1. Logistic regression (binomial distribution + logit link).
2. Nominal logistic regression (nominal multinomial distribution + generalized logit link).
3. Ordinal logistic regression (ordinal multinomial distribution + cumulative logit link).
4. Log-linear model (Poisson distribution + log link).
5. Negative binomial regression (negative binomial distribution + log link).

Testing

After estimating parameters and calculating relevant statistics, several tests for the given model are performed.

Goodness of fit

Information criteria

Information criteria are used when comparing different models for the same data. The formulas for various criteria are as follows.

Finite sample corrected (AICC)	$-2\ell+\frac{2d\cdot N}{(N-d-1)}$
Bayesian information criteria (BIC)	$-2\ell+d\ln(N)$

where ℓ is the restricted log-pseudo-likelihood evaluated at the parameter estimates. For REPL, N is the effective sample size minus the number of non-redundant parameters in fixed effects $(\sum_{i=1}^{n} f_i - p_x)$ and d is the number of covariance parameters.

Note that the restricted log-pseudo-likelihood values are of the linearized model, not on the original scale. Thus the information criteria should not be compared across models with different distribution and link function and they should be interpreted with caution.

Tests of fixed effects

For each effect specified in the model, a type III test matrix **L** is constructed and H_0: $\mathbf{L}_\mathrm{i}\boldsymbol{\beta}=\mathbf{0}$ is tested. Construction of **L** and the generating estimable function (GEF) is based on the generating matrix $\mathrm{H}_\omega=\left(\mathrm{X}^\mathrm{T}\Psi\mathrm{X}\right)^{-}\mathrm{X}^\mathrm{T}\Psi\mathrm{X}$, where $\Psi=\mathrm{diag}(f_1\omega_1,\ldots f_n\omega_n)$, such that $\mathbf{L}_\mathrm{i}\boldsymbol{\beta}$ is estimable; that is, $\mathrm{L}_i=\mathrm{L}_i\mathrm{H}_\omega$. It involves parameters only for the given effect and the effects containing the given effect. For type III analysis, **L** does not depend on the order of effects specified in the model. If such a matrix cannot be constructed, the effect is not testable.

Then the **L** matrix is then used to construct the test statistic

$$\mathbf{F} = \frac{\hat{\beta}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\left(\mathbf{L}\sum\mathbf{L}^{\mathrm{T}}\right)^{-1}\mathbf{L}\hat{\beta}}{r_c}$$

where $r_c = rank\left(\mathbf{L}\sum\mathbf{L}^{\mathrm{T}}\right)$. The statistic has an approximate *F* distribution. The numerator degrees of freedom is r_c and the denominator degrees of freedom is υ. See "Method for computing degrees of freedom " on p. 203 for details on computing the denominator degrees of freedom.

In addition, we test a null hypothesis that all regression parameters (except intercept if there is one) equal zero. The test statistic would be the same as the above F statistic except the L matrix is from GEF. If there is no intercept, the L matrix is the whole GEF. If there is an intercept, the L matrix is GEF without the first row which corresponds to the intercept. This test is similar to the "corrected model" in linear models.

Estimated marginal means

There are two types of estimated marginal means calculated here. One corresponds to the specified factors for the linear predictor of the model and the other corresponds to those for the original scale of the target.

Estimated marginal means are based on the estimated cell means. For a given fixed set of factors, or their interactions, we estimate marginal means as the mean value averaged over all cells generated by the rest of the factors in the model. Covariates may be fixed at any specified value. If not specified, the value for each covariate is set to its overall mean estimate.

Estimated marginal means are not available for the multinomial distribution.

Estimated marginal means for the linear predictor

Calculating estimated marginal means for the linear predictor

Estimated marginal means for the linear predictor are based on the link function transformation, and constructed such that **LB** is estimable.

Suppose there are *r* combined levels of the specified categorical effect. This *r*×1 vector can be expressed in the form $\hat{\mathbf{u}} = \mathbf{L}\hat{\beta}$. The variance matrix of $\hat{\mathbf{u}}$ is then computed by

$$\mathrm{V}(\hat{\mathbf{u}})\mathrm{=L\Sigma L}^{\mathrm{T}}$$

The standard error for the *j*th element of $\hat{\mathbf{u}}$ is the square root of the *j*th diagonal element of $\mathrm{V}(\hat{\mathbf{u}})$. Let the *j*th element of $\hat{\mathbf{u}}$ and its standard error be $\hat{u}_j$ and $\hat{\sigma}_{u_j}$, respectively, then the corresponding 100(1 – α)% confidence interval for $u_j, j = 1, \ldots, r,$ is given by

$$\hat{u}_j \pm t_{\upsilon^j,\alpha/2}\hat{\sigma}_{u_j}$$

where $t_{v^j,\alpha/2}$ is the $(1-\alpha/2)100th$ percentile of the *t* distribution with v^j degrees of freedom. See "Method for computing degrees of freedom " on p. 203 for details on computing the degrees of freedom.

Comparing estimated marginal means for the linear predictor

We can compare estimated marginal means for the linear predictor based on a selected contrast type, for which a set of contrasts for the factor is created. Let this set of contrasts define matrix **C** used for testing the hypothesis $H_0 : \mathrm{C}u = 0$. An F statistic is used for testing given set of contrasts for the factor as follows:

$$F = \frac{(\mathrm{C}\hat{u})^{\mathrm{T}}\left(\mathrm{CV}(\hat{u})\mathrm{C}^{\mathrm{T}}\right)^{-}(\mathrm{C}\hat{u})}{r_I}$$

which has an asymptotic *F* distribution with r_I degrees of freedom, where $r_I = \mathrm{rank}\left(\mathrm{CV}(\hat{\mathbf{u}})\mathrm{C}^{\mathrm{T}}\right)$. See "Method for computing degrees of freedom " on p. 203 for details on computing the denominator degrees of freedom. The *p*-values can be calculated accordingly. Note that adjusted *p*-values based on multiple comparisons adjustments won't be computed for the overall test.

Each row $\mathrm{c}_i^{\mathrm{T}}$ of matrix **C** is also tested separately. The estimate for the *i*th row is given by $\mathrm{c}_i^{\mathrm{T}}\hat{\mathbf{u}}$ and its standard error by $\sqrt{\mathrm{c}_i^{\mathrm{T}}\mathrm{V}(\hat{\mathbf{u}})\mathrm{c}_i}$. The corresponding 100(1 – α)% confidence interval is given by

$$\mathrm{c}_i^{\mathrm{T}}\hat{u} \pm t_{v^i,\alpha/2}\hat{\sigma}_{cu_i}$$

The test statistic for $H_0 : \mathrm{c}_i^{\mathrm{T}}\mathbf{u} = 0$ is

$$t_i = \frac{\mathrm{c}_i^{\mathrm{T}}\hat{u}}{\hat{\sigma}_{cu_i}}$$

It has an asymptotic *t* distribution. See "Method for computing degrees of freedom " on p. 203 for details on computing the degrees of freedom. The *p*-values can be calculated accordingly. In addition, adjusted *p*-values for multiple comparisons can also computed.

Estimated marginal means in the original scale

Estimated marginal means for the target are based on the original scale. As a conditional predictor defined by Lane and Nelder (1982), estimated marginal means for the target are derived from those for the linear predictor.

Calculating estimated marginal means for the target

The estimated marginal means for the target are defined as

$$\hat{\mathbf{M}} = g^{-1}\left(\mathrm{L}\hat{\beta}\right) = g^{-1}(\hat{\mathbf{u}})$$

The variance of estimated marginal means for the target is

$$V\left(\hat{\mathbf{M}}\right) = diag\left(\frac{\partial g^{-1}(\hat{v}_j)}{\partial \hat{u}_j}\right) L\Sigma L^T diag\left(\frac{\partial g^{-1}(\hat{u}_j)}{\partial \hat{v}_j}\right)$$

where $diag\left(\partial g^{-1}(\hat{u}_j)/\partial \hat{u}_j\right)$ is a *r*×*r* matrix and $\partial g^{-1}(\hat{u}_j)/\partial \hat{u}_j$ is the derivative of the inverse of the link with respect to the *j*th value in $\hat{\mathbf{u}}$ and $\partial g^{-1}(\hat{u}_j)/\partial \hat{u}_j = 1/g^{'}\left(\hat{M}_j\right)$ where $g^{'}\left(\hat{M}_j\right)$ is from Table 19-4 on p. 190.

The 100(1 – α)% confidence interval for $M_i, i = 1, \ldots, r$, is given by

$$g^{-1}\left(\hat{u}_i \pm t_{v^i,\alpha/2}\hat{\sigma}_{u_i}\right).$$

Note: $\hat{\mathbf{M}}$ is estimated marginal means for the proportion, not for the number of events when events and trials variables are used for the binomial distribution.

Comparing estimated marginal means for the target

This is similar to comparing estimated marginal means for the linear predictor; just replace $\hat{\mathbf{u}}$ with $\hat{\mathbf{M}}$ and $V(\hat{\mathbf{u}})$ with $V\left(\hat{\mathbf{M}}\right)$. For more information, see the topic “Estimated marginal means for the linear predictor” on p. 200.

Multiple comparisons

The hypothesis $H_0 : \mathbf{Cu} = \mathbf{0}$ can be tested using the multiple row hypotheses testing technique. Let $\mathbf{c}_i^T$ be the *i*th row vector of matrix **C**. The *i*th row hypothesis is $H_{0i} : \mathbf{c}_i^T\mathbf{u} = 0$. Testing H_0 is the same as testing multiple non-redundant row hypotheses $\{H_{0i}^*\}_{i=1}^R$ simultaneously, where *R* is the number of non-redundant row hypotheses, and H_{0i}^* represents the *i*th non-redundant hypothesis. A hypothesis H_{0i} is redundant if there exists another hypothesis $H_{0j},\ j \neq i$ such that $c_i = ac_j,\ a \neq 0$.

Adjusted p-values. For each individual hypothesis H_{0i}, test statistics can be calculated. Let p_i denote the *p*-value for testing H_{0i} and p_i^* denote the adjusted *p*-value. The conclusion from multiple testing is, at level α (the family-wise type I error),

reject $H_{0i} : \mathbf{c}_i^T\mathbf{u} = 0$ if $p_i^* < \alpha$;

reject $H_0 : \mathbf{Cu} = \mathbf{0}$ if $\min_i\left(p_i^*\right) < \alpha$.

Several different methods to adjust *p*-values are provided here. Please note that if the adjusted *p*-value is bigger than 1, it is set to 1 in all the methods.

Adjusted confidence intervals. Note that if confidence intervals are also calculated for the above hypothesis, then adjusting confidence intervals is required to correspond to adjusted *p*-values. The only item needed to be adjusted in the confidence intervals is the critical value from the standard normal distribution. Assume that the original critical value is $z_{1-\alpha/2}$ and the adjusted critical value is z^*.

LSD (Least Significant Difference)

The adjusted p-values are the same as the original p-values:

$$p_i^* = p_i$$

The adjusted critical value is:

$$t^* = t_{v^i,\alpha/2}$$

Sequential Bonferroni

The adjusted p-values are:

$$p_{(i)}^* = \begin{cases} Rp_{(1)} & i = 1 \\ \max\left((R - i + 1)\, p_{(i)}, p_{(i-1)}^* \right) & i \geq 2 \end{cases}$$

The adjusted critical values will correspond to the ordered adjusted p-values as follows:

$$t_{v^{(i)}}^* = \begin{cases} t_{v^{(i)}, \frac{\alpha}{2R}} & \text{if } i = 1 \\ t_{v^{(i)}, \frac{\alpha}{2(R-i+1)}} & \text{if } p_{(i)}^* {=} (R - i + 1) p_{(i)} \text{ for } i \geq 2 \\ t_{v^{(i)}, \frac{\alpha}{2\left(p_{(i-1)}^* / p_{(i)}\right)}} & \text{if } p_{(i)}^* {=} p_{(i-1)}^* \text{ for } i \geq 2 \end{cases}$$

Sequential Sidak

The adjusted p-values are:

$$p_{(i)}^* = \begin{cases} 1 - \left(1 - p_{(1)}\right)^R & i = 1 \\ \max\left(1 - \left(1 - p_{(i)}\right)^{R-i+1}, p_{(i-1)}^* \right) & i \geq 2 \end{cases}$$

The adjusted critical values will correspond to the ordered adjusted p-values as follows:

$$t_{v^{(i)}}^* = \begin{cases} t_{v^{(i)}, \frac{1-(1-\alpha)^{1/R}}{2}} & \text{if } i = 1, \\ t_{v^{(i)}, \frac{1-(1-\alpha)^{1/(R-i+1)}}{2}} & \text{if } p_{(i)}^* = (R - i + 1) p_{(i)} \text{ for } i \geq 2 \\ t_{v^{(i)}, \frac{1-(1-\alpha)^{1/x}}{2}} & \text{if } p_{(i)}^* {=} p_{(i-1)}^* \text{ for } i \geq 2 \end{cases};$$

where $x = \frac{\ln\left(1 - p_{(i-1)}^*\right)}{\ln\left(1 - p_{(i)}\right)}$.

Method for computing degrees of freedom

Residual method

The value of degrees of freedom is given by $N - rank(\mathrm{X})$, where N is the effective sample size and **X** is the design matrix of fixed effects.

Satterthwaite's approximation

First perform the spectral decomposition $\mathbf{L}\hat{\mathbf{C}}\mathbf{L}^T = \mathbf{\Gamma}^T\mathbf{D}\mathbf{\Gamma}$ where $\mathbf{\Gamma}$ is an orthogonal matrix of eigenvectors and $\mathbf{D}$ is a diagonal matrix of eigenvalues. If l_m is the *m*th row of $\mathbf{\Gamma}\mathbf{L}$, d_m is the *m*th eigenvalues and

$$\nu_m = \frac{2d_m}{\mathbf{g_m}\Sigma\left(\hat{\theta}\right)^{-1}\mathbf{g_m}}$$

where $\mathbf{g}_m = \frac{\partial l_m \mathbf{C} l_m^T}{\partial\theta}|_{\theta=\hat{\theta}}$ and $\Sigma_{\hat{\theta}}$ is the asymptotic covariance matrix of $\hat{\theta}$ obtained from the Hessian matrix of the objective function; that is, $\Sigma_{\hat{\theta}} = 2\mathrm{H}^{-1}$. If

$$E = \sum_{m=1}^{q} \frac{\nu_m}{\nu_m - 2} I\left(\nu_m > 2\right)$$

then the denominator degree of freedom is given by

$$\nu = \frac{2E}{E-q}$$

Note that the degrees of freedom can only be computed when *E*>*q*.

Scoring

For GLMMs, predicted values and relevant statistics can be computed based on solutions of random effects. PQL-type predictions use $\hat{\gamma}$ as the solution for the random effects to compute predicted values and relevant statistics.

PQL-type predicted values and relevant statistics

Predicted value of the linear predictor

$$\mathrm{x}_i^{\mathrm{T}}\hat{\beta} + \mathrm{z}_i^{\mathrm{T}}\hat{\gamma} + o_i$$

Standard error of the linear predictor

$$\hat{\sigma}_\eta = \sqrt{\mathrm{x}_i^{\mathrm{T}}\Sigma\mathrm{x}_i + \mathrm{z}_i^{\mathrm{T}}\hat{C}_{22}\mathrm{z}_i + 2\mathrm{z}_i^{\mathrm{T}}\hat{C}_{21}\mathrm{x}_i},$$

Predicted value of the mean

$$g^{-1}\left(\mathrm{x}_i^{\mathrm{T}}\hat{\beta} + \mathrm{z}_i^{\mathrm{T}}\hat{\gamma} + o_i\right)$$

For the binomial distribution with 0/1 binary target variable, the predicted category $c(\mathrm{x}_i)$ is

$$c(\mathrm{x}_i) = \begin{cases} 1 \text{ (or sucess)} & \text{if } \hat{\mu}_i \geq 0.5 \\ 0 \text{ (or failure)} & \text{otherwise} \end{cases}.$$

Approximate 100(1−α)% confidence intervals for the mean

$$g^{-1}\left(\mathrm{x}_i^{\mathrm{T}}\hat{\beta}+\mathrm{z}_i^{\mathrm{T}}\hat{\gamma}+o_i \pm t_{\upsilon,\alpha/2}\hat{\sigma}_{\eta}\right)$$

Raw residual on the link function transformation

$$r_{\eta,i}^{R}=v_i-\hat{\eta}_i$$

Raw residual on the original scale of the target

$$r_i^{R}=y_i-\hat{\mu}_i$$

Pearson-type residual on the link function transformation

$$r_{\eta,i}^{P}=\frac{r_{\eta,i}^{R}}{\sqrt{v\hat{a}r(v_i|\gamma)}},$$

where $v\hat{a}r(v_i|\gamma)$ is the ith diagonal element of $v\hat{a}r(\mathrm{v}|\gamma)$ and $v\hat{a}r(\mathrm{v}|\gamma)=g^{'}(\hat{\mu})\mathrm{A}_{\hat{\mu}}^{1/2}\hat{R}\mathrm{A}_{\hat{\mu}}^{1/2}g^{'}(\hat{\mu})$ where $\hat{\mu}$ is an $n\times 1$ vector of PQL-type predicted values of the mean.

Pearson-type residual on the original scale of the target

$$r_i^{P}=\frac{r_i^{R}}{\sqrt{v\hat{a}r(y_i|\gamma)}},$$

where $v\hat{a}r(y_i|\gamma)$ is the ith diagonal element of $v\hat{a}r(\mathrm{y})=\mathrm{A}_{\hat{\mu}_m}^{1/2}\hat{R}\mathrm{A}_{\hat{\mu}_m}^{1/2}$ and $\hat{\mu}_m=\hat{\mu}$.

Classification Table

Suppose that $c\left(j,j^{'}\right)$ is the sum of the frequencies for the observations whose actual target category is j (as row) and predicted target category is $j^{'}$ (as column), $j,j^{'}=1,\cdots,J$ (note that $J=$ 2 for binomial), then

$$c\left(j,j^{'}\right)=\sum_{i=1}^{n}f_iI\left(y_i=j,c\left(x_i\right)=j^{'}\right)$$

where $I\left(\cdot\right)$ is indicator function.

Suppose that $p\left(j,j^{'}\right)$ is the $\left(j,j^{'}\right)^{\mathrm{th}}$ element of the classification table, which is the row percentage, then

$$p_{j,j'} = \left(\frac{c\left(j, j'\right)}{\sum_{j'=1}^{J} c\left(j, j'\right)} \right) \times 100\%$$

The percentage of total correct predictions of the model (or “overall percent correct”) is

$$p_{total} = \left(\frac{\sum_{j=1}^{J} c\left(j, j\right)}{\sum_{j=1}^{J} \sum_{j'=1}^{J} c\left(j, j'\right)} \right) \times 100\%$$

Nominal multinomial distribution

The nominal multinomial distribution requires some extra notation and explanation.

Notation

The following notation is used throughout this section unless otherwise stated:

S	Number of super subjects.
T_s	Number of cases in the *s*th super subject.
y_{st}	Nominal categorical target for the *t*th case in the *s*th super subject. Its category values are denoted as 1, 2, and so on.
J	The total number of categories for target.
y_{st}	Dummy vector of y_{st}, $y_{st} = \left(y_{st,1}, \cdots, y_{st,J-1}\right)^{\mathrm{T}}$, where $y_{st,j} = 1$ if $y_{st} = j$, otherwise $y_{st,j} = 0$. The superscript *T* means the transpose of a matrix or vector.
y_s	$\mathrm{y}_s = \left(\mathrm{y}_{s1}^{\mathrm{T}}, \ldots, \mathrm{y}_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s = 1, \cdots, S.$
y	$y = \left(y_1^{\mathrm{T}}, \cdots, y_S^{\mathrm{T}}\right)^{\mathrm{T}}$
$\pi_{st,j}$	Probability of category *j* for the *t*th case in the *s*th super subject; that is, $\pi_{st,j} = P\left(y_{st} = j\right)$.
π_{st}	$\pi_{st} = \left(\pi_{st,1}, \cdots, \pi_{st,J-1}\right)^{\mathrm{T}}$
π_s	$\pi_s = \left(\pi_{s1}^{\mathrm{T}}, \cdots, \pi_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s = 1, \cdots, S$
π	$\pi = \left(\pi_1^{\mathrm{T}}, \cdots, \pi_S^{\mathrm{T}}\right)^{\mathrm{T}}$

$\eta_{st,j}$ Linear predictor value for category *j* of the *t*th case in the *s*th super subject.

η_{st} $\eta_{st} = (\eta_{st,1}, \cdots, \eta_{st,J-1})^{\mathrm{T}}$

η_s $\eta_s = \left(\eta_{s1}^{\mathrm{T}}, \ldots, \eta_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s = 1, \cdots, S$

η (*n* (*J*–1)) × 1 vector of linear predictor. $\eta = \left(\eta_1^{\mathrm{T}}, \cdots, \eta_S^{\mathrm{T}}\right)^{\mathrm{T}}$

x_{st} *p*× 1 vector of predictor variables for the *t*th case in the *s*th super subject. The first element is 1 if there is an intercept.

X_s $X_s = (I_{J-1} \otimes x_{s1}, \cdots, I_{J-1} \otimes x_{sT_s})^T, s = 1, \cdots, S$

X (*n* (*J*–1)) × (*J*–1)p design matrix of fixed effects, $X = \left(X_1^T, \cdots, X_S^T\right)^T$

z_{st} *r*× 1 vector of coefficients for the random effect corresponding to the *t*th case in the *s*th super subject.

Z_s $Z_s = (I_{J-1} \otimes z_{s1}, \cdots, I_{J-1} \otimes z_{sT_s})^T, s = 1, \cdots, S$

Z Design matrix of random effects, $Z = \bigoplus_{s=1}^{S} Z_s$, where ⊕ is the direct sum of matrices.

O *n*× 1 vector of offsets, $O = (o_{11}, \cdots, o_{1T_1}, \cdots, o_{ST_S})^T$, where o_{st} is the offset value of the *t*th case in the *s*th super subject. This can’t be the target (y) or one of the predictors (X). The offset must be continuous.

O^* $O^* = O \otimes 1_{J-1}$, where 1_q is a length *q* vector of 1.

β_j *p*× 1 vector of unknown parameters for category *j*, $\beta_j = (\beta_{j1}, \cdots, \beta_{jp})^T, j = 1, \cdots, J$. The first element in β_j is the intercept for the category *j*, if there is one.

β $\beta = \left(\beta_1^T, \cdots, \beta_{J-1}^T\right)^T$

γ_{sj} *r* × 1 vector of random effects for category *j* in the *s*th super subject, $j = 1, \cdots, J-1$.

γ_s Random effects for the *s*th super subject, $\gamma_s = \left(\gamma_{s,1}^{\mathrm{T}}, \cdots, \gamma_{s,J-1}^{\mathrm{T}}\right)^{\mathrm{T}}$.

γ $\gamma = \left(\gamma_1^T, \cdots, \gamma_S^T\right)^T$

ω_{st} Scale weight of the *t*th case in the *s*th super subject. It does not have to be integers. If it is less than or equal to 0 or missing, the corresponding case is not used.

ω *n*× 1 vector of scale weight variable, ω $= (\omega_{11}, \cdots, \omega_{1T_1}, \cdots, \omega_{S1}, \cdots, \omega_{ST_S})^{\mathrm{T}}$.

f_{st} Frequency weight of the *t*th case in the *s*th super subject. If it is a non-integer value, it is treated by rounding the value to the nearest integer. If it is less than 0.5 or missing, the corresponding cases are not used.

f *n*× 1 vector of frequency count variable, $f = (f_{11}, \cdots, f_{1T_1}, \cdots, f_{S1,}, \cdots, f_{ST_S})^{\mathrm{T}}$.

N Effective sample size, $N = \sum_{i=1}^{n} f_i$. If frequency count variable *f* is not used, *N* = *n*.

Model

The form of a generalized linear mixed model for nominal target with the random effects is

$$\eta = g\left(E\left(y\right)|\gamma\right) = X\beta + Z\gamma + O^*$$

where η is the linear predictor; **X** is the design matrix for fixed effects; **Z** is the design matrix for random effects; γ is a vector of random effects which are assumed to be normally distributed with mean **0** and variance matrix **G**; $g(.)$ is the logit link function such that

$$\eta_{st,j} = g(\pi_{st,j}) = \log\left(\frac{\pi_{st,j}}{\pi_{st,J}}\right)$$

And its inverse function is

$$\pi_{st,j} = g^{-1}(\eta_{st,j}) = \begin{cases} \frac{\exp(\eta_{st,j})}{1+\sum\limits_{k=1}^{J-1}\exp\left(\eta_{st,k}\right)}, j = 1, \cdots, J-1, \\ \frac{1}{1+\sum\limits_{k=1}^{J-1}\exp\left(\eta_{st,k}\right)}, j = J. \end{cases}$$

The variance of **y**, conditional on the random effects is

$$Var(y|\gamma) = A_\mu^{1/2} R A_\mu^{1/2}$$

where $A_\mu = \bigoplus_{s=1}^{S} \bigoplus_{t=1}^{T_s} \left(diag(\pi_{st}) - \pi_{st}\pi_{st}^{\mathrm{T}}\right)/\omega_{st}$ and R = ϕI which means that R-side effects are not supported for the multinomial distribution. ϕ is set to 1.

Estimation

Linear mixed pseudo model

Similarly to “Linear mixed pseudo model ” on p. 192, we can obtain a weighted linear mixed model

$$v = X\beta + Z\gamma + \epsilon$$

where v ≡ $\mathrm{D}^{-1}(\mathrm{y} - \tilde{\pi}) + g(\tilde{\pi}) - \mathrm{O}^*$ and error terms ε ~ $N\left(0, \mathrm{D}^{-1} A_{\tilde{\pi}}^{1/2} R A_{\tilde{\pi}}^{1/2} \mathrm{D}^{-1}\right)$ with

$$\mathrm{D} = \bigoplus_{s=1}^{S} \bigoplus_{t=1}^{T_s} \mathrm{D}_{st} = \bigoplus_{s=1}^{S} \bigoplus_{t=1}^{T_s} \frac{\mathrm{d}g^{-1}(\tilde{\eta}_{st})}{\mathrm{d}\tilde{\eta}_{st}} = \bigoplus_{s=1}^{S} \bigoplus_{t=1}^{T_s} \left(diag(\tilde{\pi}_{st}) - \tilde{\pi}_{st}\tilde{\pi}_{st}^{\mathrm{T}}\right)$$

and

$$A_{\tilde{\mu}} = \bigoplus_{s=1}^{S} \bigoplus_{t=1}^{T_s} \left(diag(\tilde{\pi}_{st}) - \tilde{\pi}_{st}\tilde{\pi}_{st}^{\mathrm{T}}\right)/\omega_{st}.$$

And block diagonal weight matrix is

$$\tilde{W} = \mathrm{D}A_{\tilde{\mu}}^{-1}\mathrm{D} = \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \omega_{st}\mathrm{D}_{st}.$$

The Gaussian log pseudo-likelihood (PL) and restricted log pseudo-likelihood (REPL), which are expressed as the functions of covariance parameters in θ, corresponding to the linear mixed model for *v* are the following:

$$\ell(\theta;\mathrm{v}) = -\frac{1}{2}\ln|\mathrm{V}(\theta)| - \frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta) - \frac{N}{2}\ln(2\pi)$$

$$\ell_R(\theta;\mathrm{v}) = -\frac{1}{2}\ln|\mathrm{V}(\theta)| - \frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta) - \frac{1}{2}\ln\left|\mathrm{X}^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}\right| - \frac{N-p_x}{2}\ln(2\pi)$$

where $\mathrm{V}(\theta) = Z\mathrm{G}(\theta)Z^T + \tilde{W}^{-1/2}\mathrm{R}(\theta)\tilde{W}^{-1/2}$,$r(\theta) = v - X\hat{\beta}$,$N$ denotes the effective sample size, and p_x denotes the total number of non-redundant parameters for β.

The parameter θ can be estimated by linear mixed model using the objection function $-2\ell(\theta;\mathrm{v})$ or $-2\ell_R(\theta;\mathrm{v})$, β and γ are computed as

$$\hat{\beta} = \left(X^{\mathrm{T}}V\left(\hat{\theta}\right)^{-1}X\right)^{-1}X^{\mathrm{T}}V\left(\hat{\theta}\right)^{-1}v$$

$$\hat{\gamma} = \hat{G}Z^{\mathrm{T}}V\left(\hat{\theta}\right)^{-1}\hat{r}$$

Iterative process

The doubly iterative process for the estimation of θ is the same as that for other distributions, if we replace $\tilde{\mu}$ and $X\tilde{\beta} + Z\tilde{\gamma} + O$ with $\tilde{\pi}$ and $X\tilde{\beta} + Z\tilde{\gamma} + \mathrm{O}^*$ respectively, and set initial estimation of π as

$$\pi^{(0)} = \frac{y + 1/J}{2}$$

For more information, see the topic “Iterative process ” on p. 194.

Post-estimation statistics

Wald confidence intervals

The Wald confidence intervals for covariance parameter estimates are described in “Wald confidence intervals for covariance parameter estimates ” on p. 195.

Statistics for estimates of fixed and random effects

Similarly to “Statistics for estimates of fixed and random effects ” on p. 196, the approximate covariance matrix of $\left(\hat{\beta} - \beta, \hat{\gamma} - \gamma\right)$ is

$$\hat{C} = \begin{bmatrix} X^T R^{*-1} X & X^T R^{*-1} Z \\ Z^T R^{*-1} X & Z^T R^{*-1} Z + G\left(\hat{\theta}\right)^{-1} \end{bmatrix}^{-} = \begin{bmatrix} C_{11} & C_{21}^T \\ C_{21} & C_{22} \end{bmatrix}$$

Where $R^* = v\hat{a}r\,(v|\gamma) = \hat{D}^{-1} A_{\hat{\pi}}^{1/2} R A_{\hat{\pi}}^{1/2} \hat{D}^{-1}$ with $\hat{D}= \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \left(diag\,(\hat{\pi}_{st}) - \hat{\pi}_{st}\hat{\pi}_{st}^{\mathrm{T}}\right)$, and

$$\hat{C}_{11} = \left(X^T \hat{V}^{-1} X\right)^{-}$$

$$\hat{C}_{21} = -\hat{G} Z^T \hat{V}^{-1} X \hat{C}_{11}$$

$$\hat{C}_{22} = \left(Z^T \hat{R}^{-1} Z + \hat{G}^{-1}\right)^{-1} - \hat{C}_{21} X^T \hat{V}^{-1} Z G$$

Statistics for estimates of fixed and random effects on original scale

If the fixed effects are transformed when constructing matrix **X**, then the final estimates of β, C_{11}, C_{21}, and C_{22} above are based on transformed scale, denoted as $\hat{\beta}^*$, $\hat{C}_{11}^*$, $\hat{C}_{21}^*$ and $\hat{C}_{22}^*$, respectively. They would be transformed back on the original scale, denoted as $\hat{\beta}$, $\hat{C}_{11}$, $\hat{C}_{21}$, and $\hat{C}_{22}$, respectively, as follows:

$$\hat{\beta} = T\hat{\beta}^*$$

$$\hat{C}_{11} = T\hat{C}_{11}^* T^{\mathrm{T}}$$

$$\hat{C}_{21} = \hat{C}_{21}^* T^{\mathrm{T}}$$

$$\hat{C}_{22} = \hat{C}_{22}^*$$

where $T = \underset{j=1}{\overset{J-1}{\oplus}} \mathrm{A}_j$.

Estimated covariance matrix of the fixed effects parameters

Model-based estimated covariance

$$\Sigma_m = \hat{C}_{11}$$

Robust estimated covariance of the fixed effects parameters

$$\Sigma_r = \Sigma_m \left(\sum_{s=1}^{S} X_s^T \hat{V}_s^{-1} \hat{r}_s \hat{r}_s^T \hat{V}_s^{-1} X_s \right) \Sigma_m$$

where $\hat{r}_s = v_s - X_s\hat{\beta}$, and v_s is a part of v corresponding to the *s*th super subject.

Standard error for estimates in fixed effects and predictions in random effects

Let $\hat{\beta}_{jc}$ denote a non-redundant fixed effects parameter estimate. Its standard error is the square root of the $((j-1)\,p+c)\,th$ diagonal element of Σ

$$\hat{\sigma}_{\beta_{jc}} = \sqrt{\sigma_{((j-1)p+c),((j-1)p+c)}}$$

The standard error for redundant parameter estimates is set to system missing value.

Similarly, let $\hat{\gamma}_i$ denote the *i*th random effects prediction. Its standard error is the square root of the *i*th diagonal element of $\hat{C}_{22}$:

$$\hat{\sigma}_{\gamma_i} = \sqrt{\hat{C}_{22,ii}}$$

Test statistics for estimates in fixed effects and predictions in random effects

Test statistics for estimates in fixed effects and predictions in random effects are as those described in “Statistics for estimates of fixed and random effects ” on p. 196.

Wald confidence intervals for estimates in fixed effects and random effects predictions

Wald confidence intervals are as those described in “Statistics for estimates of fixed and random effects ” on p. 196.

Testing

Information criteria

These are as described in “Goodness of fit ” on p. 199.

Tests of fixed effects

For each effect specified in the model, a type III test matrix **L** is constructed from the generating matrix $H_\omega = \left(x^T\Omega x\right)^- x^T\Omega x$, where $x = \left(x_{11}^T, \cdots, x_{st}^T, \cdots x_{ST_S}^T\right)^T$ and $\Omega = diag\left(\omega_{11}, \cdots, \omega_{1T_1,}, \cdots, \omega_{S1}, \cdots, \omega_{ST_S}\right)$. Then the test statistic is

$$F = \frac{\hat{\beta}^T L^{*T}\left(L^*\Sigma L^{*T}\right)^{-1} L^*\hat{\beta}}{r_c}$$

where $r_c = rank\left(L^*\Sigma L^{*T}\right)$ and $L^* = I_{J-1} \otimes$ L. The statistic has an approximate F distribution. The numerator degrees of freedom is r_c and the denominator degree of freedom is v. For more information, see the topic “Method for computing degrees of freedom ” on p. 203.

Scoring

PQL-type predicted values and relevant statistics

$(J-1)\times 1$ predicted vector of the linear predictor

$$\hat{\eta}_{st} = (I_{J-1}\otimes x_{st})^{\mathrm{T}}\hat{\beta} + (I_{J-1}\otimes \mathrm{z}_{st})^{\mathrm{T}}\hat{\gamma}_s + 1_{J-1}\otimes o_{st}$$

Estimated covariance matrix of the linear predictor

$$\begin{aligned}\Sigma_{\hat{\eta}_{st}} = {} & (I_{J-1}\otimes x_{st})^T\Sigma(I_{J-1}\otimes x_{st}) + (I_{J-1}\otimes \mathrm{z}_{st})^T\hat{C}^s_{22}(I_{J-1}\otimes \mathrm{z}_{st}) \\ & +(I_{J-1}\otimes \mathrm{z}_{st})^T\hat{C}^s_{21}(I_{J-1}\otimes x_{st}) + (I_{J-1}\otimes x_{st})^T\left(\hat{C}^s_{21}\right)^T(I_{J-1}\otimes \mathrm{z}_{st})\end{aligned}$$

where $\hat{C}^s_{22}$ is a diagonal block corresponding to the *s*th super subject, the approximate covariance matrix of $\hat{\gamma}_s-\gamma_s$; $\hat{C}^s_{21}$ is a part of $\hat{C}_{21}$ corresponding to the *s*th super subject.

The estimated standard error of the *j*th element in $\hat{\eta}_{st}$, $\hat{\eta}_{st,j}$, is the square root of the *j*th diagonal element of $\Sigma_{\hat{\eta}_{st}}$,

$$\sigma_{\hat{\eta}_{st,j}} = \sqrt{\sigma_{\hat{\eta}_{st},jj}}$$

Predicted value of the probability for category *j*

$$\hat{\pi}_{st,j} = g^{-1}(\hat{\eta}_{st,j}) = \begin{cases} \dfrac{\exp\left(\hat{\eta}_{st,j}\right)}{1+\displaystyle\sum_{k=1}^{J-1}\exp\left(\hat{\eta}_{st,k}\right)}, j=1,\cdots,J-1, \\ \dfrac{1}{1+\displaystyle\sum_{k=1}^{J-1}\exp\left(\hat{\eta}_{st,k}\right)}, j=J. \end{cases}$$

Predicted category

$$c(\mathrm{x}_{st}) = arg\max_j \hat{\pi}_{st,j},$$

If there is a tie in determining the predicted category, the tie will be broken by choosing the category with the highest $N_j=\sum_{s=1}^{S}\sum_{t=1}^{T_s} f_{st}y_{st,j}$.If there is still a tie, the one with the lowest category number is chosen.

Approximate 100(1−α)% confidence intervals for the predicted probabilities

The covariance matrix of $\hat{\pi}_{st}$ can be computed as

$$Cov\left(\hat{\pi}_{st}\right) = \nabla g^{-1}(\hat{\eta}_{st})^T\Sigma_{\hat{\eta}_{st}}\nabla g^{-1}\left(\hat{\eta}_{st}\right)$$

where

$$\nabla g^{-1}\left(\hat{\eta}_{st}\right)=\begin{bmatrix} \frac{\partial\hat{\pi}_{st,1}}{\partial\hat{\eta}_{st,1}} & \cdots & \frac{\partial\hat{\pi}_{st,J-1}}{\partial\hat{\eta}_{st,1}} & \frac{\partial\hat{\pi}_{st,J}}{\partial\hat{\eta}_{st,1}} \\ \vdots & & \vdots & \vdots \\ \frac{\partial\hat{\pi}_{st,1}}{\partial\hat{\eta}_{st,J-1}} & \cdots & \frac{\partial\hat{\pi}_{st,J-1}}{\partial\hat{\eta}_{st,J-1}} & \frac{\partial\hat{\pi}_{st,J}}{\partial\hat{\eta}_{st,J-1}} \end{bmatrix}$$

with

$$\frac{\partial\hat{\pi}_{st,j}}{\partial\hat{\eta}_{st,k}}=\begin{cases} \hat{\pi}_{st,j}\left(1-\hat{\pi}_{st,j}\right), j=k \\ -\hat{\pi}_{st,j}\hat{\pi}_{st,k}, j\neq k \end{cases}$$

then the confidence interval is

$$\hat{\pi}_{st,j}\pm t_{\upsilon,\alpha/2}\hat{\sigma}_{\pi_{st,j}}, j=1,\cdots,J$$

where $\hat{\sigma}^2_{\pi_{st,j}}$ is the *j*th diagonal element of $Cov\left(\hat{\pi}_{st}\right)$ and the estimated variance of $\hat{\pi}_{st,j}, j=1,\cdots,J$.

Ordinal multinomial distribution

The ordinal multinomial distribution requires some extra notation and explanation.

Notation

The following notation is used throughout this section unless otherwise stated:

S	Number of super subjects.
T_s	Number of cases in the *s*th super subject.
y_{st}	Ordinal categorical target for the *t*th case in the *s*th super subject. Its category values are denoted as consecutive integers from 1 to *J*.
J	The total number of categories for target.
y_{st}	Indicator vector of y_{st}, $y_{st}=\left(y_{st,1},\cdots,y_{st,J-1}\right)^{\mathrm{T}}$, where $y_{st,j}=1$ if $y_{st}=j$, otherwise $y_{st,j}=0$. The superscript *T* means the transpose of a matrix or vector.
y_s	$\mathrm{y}_s=\left(\mathrm{y}_{s1}^{\mathrm{T}},\ldots,\mathrm{y}_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s=1,\cdots,S.$
y	$y=\left(y_1^{\mathrm{T}},\cdots,y_S^{\mathrm{T}}\right)^{\mathrm{T}}$
$\lambda_{st,j}$	Cumulative target probability for category *j* for the *t*th case in the *s*th super subject; $\lambda_{st,j}=P\left(y_{st}\leq j\right)$.
λ	$\lambda=\left(\lambda_1^{\mathrm{T}},\cdots,\lambda_S^{\mathrm{T}}\right)^{\mathrm{T}}$, where $\lambda_s=\left(\lambda_{s1}^{\mathrm{T}},\cdots,\lambda_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}$ and $\lambda_{st}^{\mathrm{T}}=\left(\lambda_{st,1},\cdots,\lambda_{st,J-1}\right)$, $s=1,\ldots,S$ and $t=1,\ldots,T_s$.
$\pi_{st,j}$	Probability of category *j* for the *t*th case in the *s*th super subject; that is, $\pi_{st,j}=P\left(y_{st}=j\right)$ and $\pi_{st,j}=\lambda_{st,j}-\lambda_{st,j-1}$.

π_{st}	$\pi_{st} = (\pi_{st,1}, \cdots, \pi_{st,J-1})^{\mathrm{T}}$
π_s	$\pi_s = \left(\pi_{s1}^{\mathrm{T}}, \cdots, \pi_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s = 1, \cdots, S$
π	$\pi = \left(\pi_1^{\mathrm{T}}, \cdots, \pi_S^{\mathrm{T}}\right)^{\mathrm{T}}$
$\eta_{st,j}$	Linear predictor value for category *j* of the *t*th case in the *s*th super subject.
η_{st}	$\eta_{st} = (\eta_{st,1}, \cdots, \eta_{st,J-1})^{\mathrm{T}}$
η_s	$\eta_s = \left(\eta_{s1}^{\mathrm{T}}, \ldots, \eta_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s = 1, \cdots, S$
η	(*n* (*J*–1)) × 1 vector of linear predictor. $\eta = \left(\eta_1^{\mathrm{T}}, \cdots, \eta_S^{\mathrm{T}}\right)^{\mathrm{T}}$
x_{st}	*p*× 1 vector of predictors for the *t*th case in the *s*th super subject.
z_{st}	*r*× 1 vector of coefficients for the random effect corresponding to the *t*th case in the *s*th super subject.
O	*n*× 1 vector of offsets, $O = (o_{11}, \cdots, o_{1T_1}, \cdots, o_{ST_S})^T$, where o_{si} is the offset value of the *t*th case in the *s*th super subject. This can't be the target (y) or one of the predictors (X). The offset must be continuous.
O^*	$O^* = O \otimes 1_{J-1}$, where 1_q is a length *q* vector of 1's.
ψ	*J*–1 × 1 vector of threshold parameters, $\psi = (\psi_1, \psi_2, \ldots, \psi_{J-1})^{\mathrm{T}}$ and $\psi_1 < \psi_2 < \cdots < \psi_{J-1}$
β	*p*× 1 vector of unknown parameters.
B	(*J*–1+*p*) × 1 vector of all parameters $\mathrm{B}=\left(\psi^{\mathrm{T}}, \beta^{\mathrm{T}}\right)^{\mathrm{T}}$
ω_{st}	Scale weight of the *t*th case in the *s*th super subject. It does not have to be integers. If it is less than or equal to 0 or missing, the corresponding case is not used.
ω	*n*× 1 vector of scale weight variable, $\omega = (\omega_{11}, \cdots, \omega_{1T_1}, \cdots, \omega_{S1}, \cdots, \omega_{ST_S})^{\mathrm{T}}$.
f_{st}	Frequency weight of the *i*th case in the *s*th super subject. If it is a non-integer value, it is treated by rounding the value to the nearest integer. If it is less than 0.5 or missing, the corresponding cases are not used.
f	*n*× 1 vector of frequency count variable, $f = (f_{11}, \cdots, f_{1T_1}, \cdots, f_{S1,}, \cdots, f_{ST_S})^{\mathrm{T}}$.
N	Effective sample size, $N = \sum_{i=1}^{n} f_i$. If frequency count variable *f* is not used, *N* = *n*.
A ⊗ B	direct (or Kronecker) product of **A** and **B**, which is equal to $\begin{bmatrix} a_{11}\mathrm{B} & a_{12}\mathrm{B} & a_{13}\mathrm{B} \\ a_{21}\mathrm{B} & a_{22}\mathrm{B} & a_{23}\mathrm{B} \\ a_{31}\mathrm{B} & a_{32}\mathrm{B} & a_{33}\mathrm{B} \end{bmatrix}$
1_m	*m*× 1 vector of 1's; $1_m = (1, \ldots, 1)^{\mathrm{T}}$

Model

The form of a generalized linear mixed model for an ordinal target with random effects is

$$\eta = g\left(\lambda\right) = XB + Z\gamma + O^*$$

where η is the expanded linear predictor vector; **λ** is the expanded cumulative target probability vector; $g\,(.)$ is a cumulative link function; **X** is the expanded design matrix for fixed effects arranged as follows

$$\mathrm{X} = \begin{pmatrix} \mathrm{X}_1 \\ \vdots \\ \mathrm{X}_S \end{pmatrix},$$

$$\mathrm{X}_s = \begin{pmatrix} \mathrm{X}_{s1} \\ \vdots \\ \mathrm{X}_{sT_s} \end{pmatrix}_{T_s(J-1)\times(J-1+p)},$$

$$\mathrm{X}_{st} = \begin{pmatrix} \mathrm{I}_{J-1} & 1_{J-1} \otimes -\mathrm{x}_{st}^{\mathrm{T}} \end{pmatrix}_{(J-1)\times(J-1+p)}$$

$$= \begin{pmatrix} 1 & \cdots & 0 & -\mathrm{x}_{st}^{\mathrm{T}} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -\mathrm{x}_{st}^{\mathrm{T}} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \cdots & 0 & -x_{st,1} & \cdots & -x_{st,p} \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 1 & -x_{st,1} & \cdots & -x_{st,p} \end{pmatrix}$$

B=$\left(\psi^{\mathrm{T}},\beta^{\mathrm{T}}\right)^{\mathrm{T}} = \left((\psi_1,\ldots,\psi_{J-1}),\beta^{\mathrm{T}}\right)^{\mathrm{T}}$;**Z** is the expanded design matrix for random effects arranged as follows

$$\mathrm{Z} = \begin{pmatrix} \mathrm{Z}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathrm{Z}_S \end{pmatrix},$$

$$\mathrm{Z}_s = \begin{pmatrix} \mathrm{Z}_{s1} \\ \vdots \\ \mathrm{Z}_{sT_s} \end{pmatrix}_{T_s(J-1)\times r},$$

$$\mathrm{Z}_{st} = \begin{pmatrix} 1_{J-1} \otimes -\mathrm{z}_{st}^{\mathrm{T}} \end{pmatrix}_{(J-1)\times r},$$

γ is a vector of random effects which are assumed to be normally distributed with mean **0** and variance matrix **G**.

The variance of **y**, conditional on the random effects is

$$Var\,(y|\gamma) = A_\mu^{1/2} R A_\mu^{1/2}$$

where $A_\mu = \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \left(diag\,(\pi_{st}) - \pi_{st}\pi_{st}^{\mathrm{T}}\right)/\omega_{st}$ and R $= \phi$I which means that R-side effects are not supported for the multinomial distribution. ϕ is set to 1.

Estimation

Linear mixed pseudo model

Similarly to “Linear mixed pseudo model ” on p. 192, we can obtain a weighted linear mixed model

$$v = X\beta + Z\gamma + \epsilon$$

where $\mathrm{v} \equiv \mathrm{D}^{-1}(\mathrm{y} - \tilde{\pi}) + g\left(\tilde{\lambda}\right) - \mathrm{O}^*$ and error terms $\varepsilon \sim N\left(0, \mathrm{D}^{-1} A_{\tilde{\pi}}^{1/2} R A_{\tilde{\pi}}^{1/2} \left(\mathrm{D}^{-1}\right)^{\mathrm{T}}\right)$ with

$$\mathrm{D} = \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \mathrm{D}_{st} = \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \frac{\mathrm{d}g^{-1}(\tilde{\eta}_{st})}{\mathrm{d}\tilde{\eta}_{st}} = \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \frac{\mathrm{d}\tilde{\lambda}_{st}}{\mathrm{d}\tilde{\eta}_{st}}$$

$$\mathrm{D}_{st} = \begin{bmatrix} \frac{\partial \tilde{\lambda}_{st,1}}{\partial \tilde{\eta}_{st,1}} & 0 & \cdots & 0 & 0 \\ -\frac{\partial \tilde{\lambda}_{st,1}}{\partial \tilde{\eta}_{st,1}} & \frac{\partial \tilde{\lambda}_{st,2}}{\partial \tilde{\eta}_{st,2}} & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & \frac{\partial \tilde{\lambda}_{st,J-2}}{\partial \tilde{\eta}_{st,J-2}} & 0 \\ 0 & 0 & \cdots & -\frac{\partial \tilde{\lambda}_{st,J-2}}{\partial \tilde{\eta}_{st,J-2}} & \frac{\partial \tilde{\lambda}_{st,J-1}}{\partial \tilde{\eta}_{st,J-1}} \end{bmatrix}$$

and

$$A_{\tilde{\mu}} = \underset{s=1}{\overset{S}{\oplus}} \underset{t=1}{\overset{T_s}{\oplus}} \left(diag\left(\tilde{\pi}_{st}\right) - \tilde{\pi}_{st} \tilde{\pi}_{st}^{\mathrm{T}} \right) / \omega_{st}.$$

And block diagonal weight matrix is

$$\tilde{W} = \mathrm{D}^{\mathrm{T}} A_{\tilde{\mu}}^{-1} \mathrm{D}$$

The Gaussian log pseudo-likelihood (PL) and restricted log pseudo-likelihood (REPL), which are expressed as the functions of covariance parameters in θ, corresponding to the linear mixed model for v are the following:

$$\ell(\theta; \mathrm{v}) = -\frac{1}{2} \ln \left|\mathrm{V}\left(\theta\right)\right| - \frac{1}{2} \mathrm{r}(\theta)^{\mathrm{T}} \mathrm{V}(\theta)^{-1} \mathrm{r}(\theta) - \frac{N}{2} \ln\left(2\pi\right)$$

$$\ell_R(\theta; \mathrm{v}) = -\frac{1}{2} \ln \left|\mathrm{V}\left(\theta\right)\right| - \frac{1}{2} \mathrm{r}(\theta)^{\mathrm{T}} \mathrm{V}(\theta)^{-1} \mathrm{r}(\theta) - \frac{1}{2} \ln \left|\mathrm{X}^{\mathrm{T}} \mathrm{V}(\theta)^{-1} \mathrm{X}\right| - \frac{N - p_x}{2} \ln\left(2\pi\right)$$

where $\mathrm{V}\left(\theta\right) = Z\mathrm{G}\left(\theta\right) Z^T + \tilde{W}^{-1/2} \mathrm{R}\left(\theta\right) \tilde{W}^{-1/2}$,$r(\theta) = v - X\hat{\mathbf{B}}$,$N$ denotes the effective sample size, and p_x denotes the total number of non-redundant parameters for $\mathbf{B}$.

The parameter θ can be estimated by linear mixed model using the objection function $-2\ell(\theta; \mathrm{v})$ or $-2\ell_R(\theta; \mathrm{v})$, $\mathbf{B}$ and γ are computed as

$$\hat{\mathbf{B}} = \left(X^{\mathrm{T}} V\left(\hat{\theta}\right)^{-1} X\right)^{-1} X^{\mathrm{T}} V\left(\hat{\theta}\right)^{-1} v$$

$$\hat{\gamma} = \hat{G} Z^{\mathrm{T}} V\left(\hat{\theta}\right)^{-1} \hat{r}$$

Iterative process

The doubly iterative process for the estimation of θ is the same as that for other distributions, if we replace $\tilde{\mu}$ and $X\tilde{\mathbf{B}} + Z\tilde{\gamma} + O$ with $\tilde{\pi}$ and $X\tilde{\mathbf{B}} + Z\tilde{\gamma} + \mathrm{O}^*$ respectively, and set initial estimation of π as

$$\pi^{(0)} = \frac{y + 1/J}{2}$$

For more information, see the topic “Iterative process ” on p. 194.

Post-estimation statistics

Wald confidence intervals

The Wald confidence intervals for covariance parameter estimates are described in “Wald confidence intervals for covariance parameter estimates ” on p. 195.

Statistics for estimates of fixed and random effects

$\hat{C}$ is the approximate covariance matrix of $\left(\hat{\mathbf{B}} - \mathbf{B}, \hat{\gamma} - \gamma\right)$ and R^* in $\hat{C}$ should be

$$R^* = v\hat{a}r\,(v|\gamma) = \mathrm{D}^{-1} A_{\tilde{\pi}}^{1/2} R A_{\tilde{\pi}}^{1/2} \left(\mathrm{D}^{-1}\right)^{\mathrm{T}}.$$

Statistics for estimates of fixed and random effects on original scale

If the fixed effects are transformed when constructing matrix **X**, then the final estimates of **B**, denoted as $\hat{B}^*$. They would be transformed back on the original scale, denoted as $\hat{B}$, as follows:

$$\mathrm{B} = \begin{pmatrix} \psi \\ \beta \end{pmatrix} = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{J-1} \\ \beta \end{pmatrix} = \mathrm{A}\begin{pmatrix} \psi^* \\ \beta^* \end{pmatrix} = \mathrm{AB}^*$$

where

$$\mathrm{A} = \begin{pmatrix} \mathrm{I}_{J-1} & 1_{J-1} \otimes \left(\mathbf{c}^{\mathrm{T}} \mathrm{S}^{-1}\right) \\ 0 & \mathrm{S}^{-1} \end{pmatrix}$$

Estimated covariance matrix of the fixed effects parameters

The estimated covariance matrix of the fixed effects parameters are described in “Statistics for estimates of fixed and random effects ” on p. 196.

Standard error for estimates in fixed effects and predictions in random effects

Let $\hat{\psi}_j, j = 1, \ldots, J-1$, be threshold parameter estimates and $\hat{\beta}_i, i = 1, \ldots, p$, denote non-redundant regression parameter estimates. Their standard errors are the square root of the diagonal elements of Σ_m or Σ_r: $\hat{\sigma}_{\psi_j} = \sqrt{\sigma_{jj}}$ and $\hat{\sigma}_{\beta_i} = \sqrt{\sigma_{(J-1+i),(J-1+i)}}$, respectively, where σ_{ii} is the *i*th diagonal element of Σ_m or Σ_r.

Standard errors for predictions in random effects are as those described in “Statistics for estimates of fixed and random effects ” on p. 196.

Test statistics for estimates in fixed effects and predictions in random effects

The hypotheses $H_{0j} : \psi_j = 0, j = 1, \ldots, J-1$, are tested for threshold parameters using the *t* statistic:

$$t_{\psi_j} = \frac{\hat{\psi}_j}{\hat{\sigma}_{\psi_j}}$$

Test statistics for estimates in fixed effects and predictions in random effects are otherwise as those described in “Statistics for estimates of fixed and random effects ” on p. 196.

Wald confidence intervals for estimates in fixed effects and random effects predictions

The 100(1 – α)% Wald confidence interval for threshold parameter is given by

$$\left(\hat{\psi}_j - t_{\upsilon,\alpha/2}\hat{\sigma}_{\psi_j}, \hat{\psi}_j + t_{\upsilon,\alpha/2}\hat{\sigma}_{\psi_j}\right)$$

Wald confidence intervals are otherwise as those described in “Statistics for estimates of fixed and random effects ” on p. 196.

The degrees of freedom can be computed by the residual method or Satterthwaite method. For the residual method, $\upsilon = N - (J - 1 + p_x)$. For the Satterthwaite method, it should be similar to that described in “Method for computing degrees of freedom ” on p. 203.

Testing

Information criteria

These are as described in “Goodness of fit ” on p. 199, with the following modifications.

For REPL, the value of *N* is chosen to be effective sample size minus number of non-redundant parameters in fixed effects, $\sum_{i=1}^{n} f_i - (J - 1 + p_x)$, where p_x is the number of non-redundant parameters in fixed effects, and *d* is the number of covariance parameters.

For PL, the value of *N* is effective sample size, $\sum_{i=1}^{n} f_i$, and *d* is the number of number of non-redundant parameters in fixed effects, $J - 1 + p_x$, plus the number of covariance parameters.

Tests of fixed effects

For each effect specified in the model excluding threshold parameters, a type I or III test matrix $\mathbf{L}_\mathrm{i}$ is constructed and H_0: $\mathbf{L}_\mathrm{i}\mathbf{B} = \mathbf{0}$ is tested. Construction of matrix $\mathbf{L}_\mathrm{i}$ is based on matrix $\mathrm{H}_\omega = \left(\mathrm{X}_1^\mathrm{T}\boldsymbol{\Omega}\mathrm{X}_1\right)^- \mathrm{X}_1^\mathrm{T}\boldsymbol{\Omega}\mathrm{X}_1$, where $\mathrm{X}_1 = (1, -\mathrm{X})$ and such that $\mathbf{L}_\mathrm{i}\mathbf{B}$ is estimable. Note that $\mathbf{L}_\mathrm{i}\mathbf{B}$ is estimable if and only if $\mathrm{L}_0 = \mathrm{L}_0\mathrm{H}_\omega$, where $\mathrm{L}_0 = (\mathrm{l}_0, \mathrm{L}(\beta))$. Construction of L_0 considers a partition of the more general test matrix $\mathrm{L}_i = (\mathrm{L}_i(\psi), \mathrm{L}_i(\beta))$ first, where $\mathrm{L}_i(\psi) = (\mathrm{l}_1, \ldots, \mathrm{l}_{J-1})$ consists of columns corresponding to the threshold parameters and $\mathrm{L}_i(\beta)$ is the part of $\mathbf{L}_\mathrm{i}$ corresponding to regression parameters, then replace $\mathrm{L}_i(\psi)$ with their sum $\mathrm{l}_0 = \sum_{j=1}^{J-1} \mathrm{l}_j$ to get L_0.

Note that the threshold-parameter effect is not tested for both type I and III analyses and construction of $\mathbf{L}_\mathrm{i}$ is the same as in GENLIN. For more information, see the topic “Default Tests of Model Effects ” on p. 182. Similarly, if the fixed effects are transformed when constructing matrix **X**, then H_ω should be constructed based on transformed values.

Scoring

PQL-type predicted values and relevant statistics

$(J - 1) \times 1$ predicted vector of the linear predictor

$$\hat{\eta}_{st} = X_{st}\hat{\mathbf{B}} + Z_{st}\hat{\gamma}_s + 1_{J-1} \otimes o_{st}$$

Estimated covariance matrix of the linear predictor

$$\Sigma_{\hat{\eta}_{st}} = X_{st}\Sigma X_{st}^\mathrm{T} + \mathrm{Z}_{st}\hat{C}_{22}^s Z_{st}^\mathrm{T} + \mathrm{Z}_{st}\hat{C}_{21}^s X_{st}^\mathrm{T} + X_{st}\left(\hat{C}_{21}^s\right)^\mathrm{T} Z_{st}^\mathrm{T}$$

where $\hat{C}_{22}^s$ is a diagonal block corresponding to the *s*th super subject, the approximate covariance matrix of $\hat{\gamma}_s - \gamma_s$; $\hat{C}_{21}^s$ is a part of $\hat{C}_{21}$ corresponding to the *s*th super subject.

The estimated standard error of the *j*th element in $\hat{\eta}_{st}$, $\hat{\eta}_{st,j}$, is the square root of the *j*th diagonal element of $\Sigma_{\hat{\eta}_{st}}$,

$$\sigma_{\hat{\eta}_{st,j}} = \sqrt{\sigma_{\hat{\eta}_{st},jj}}$$

Predicted value of the cumulative probability for category *j*

$$\hat{\gamma}_{st,j}=g^{-1}(\hat{\eta}_{st,j}), j=1,\ldots,J-1$$

with $\hat{\gamma}_{i,J}=1$.

Predicted category

$$c(\mathrm{x}_{st})=arg\max_{j}\hat{\pi}_{st,j},$$

where $\hat{\pi}_{st,j}=\hat{\gamma}_{st,j}-\hat{\gamma}_{st,j-1}$.

If there is a tie in determining the predicted category, the tie will be broken by choosing the category with the highest $N_j=\sum_{s=1}^{S}\sum_{t=1}^{T_s}f_{st}y_{st,j}$.If there is still a tie, the one with the lowest category number is chosen.

Approximate 100(1−α)% confidence intervals for the cumulative predicted probabilities

$$g^{-1}\left(\hat{\eta}_{st,j}\pm t_{v,\alpha/2}\hat{\sigma}_{\hat{\eta}_{st,j}}\right), j=1,\ldots,J-1,$$

If either endpoint in the argument is outside the valid range for the inverse link function, the corresponding confidence interval endpoint is set to a system missing value.

The degrees of freedom can be computed by the residual method or Satterthwaite method. For the residual method, $v=N-(J-1+p_x)$. For Satterthwaite's approximation, the **L** matrix is constructed by $\left(\mathrm{X}_{st,j},\mathrm{Z}_{st,j}\right)$, where $\mathrm{X}_{st,j}$ and $\mathrm{Z}_{st,j}$ are the *j*th rows of X_{st} and Z_{st}, respectively, corresponding to the *j*th category. For example, the **L** matrix is $\left(1,0,\ldots,0,\ -\mathrm{x}_{st}^{\mathrm{T}},\ -\mathrm{z}_{st}^{\mathrm{T}}\right)_{1\times(J-1+p+r)}$ for the 1st category. The computation should then be similar to that described in "Method for computing degrees of freedom " on p. 203.

References

Agresti, A., J. G. Booth, and B. Caffo. 2000. Random-effects Modeling of Categorical Response Data. *Sociological Methodology*, 30, 27–80.

Diggle, P. J., P. Heagerty, K. Y. Liang, and S. L. Zeger. 2002. *The analysis of Longitudinal Data*, 2 ed. Oxford: Oxford University Press.

Fahrmeir, L., and G. Tutz. 2001. *Multivariate Statistical Modelling Based on Generalized Linear Models*, 2nd ed. New York: Springer-Verlag.

Hartzel, J., A. Agresti, and B. Caffo. 2001. Multinomial Logit Random Effects Models. *Statistical Modelling*, 1, 81–102.

Hedeker, D. 1999. Generalized Linear Mixed Models. In: *Encyclopedia of Statistics in Behavioral Science,* B. Everitt, and D. Howell, eds. London: Wiley, 729–738.

McCulloch, C. E., and S. R. Searle. 2001. *Generalized, Linear, and Mixed Models*. New York: John Wiley and Sons.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

Tuerlinckx, F., F. Rijmen, G. Molenberghs, G. Verbeke, D. Briggs, W. Van den Noortgate, M. Meulders, and P. De Boeck. 2004. Estimation and Software. In: *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach,* P. De Boeck, and M. Wilson, eds. New York: Springer-Verlag, 343–373.

Wolfinger, R., and M. O'Connell. 1993. Generalized Linear Mixed Models: A Pseudo-Likelihood Approach. *Journal of Statistical Computation and Simulation*, 4, 233–243.

Wolfinger, R., R. Tobias, and J. Sall. 1994. Computing Gaussian likelihoods and their derivatives for general linear mixed models. *SIAM Journal on Scientific Computing*, 15:6, 1294–1310.

Imputation of Missing Values

The following methods are available for imputing missing values:

Fixed. Substitutes a fixed value (either the field mean, midpoint of the range, or a constant that you specify).

Random. Substitutes a random value based on a normal or uniform distribution.

Expression. Allows you to specify a custom expression. For example, you could replace values with a global variable created by the Set Globals node.

Algorithm. Substitutes a value predicted by a model based on the C&RT algorithm. For each field imputed using this method, there will be a separate C&RT model, along with a Filler node that replaces blanks and nulls with the value predicted by the model. A Filter node is then used to remove the prediction fields generated by the model.

Details of each imputation method are provided below.

Imputing Fixed Values

For fixed value imputation, three options are available:

Mean. Substitutes the mean of the valid training data values for the field being imputed,

$$\frac{\sum_{i=1}^{n_{valid}} x_i}{n_{valid}}$$

where x_i is the value of field *x* for record *i*, excluding missing values, and n_{valid} is the number of records with valid values for field *x*.

Midrange. Substitutes the value halfway between the minimum and maximum valid values for the field being imputed,

$$x_{\min} + \frac{x_{\max} - x_{\min}}{2} = \frac{x_{\max} + x_{\min}}{2}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum observed valid values for field *x*, respectively.

Constant. Substitutes the user-specified constant value.

For imputing fixed missing values in set or flag fields, only the Constant option is available.

Note: Using fixed imputed values for scale fields will artificially reduce the variance for that field, which can interfere with model building using the field. If you impute using fixed values and find that the field no longer has the expected effect in a model, consider imputing with a different method that has a smaller impact on the field’s variance.

Imputing Random Values

For random value imputation, the options depend on the type of the field being imputed.

Range Fields

For range fields, you can select from a uniform distribution or a normal distribution.

Uniform distribution. Values are generated randomly on the inverval $[x_{\min}, x_{\max}]$, where each value in the interval is equally likely to be generated.

Normal distribution. Values are generated from a normal distribution with mean $\overline{x}_{valid}$ and variance s^2_{valid}, where $\overline{x}_{valid}$ and s^2_{valid} are derived from the valid observed values of *x* in the training data,

$$\overline{x}_{valid} = \frac{\sum_{i=1}^{n_{valid}} x_i}{n_{valid}}$$

$$s^2_{valid} = \frac{\sum_{i=1}^{n_{valid}} \left(x_i - \overline{x}_{valid}\right)^2}{n_{valid} - 1}$$

Set Fields

For set fields, random imputed values are selected from the list of observed values. By default, the probabilities of all values are equal,

$$p(k) = \frac{1}{j}$$

for the *j* possible values of *k*. The Equalize button will return any modified values to the default equal probabilities.

If you select Based on Audit, probabilities are assigned proportional to the relative frequencies of the values in the training data

$$p(k) = \frac{n_k}{n_{valid}}$$

where n_k is the number of records for which $x_i = k$.

If you select Normalize, values are adjusted to sum to 1.0, maintaining the same relative proportions,

$$p_{normalized}(k) = \frac{p(k)}{\sum_k p(k)}$$

This is useful if you want to enter your own weights for generated random values, but they aren't expressed as probabilities. For example, if you know you want twice as many *No* values as *Yes* values, you can enter 2 for *No* and 1 for *Yes* and click Normalize. Normalization will adjust the values to 0.667 and 0.333, preserving the relative weights but expressing them as probabilities.

Imputing Values Derived from an Expression

For expression-based imputation, imputed values are based on a user-specified CLEM expression. The expression is evaluated just as it would be for a filler node. Note that some expressions may return $null or other missing values, with the result that missing values may exist even after imputation with this method.

Imputing Values Derived from an Algorithm

For the Algorithm method, a C&RT model is built for each field to be imputed, using all other input fields as predictors. For each record that is imputed, the model for the field to be imputed is applied to the record to produce a prediction, which is used as the imputed value. For more information, see the topic "Overview of C&RT" on p. 59.

K-Means Algorithm

Overview

The *k*-means method is a clustering method, used to group records based on similarity of values for a set of input fields. The basic idea is to try to discover *k* clusters, such that the records within each cluster are similar to each other and distinct from records in other clusters. *K*-means is an iterative algorithm; an initial set of clusters is defined, and the clusters are repeatedly updated until no more improvement is possible (or the number of iterations exceeds a specified limit).

Primary Calculations

In building the *k*-means model, input fields are encoded to account for differences in measurement scale and type, and the clusters are defined and updated to generate the final model. These calculations are described below.

Field Encoding

Input fields are recoded before the values are input to the algorithm as described below.

Scaling of Range Fields

In most datasets, there's a great deal of variability in the scale of range fields. For example, consider *age* and *number of cars per household*. Depending on the population of interest, *age* may take values up to 80 or even higher. Values for *number of cars per household*, however, are unlikely to exceed three or four in the vast majority of cases.

If you use both of these fields in their natural scale as inputs for a model, the *age* field is likely to be given much more weight in the model than *number of cars per household*, simply because the values (and therefore the differences between records) for the former are so much larger than for the latter.

To compensate for this effect of scale, range fields are transformed so that they all have the same scale. In IBM® SPSS® Modeler, range fields are rescaled to have values between 0 and 1. The transformation used is

$$x_i{}' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}},$$

where x'_{i} is the rescaled value of input field x for record i, x_{i} is the original value of x for record i, $x_{\min}$ is the minimum value of x for all records, and $x_{\max}$ is the maximum value of x for all records.

Numeric Coding of Symbolic Fields

For modeling algorithms that base their calculations on numerical differences between records, symbolic fields pose a special challenge. How do you calculate a numeric difference for two categories?

A common approach to the problem, and the approach used in IBM® SPSS® Modeler, is to recode a symbolic field as a group of numeric fields with one numeric field for each category or value of the original field. For each record, the value of the derived field corresponding to the category of the record is set to 1.0, and all the other derived field values are set to 0.0. Such derived fields are sometimes called **indicator fields**, and this recoding is called **indicator coding**.

For example, consider the following data, where *x* is a symbolic field with possible values A, B, and C:

Record #	***X***	***X***$\boldsymbol{X_1}$**'**	$\boldsymbol{X_2}$**'**	$\boldsymbol{X_3}$**'**
1	B	0	1	0
2	A	1	0	0
3	C	0	0	1

In this data, the original set field *x* is recoded into three derived fields x_1', x_2', and x_3'. x_1' is an indicator for category A, x_2' is an indicator for category B, and x_3' is an indicator for category C.

Applying the Set Encoding Value

After recoding set fields as described above, the algorithm can calculate a numerical difference for the set field by taking the differences on the *k* derived fields (where *k* is the number of categories in the original set). However, there is an additional problem. For algorithms that use the Euclidean distance to measure differences between records, the difference between two records with different values i and j for the set is

$$\sqrt{\sum_{k=1}^{J} (x_{k1} - x_{k2})^2}$$

where *J* is the number of categories, and x_{kn} is value of the derived indicator for category *k* for record *n*. But the values will be different on *two* of the derived indicators, x_{i} and x_{j}. Thus, the sum will be $\sqrt{(1-0)^2 + (0-1)^2} = \sqrt{2} \approx 1.414$, which is larger than 1.0. That means that based on this coding, set fields will have more weight in the model than range fields that are rescaled to 0-1 range.

To account for this bias, *k*-means applies a scaling factor to the derived set fields, such that a difference of values on a set field produces a Euclidean distance of 1.0. The default scaling factor is $\sqrt{\frac{1}{2}} \approx 0.707$. You can see that this value gives the desired result by inserting the value into the distance formula:

$$\sqrt{\left(\sqrt{\frac{1}{2}} - 0\right)^2 + \left(0 - \sqrt{\frac{1}{2}}\right)^2} = \sqrt{\frac{1}{2} + \frac{1}{2}} = 1$$

The user can specify a different scaling factor by changing the Encoding value for sets parameter in the *K*-Means node expert options.

Encoding of Flag Fields

Flag fields are a special case of symbolic fields. However, because they have only two values in the set, they can be handled in a slightly more efficient way than other set fields. Flag fields are represented by a single numeric field, taking the value of 1.0 for the "true" value and 0.0 for the "false" value. Blanks for flag fields are assigned the value 0.5.

Model Parameters

The primary calculation in *k*-means is an iterative process of calculating cluster centers and assigning records to clusters. The primary steps in the procedure are:

1. Select initial cluster centers
2. Assign each record to the nearest cluster
3. Update the cluster centers based on the records assigned to each cluster
4. Repeat steps 2 and 3 until either:
    - In step 3, there is no change in the cluster centers from the previous iteration, or
    - The number of iterations exceeds the maximum iterations parameter

Clusters are defined by their centers. A **cluster center** is a vector of values for the (encoded) input fields. The vector values are based on the mean values for records assigned to the cluster.

Selecting Initial Cluster Centers

The user specifes *k*, the number of clusters in the model. Initial cluster centers are chosen using a maximin algorithm:

1. Initialize the first cluster center as the values of the input fields for the first data record.
2. For each data record, compute the minimum (Euclidean) distance between the record and each defined cluster center.
3. Select the record with the largest minimum distance from the defined cluster centers. Add a new cluster center with values of the input fields for the selected record.
4. Repeat steps 2 and 3 until *k* cluster centers have been added to the model.

Once initial cluster centers have been chosen, the algorithm begins the iterative assign/update process.

Assigning Records to Clusters

In each iteration of the algorithm, each record is assigned to the cluster whose center is closest. Closeness is measured by the usual squared Euclidean distance

$$d_{ij} = ||X_i - C_j||^2 = \sum_{q=1}^{Q} (x_{qi} - c_{qj})^2$$

where X_i is the vector of encoded input fields for record *i*, C_j is the cluster center vector for cluster *j*, *Q* is the number of encoded input fields, x_qi is the value of the *q*th encoded input field for the *i*th record, and c_qj is the value of the *q*th encoded input field for the *j*th record.

For each record, the distance between the record and each cluster center is calculated, and the cluster center whose distance from the record is smallest is assigned as the record's new cluster. When all records have been assigned, the cluster centers are updated.

Updating Cluster Centers

After records have been (re)assigned to their closest clusters, the cluster centers are updated. The cluster center is calculated as the mean vector of the records assigned to the cluster:

$$C_j = \overline{X}_j$$

where the components of the mean vector $\overline{X}_j$ are calculated in the usual manner,

$$\overline{x}_{qj} = \frac{\sum_{i=1}^{n_j} x_{qi}(j)}{n_j}$$

where n_j is the number of records in cluster *j*, $x_\mathrm{qi}(j)$ is the *q*th encoded field value for record *i* which is assigned to cluster *j*.

Blank Handling

In *k*-means, blanks are handled by substituting "neutral" values for the missing ones. For range and flag fields with missing values (blanks and nulls), the missing value is replaced with 0.5. For set fields, the derived indicator field values are all set to 0.0.

Effect of Options

There are several options that affect the way the model calculations are carried out.

Maximum Iterations

The maximum iterations parameter controls how long the algorithm will continue searching for a stable cluster solution. The algorithm will repeat the classify/update cycle no more than the number of times specified. If and when this limit is reached, the algorithm terminates and produces the current set of clusters as the final model.

Error Tolerance

The error tolerance parameter provides another means of controlling how long the algorithm will continue searching for a stable cluster solution. The maximum change in cluster means for an iteration *t* is calculated as

$$\max_{J} || \, C_j(t) - C_j(t-1) \, ||$$

where $C_j(t)$ is the cluster center vector for the *j*th cluster at iteration *t* and $C_j(t - 1)$ is the cluster center vector at the previous iteration. If the maximum change is less than the specified tolerance for the current iteration, the algorithm terminates and produces the current set of clusters as the final model.

Encoding Value for Sets

The encoding value for sets parameter controls the relative weighting of set fields in the *k*-means algorithm. The default value of $\sqrt{0.5} \approx 0.707$ provides an equal weighting between range fields and set fields. To emphasize set fields more heavily, you can set the encoding value closer to 1.0; to emphasize range fields more, set the encoding value closer to 0.0. For more information, see the topic “Numeric Coding of Symbolic Fields” on p. 227.

Model Summary Statistics

Cluster proximities are calculated as the Euclidean distance between cluster centers,

$$d_{ij} = ||C_i - C_j|| = \sqrt{\sum_{q=1}^{Q} (c_{qi} - c_{qj})^2}$$

Generated Model/Scoring

Generated *k*-means models provide predicted cluster memberships and distance from cluster center for each record.

Predicted Cluster Membership

When assigning a new record with a predicted cluster membership, the Euclidean distance between the record and each cluster center is calculated (in the same manner as for assigning records during the model building phase), and the cluster center closest to the record is assigned as the predicted cluster for the record.

Distances

The value of the **distance field** for each record, if requested, is calculated as the Euclidean distance between the record and its assigned cluster center,

$$d_{ij} = ||X_i - C_j|| = \sqrt{\sum_{q=1}^{Q} (x_{qi} - c_{qj})^2}$$

Blank Handling

In *k*-means, scoring records with a generated model handles blanks in the same way they are handled during model building. For more information, see the topic "Blank Handling" on p. 230.

KNN Algorithms

Nearest Neighbor Analysis is a method for classifying cases based on their similarity to other cases. In machine learning, it was developed as a way to recognize patterns of data without requiring an exact match to any stored patterns, or cases. Similar cases are near each other and dissimilar cases are distant from each other. Thus, the distance between two cases is a measure of their dissimilarity.

Cases that are near each other are said to be “neighbors.” When a new case (holdout) is presented, its distance from each of the cases in the model is computed. The classifications of the most similar cases – the nearest neighbors – are tallied and the new case is placed into the category that contains the greatest number of nearest neighbors.

You can specify the number of nearest neighbors to examine; this value is called *k*. The pictures show how a new case would be classified using two different values of *k*. When *k* = 5, the new case is placed in category *1* because a majority of the nearest neighbors belong to category *1*. However, when *k* = 9, the new case is placed in category *0* because a majority of the nearest neighbors belong to category *0*.

Nearest neighbor analysis can also be used to compute values for a continuous target. In this situation, the average or median target value of the nearest neighbors is used to obtain the predicted value for the new case.

Notation

The following notation is used throughout this chapter unless otherwise stated:

$\mathbf{Y}$	Optional 1×N vector of responses with element y_n, where n=1,...,N indexes the cases.
$\mathbf{X}^0$	P^0×N matrix of features with element x^0_{pn}, where p=1,...,P^0 indexes the features and n=1,...,N indexes the cases.
$\mathbf{X}$	P×N matrix of encoded features with element x_{pn}, where p=1,...,P indexes the features and n=1,...,N indexes the cases.
P	Dimensionality of the feature space; the number of continuous features plus the number of categories across all categorical features.
N	Total number of cases.
$N_j, j = 1, 2, \cdots, J$	The number of cases with $Y = j$, where Y is a response variable with J categories
$\hat{N}_j$	The number of cases which belong to class j and are correctly classified as j.
$\hat{N}^*_j$	The total number of cases which are classified as j.

Preprocessing

Features are coded to account for differences in measurement scale.

Continuous

Continuous features are optionally coded using adjusted normalization:

$$x_{pn} = \frac{2\left(x_{pn}^0 - \min\left(x_p^0\right)\right)}{\max\left(x_p^0\right) - \min\left(x_p^0\right)} - 1$$

where x_{pn} is the normalized value of input feature *p* for case *n*, x_p^0 is the original value of the feature for case *n*, $\min\left(x_p^0\right)$ is the minimum value of the feature for all training cases, and $\max\left(x_p^0\right)$ is the maximum value for all training cases.

Categorical

Categorical features are always temporarily recoded using one-of-*c* coding. If a feature has *c* categories, then it is is stored as *c* vectors, with the first category denoted (1,0,...,0), the next category (0,1,0,...,0), ..., and the final category (0,0,...,0,1).

Training

Training a nearest neighbor model involves computing the distances between cases based upon their values in the feature set. The nearest neighbors to a given case have the smallest distances from that case. The distance metric, choice of number of nearest neighbors, and choice of the feature set have the following options.

Distance Metric

We use one of the following metrics to measure the similarity of query cases and their nearest neighbors.

Euclidean Distance. The distance between two cases is the square root of the sum, over all dimensions, of the weighted squared differences between the values for the cases.

$$Euclidean_{ih} = \sqrt{\sum_{p=1}^{P} w_{(p)} \left(x_{(p)i} - x_{(p)h}\right)^2}$$

City Block Distance. The distance between two cases is the sum, over all dimensions, of the weighted absolute differences between the values for the cases.

$$CityBlock_{ih} = \sum_{p=1}^{P} w_{(p)} \left|x_{(p)i} - x_{(p)h}\right|$$

The feature weight $w(p)$ is equal to 1 when feature importance is not used to weight distances; otherwise, it is equal to the normalized feature importance:

$$w_{(p)} = FI_{(p)} / \sum_{p=1}^{P} FI_{(p)}$$

See "Output Statistics " for the computation of feature importance $FI_{(p)}$.

Crossvalidation for Selection of k

Cross validation is used for automatic selection of the number of nearest neighbors, between a minimum $k_{\min}$ and maximum $k_{\max}$. Suppose that the training set has a cross validation variable with the integer values 1,2,..., *V*. Then the cross validation algorithm is as follows:

- For each $k \in [k_{\min}, k_{\max}]$, compute the average error rate or sum-of square error of *k*: $CV_k = \sum_{v=1}^{V} e_v / V$, where e_v is the error rate or sum-of square error when we apply the Nearest Neighbor model to make predictions on the cases with $X = v$; that is, when we use the other cases as the training dataset.
- Select the optimal *k* as: $\hat{k} = arg\{\min CV_k : k_{\min} \leq k \leq k_{\max}\}$.

Note: If multiple values of *k* are tied on the lowest average error, we select the smallest *k* among those that are tied.

Feature Selection

Feature selection is based on the wrapper approach of Cunningham and Delany (2007) and uses forward selection which starts from J_{Forced} features which are entered into the model. Further features are chosen sequentially; the chosen feature at each step is the one that causes the largest decrease in the error rate or sum-of squares error.

Let S_J represent the set of *J* features that are currently chosen to be included, S_J^c represents the set of remaining features and e_J represents the error rate or sum-of-squares error associated with the model based on S_J .

The algorithm is as follows:

- Start with $J = J_{Forced}$ features.
- For each feature in S_J^c , fit the *k* nearest neighbor model with this feature plus the existing features in S_J and calculate the error rate or sum-of square error for each model. The feature in S_J^c whose model has the smallest error rate or sum-of square error is the one to be added to create S_{J+1}.
- Check the selected stopping criterion. If satisfied, stop and report the chosen feature subset. Otherwise, *J*=*J*+1 and go back to the previous step.

Note: the set of encoded features associated with a categorical predictor are considered and added together as a set for the purpose of feature selection.

Stopping Criteria

One of two stopping criteria can be applied to the feature selection algorithm.

Fixed number of features. The algorithm adds a fixed number of features, J_{add}, in addition to those forced into the model. The final feature subset will have $J_{add} + J_{Forced}$ features. J_{add} may be user-specified or computed automatically; if computed automatically the value is

$$J_{add} = \max\left\{\min\left(20, P^0\right) - J_{Forced},\;\; 0\right\}$$

When this is the stopping criterion, the feature selection algorithm stops when J_{add} features have been added to the model; that is, when $J_{add} = J + 1$, stop and report S_{J+1} as the chosen feature subset.

Note: if $J_{add} = 0$, no features are added and S_J with $J = J_{Forced}$ is reported as the chosen feature subset.

Change in error rate or sum of squares error. The algorithm stops when the change in the absolute error ratio indicates that the model cannot be further improved by adding more features. Specifically, if $e_{J+1} = 0$ or $e_J \geq e_{J+1}$ and

$$\frac{|e_J - e_{J+1}|}{e_J} \leq \Delta_{\min}$$

where $\Delta_{\min}$ is the specified minimum change, stop and report S_{J+1} as the chosen feature subset.

If $e_J < e_{J+1}$ and

$$\frac{|e_J - e_{J+1}|}{e_J} > 2\Delta_{\min}$$

stop and report S_J as the chosen feature subset.

Note: if $e_J = 0$ for $J = J_{Forced}$, no features are added and S_J with $J = J_{Forced}$ is reported as the chosen feature subset.

Combined k and Feature Selection

The following method is used for combined neighbors and features selection.

1. For each *k*, use the forward selection method for feature selection.
2. Select the *k*, and accompanying feature set, with the lowest error rate or the lowest sum-of-squares error.

Blank Handling

All records with missing values for any input or output field are excluded from the estimation of the model.

Output Statistics

The following statistics are available.

Percent correct for class j

$$\frac{\hat{N}_j}{N_j} \times 100\%$$

Overall percent for class j

$$\frac{\hat{N}_j^*}{N} \times 100\%$$

Intersection of Overall percent and percent correct

$$\left(\sum_{j=1}^{J} \hat{N}_j / N\right) \times 100\%$$

Error rate of classification

$$\left(1 - \sum_{j=1}^{J} \hat{N}_j / N\right) \times 100\%$$

Sum-of-Square Error for continuous response

$$\sum_{n=1}^{N} (y_n - \hat{y}_n)^2$$

where $\hat{y}_n$ is the estimated value of y_n.

Feature Importance

Suppose there are $X_{(1)}, X_{(2)} \cdots X_{(m)}$ $\left(1 \leq m \leq P^0\right)$ in the model from the forward selection process with the error rate or sum-of-squares error e. The importance of feature $X_{(p)}$ in the model is computed by the following method.

- Delete the feature $X_{(p)}$ from the model, make predictions and evaluate the error rate or sum-of-squares error $e_{(p)}$ based on features $X_{(1)}, X_{(2)} \cdots X_{(p-1)}, X_{(p+1)}, \cdots, X_{(m)}$.
- Compute the error ratio $e_{(p)} + \frac{1}{m}$.

The feature importance of $X_{(p)}$ is $FI_{(p)} = e_{(p)} + \frac{1}{m}$

Scoring

After we find the *k* nearest neighbors of a case, we can classify it or predict its response value.

Categorical response

Classify each case by majority vote of its *k* nearest neighbors among the training cases.

- If multiple categories are tied on the highest predicted probability, then the tie should be broken by choosing the category with largest number of cases in training set.
- If multiple categories are tied on the largest number of cases in the training set, then choose the category with the smallest data value among the tied categories. In this case, categories are assumed to be in the ascending sort or lexical order of the data values.

We can also compute the predicted probability of each category. Suppose k_j is the number of cases of the *j*th category among the *k* nearest neighbors. Instead of simply estimating the predicted probability for the *j*th category by $\frac{k_j}{k}$, we apply a Laplace correction as follows:

$$\frac{k_j + 1}{k + J}$$

where *J* is the number of categories in the training data set.

The effect of the Laplace correction is to shrink the probability estimates towards to 1/*J* when the number of nearest neighbors is small. In addition, if a query case has *k* nearest neighbors with the same response value, the probability estimates are less than 1 and larger than 0, instead of 1 or 0.

Continuous response

Predict each case using the mean or median function.

Mean function.$\hat{y}_n = \sum_{m \in Nearest(n)} y_m / k$, where $Nearest\,(n)$ is the index set of those cases that are the nearest neighbors of case *n* and y_m is the value of the continuous response variable for case *m*.

Median function. Suppose that $y_m, m \in Nearest\,(n)$ are the values of the continuous response variable, and we arrange $y_m, m \in Nearest\,(n)$ from the lowest value to the highest value and denote them as $y_{(j_1)} \le y_{(m_2)} \le \cdots \le y_{(m_k)}$, then the median is

$$\hat{y}_n = \begin{cases} y_{\left(\frac{k+1}{2}\right)} & k \text{ is odd} \\ \frac{y_{\left(\frac{k}{2}\right)} + y_{\left(\frac{k}{2}\right)+1}}{2} & k \ \text{ is even} \end{cases}$$

Blank Handling

Records with missing values for any input field cannot be scored and are assigned a predicted value and probability value(s) of $null$.

References

Arya, S., and D. M. Mount. 1993. Algorithms for fast vector quantization. In: *Proceedings of the Data Compression Conference 1993,* , 381–390.

Cunningham, P., and S. J. Delaney. 2007. k-Nearest Neighbor Classifiers. *Technical Report UCD-CSI-2007-4, School of Computer Science and Informatics, University College Dublin, Ireland*, , – .

Friedman, J. H., J. L. Bentley, and R. A. Finkel. 1977. An algorithm for finding best matches in logarithm expected time. *ACM Transactions on Mathematical Software*, 3, 209–226.

Kohonen Algorithms

Overview

Kohonen models (Kohonen, 2001) are a special kind of neural network model that performs **unsupervised learning**. It takes the input vectors and performs a type of spatially organized clustering, or feature mapping, to group similar records together and collapse the input space to a two-dimensional space that approximates the multidimensional proximity relationships between the clusters.

The Kohonen network model consists of two layers of neurons or units: an input layer and an output layer. The input layer is fully connected to the output layer, and each connection has an associated weight. Another way to think of the network structure is to think of each output layer unit having an associated center, represented as a vector of inputs to which it most strongly responds (where each element of the center vector is a weight from the output unit to the corresponding input unit).

Primary Calculations

Field Encoding

Scaling of Range Fields

In most datasets, there's a great deal of variability in the scale of range fields. For example, consider *age* and *number of cars per household*. Depending on the population of interest, *age* may take values up to 80 or even higher. Values for *number of cars per household*, however, are unlikely to exceed three or four in the vast majority of cases.

If you use both of these fields in their natural scale as inputs for a model, the *age* field is likely to be given much more weight in the model than *number of cars per household*, simply because the values (and therefore the differences between records) for the former are so much larger than for the latter.

To compensate for this effect of scale, range fields are transformed so that they all have the same scale. In IBM® SPSS® Modeler, range fields are rescaled to have values between 0 and 1. The transformation used is

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}},$$

where x'_i is the rescaled value of input field x for record i, x_i is the original value of x for record i, $x_{\min}$ is the minimum value of x for all records, and $x_{\max}$ is the maximum value of x for all records.

Numeric Coding of Symbolic Fields

For modeling algorithms that base their calculations on numerical differences between records, symbolic fields pose a special challenge. How do you calculate a numeric difference for two categories?

A common approach to the problem, and the approach used in IBM® SPSS® Modeler, is to recode a symbolic field as a group of numeric fields with one numeric field for each category or value of the original field. For each record, the value of the derived field corresponding to the category of the record is set to 1.0, and all the other derived field values are set to 0.0. Such derived fields are sometimes called **indicator fields**, and this recoding is called **indicator coding**.

For example, consider the following data, where *x* is a symbolic field with possible values A, B, and C:

Record #	***X***	***X*₁'**	***X*₂'**	***X*₃'**
1	B	0	1	0
2	A	1	0	0
3	C	0	0	1

In this data, the original set field *x* is recoded into three derived fields x_1', x_2', and x_3'. x_1' is an indicator for category A, x_2' is an indicator for category B, and x_3' is an indicator for category C.

Encoding of Flag Fields

Flag fields are a special case of symbolic fields. However, because they have only two values in the set, they can be handled in a slightly more efficient way than other set fields. Flag fields are represented by a single numeric field, taking the value of 1.0 for the "true" value and 0.0 for the "false" value. Blanks for flag fields are assigned the value 0.5.

Model Parameters

In a Kohonen model, the parameters are represented as **weights** between input units and output units, or alternately, as a **cluster center** associated with each output unit. Input records are presented to the network, and the cluster centers are updated in a manner similar to that used in building a *k*-means model, with an important difference: the clusters are arranged spatially in a two-dimensional grid, and each record affects not only the unit (cluster) to which it is assigned but also units within a **neighborhood** about the winning unit. For more information, see the topic "Neighborhoods" on p. 243.

Training of the Kohonen network proceeds as follows:

- The network is initialized with small random weights.
- Input records are presented to the network in random order. As each record is presented, the output unit with the closest center to the input vector is identified as the winning unit.For more information, see the topic "Distances" on p. 243.
- The weights of the winning unit are adjusted to move the cluster center closer to the input vector. For more information, see the topic "Weight Updates" on p. 243.
- If the neighborhood size is greater than zero, then other output units that are within the neighborhood of the winning unit are also updated so their centers are closer to the input vector.
- At the end of each cycle, the learning rate parameter η (eta) is updated.

- This process repeats until one of the stopping criteria is met. Training proceeds in two phases, a gross structure phase and a fine tuning phase. Typically the first phase has a relatively large neighborhood size and large eta to learn the overall structure of the data, and the second phase uses a smaller neighborhood and smaller eta to fine tune the cluster centers.

Distances

Distances in a Kohonen network are calculated as Euclidean distance between the encoded input vector and the cluster center for the output unit,

$$d_{ij} = \sqrt{\sum_k \left(x_{ik} - w_{jk}\right)^2}$$

where x_{ik} is the value of the *k*th input field for the *ith* record, and w_{jk} is the weight for the *k*th input field on the *j*th output unit.

The activation of an output unit is simply the Euclidean distance between the output unit's weight vector (its center) and the input vector. Note that for Kohonen networks, the output unit with the *lowest* activation is the winning unit. This is in contrast to other types of neural networks, where higher activation represents stronger response.

Neighborhoods

The neighborhood function is based on the Chebychev distance, which considers only the maximum distance on any single dimension:

$$d_c\left(x, y\right) = \max_i \left|x_i - y_i\right|$$

where x_i is the location of unit *x* on dimension *i* of the output grid, and y_i is the location of another unit *y* on the same dimension.

An output unit o_j is considered to be in the neighborhood of another output unit o_i if $d_c\left(o_i, o_j\right) < n$, where *n* is the neighborhood size.

Neighborhood size remains constant during each phase, but different phases usually use different neighborhood sizes. By default, $n = 2$ for Phase 1 and $n = 1$ for Phase 2.

Weight Updates

For the winning output node, and its neighbors if the neighborhood is > 0, the weights are adjusted by adding a portion of the difference between the input vector and the current weight vector. The magnitude of the change is determined by the learning rate parameter η (eta). The weight change is calculated as

$$\Delta W = \eta \cdot (W - I)$$

where W is the weight vector for the output unit being updated, I is the input vector, and η is the learning rate parameter. In individual unit terms,

$$\Delta w_j = \eta \cdot (w_j - i_j)$$

where w_j is the weight corresponding to input unit *j* for the output unit being updated, and i_j is the *j*th input unit.

Eta Decay

At the end of each cycle, the value of η is updated. The value of η generally decreases across training cycles. The user can control the rate of decrease by selecting either linear or exponential decay.

Linear decay. This is the default decay rate. When this option is selected, the value of η decays in a linear fashion, decreasing by a fixed amount each cycle, according to the formula

$$\eta(t+1) = \eta(t) - \left(\frac{\eta(0) - \eta_{low}}{c}\right)$$

where $\eta\,(0)$ is the initial eta value for the current phase, and η_{low} is the low eta for the current training phase, calculated as the lesser of the initial eta values for the current phase and the following phase, and *c* is the number of cycles set for the current phase.

Exponential decay. When this option is selected, the value of η decays in an exponential fashion, decreasing by a fixed proportion each cycle, according to the formula

$$\eta(t+1) = \eta(t) \cdot \exp\left(\frac{\log\left(\frac{\eta_{low}}{\eta(0)}\right)}{c}\right)$$

The value of η_{low} has a minimum value of 0.0001 to prevent arithmetic errors in taking the logarithm.

Blank Handling

In Kohonen networks, blanks are handled by substituting “neutral” values for the missing ones. For range and flag fields with missing values (blanks and nulls), the missing value is replaced with 0.5. For range fields, numeric values outside the range limits found in the field’s type information are coerced to the type-defined range. For set fields, the derived indicator field values are all set to 0.0.

Effect of Options

Stop on. By default, training executes the specified number of cycles for each phase. If the Time option is selected, training stops when the elapsed time reaches the specified limit (or sooner if the specified number of cycles for both phases is completed before the time limit is reached).

Random seed. Sets the seed for the random number generator used to initialize the weights of the new network as well as the order of presentation for training records. Select a fixed seed value to create a reproducible network.

Generated Model/Scoring

Cluster Membership

Cluster membership for a new record is derived by presenting the input vector for the record to the network and identifying the output neuron with the closest weight vector, as described in Distances above. The predicted value is returned as the *x* and *y* coordinates of the winning neuron in the output grid.

Blank Handling

Blank handling for scoring is the same as during model building. For more information, see the topic “Blank Handling” on p. 244.

Linear modeling algorithms

Linear models predict a continuous target based on linear relationships between the target and one or more predictors.

For algorithms on enhancing model accuracy, enhancing model stability, or working with very large datasets, see “Ensembles Algorithms” on p. 125.

Notation

The following notation is used throughout this chapter unless otherwise stated:

n	Number of distinct records in the dataset. It is an integer and $n \geq 1$.
p	Number of parameters (including parameters for dummy variables but excluding the intercept) in the model. It is an integer and $p \geq 0$.
p^*	Number of non-redundant parameters (excluding the intercept) currently in the model. It is an integer and $0 \leq p^* \leq p$.
p^c	Number of non-redundant parameters currently in the model. $p^c = p^* + 1$
p^e	Number of effects excluding the intercept. It is an integer and $0 \leq p^e \leq p$
$\mathbf{y}$	$n \times 1$ target vector with elements y_i.
f	$n \times 1$ frequency weight vector.
g	$n \times 1$ regression weight vector.
N	Effective sample size. It is an integer and $N = \sum_{i=1}^{n} f_i$. If there is no frequency weight vector, N=n.
$\mathbf{X}$	$n \times (p+1)$ design matrix with element x_{ij}. The rows represent the records and the columns represent the parameters.
ϵ	$n \times 1$ vector of unobserved errors.
β	$(p+1) \times 1$ vector of unknown parameters; $\beta = (\beta_0, \beta_1, \cdots \beta_p)$. β_0 is the intercept.
$\hat{\beta}$	$(p+1) \times 1$ vector of parameter estimates.
b	$(p+1) \times 1$ vector of standardized parameter estimates. It is the result of a sweep operation on matrix $\mathbf{R}$. b_0 is the standardized estimate of the intercept and is equal to 0.
$\hat{\mathbf{y}}$	$n \times 1$ vector of predicted target values.
$\overline{X}_j$	Weighted sample mean for X_j, $j = 1, 2, \cdots p$
$\overline{y}$	Weighted sample mean for $\mathbf{y}$.
S_{ij}	Weighted sample covariance between X_i and X_j, $i, j = 1, 2, \cdots p$.
S_{iy}	Weighted sample covariance between X_i and $\mathbf{y}$.
S_{yy}	Weighted sample variance for $\mathbf{y}$.
$\mathbf{R}$	$(p+1) \times (p+1)$ weighted sample correlation matrix for $\mathbf{X}$ (excluding the intercept, if it exists) and $\mathbf{y}$.
$\tilde{\mathbf{R}}$	The resulting matrix after a sweep operation whose elements are $\tilde{r}_{ij}$.

Model

Linear regression has the form

$$y = X\beta + \varepsilon$$

where ε follows a normal distribution with mean 0 and variance $\sigma^2 D^{-1}$, where $D^{-1} = diag(1/g_1, \ldots, 1/g_n)$. The elements of ε are independent with respect to each other.

Notes:

- **X** can be any combination of continuous and categorical effects.
- Constant columns in the design matrix are not used in model building.
- If *n*=1 or the target is constant, no model is built.

Missing values

Records with missing values are deleted listwise.

Least squares estimation

The coefficients are estimated by the least squares (LS) method. First, we transform the model by pre-multiplying $D^{1/2}$ as follows:

$$D^{1/2}y = D^{1/2}X\beta + D^{1/2}\varepsilon$$

so that the new unobserved error $D^{1/2}\varepsilon$ follows a normal distribution $N_n\left(0, \sigma^2\mathbf{I}\right)$, where **I** is an identity matrix and $D^{1/2} = diag\left(\sqrt{g_1}, \ldots, \sqrt{g_n}\right)$. Then the least squares estimates of β can be obtained from the following formula

$$\hat{\beta} = arg\min_{\beta}\left(D^{1/2}y - D^{1/2}\mathbf{X}\beta\right)^{\mathrm{T}} F\left(D^{1/2}\mathbf{y} - D^{1/2}\mathbf{X}\beta\right)$$

where $F = \mathrm{diag}(f_1, \ldots, f_n)$. Note that

$$\begin{aligned}&\left(D^{1/2}\mathbf{y} - D^{1/2}\mathbf{X}\beta\right)^{\mathrm{T}} F\left(D^{1/2}\mathbf{y} - D^{1/2}\mathbf{X}\beta\right)\\ &\quad = (\mathbf{y} - \mathbf{X}\beta)^{\mathrm{T}} D^{1/2} F D^{1/2} (\mathbf{y} - \mathbf{X}\beta)\\ &\qquad = (\mathbf{y} - \mathbf{X}\beta)^{\mathrm{T}} W (\mathbf{y} - \mathbf{X}\beta)\end{aligned}$$

where $W = \mathrm{diag}(w_1, \ldots, w_n) = \mathrm{diag}(g_1 f_1, \ldots, g_n f_n)$, so the closed form solution of $\hat{\beta}$ is

$$\hat{\beta} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{y}$$

$\hat{\beta}$ is computed by applying sweep operations instead of the equation above. In addition, sweep operations are applied to the transformed scale of **X** and **y** to achieve numerical stability. Specifically, we construct the weighted sample correlation matrix **R** then apply sweep operations to it. The **R** matrix is constructed as follows.

First, compute weighted sample means, variances and covariances among $\mathbf{X}_\mathrm{i}$, $\mathbf{X}_\mathrm{j}$, $i, j = 1, \ldots, p$, and **y** :

Weighted sample means of $\mathbf{X}_\mathrm{i}$ and **y** are $\overline{X}_i = \frac{1}{\sum_{k=1}^{n} w_k} \sum_{k=1}^{n} w_k x_{ki}$ and $\overline{y} = \frac{1}{\sum_{k=1}^{n} w_k} \sum_{k=1}^{n} w_k y_k$;

Weighted sample covariance for $\mathbf{X}_\mathrm{i}$ and $\mathbf{X}_\mathrm{j}$ is $S_{ij} = \frac{1}{N-1} \sum_{k=1}^{n} w_k \left(x_{ki} - \overline{X}_i\right) \left(x_{kj} - \overline{X}_j\right)$;

Weighted sample covariance for $\mathbf{X}_\mathrm{i}$ and **y** is $S_{iy} = \frac{1}{N-1} \sum_{k=1}^{n} w_k \left(x_{ki} - \overline{X}_i\right) \left(y_k - \overline{y}\right)$;

Weighted sample variance for **y** is $S_{yy} = \frac{1}{N-1} \sum_{k=1}^{n} w_k (y_k - \overline{y})^2$.

Second, compute weighted sample correlations $r_{ij} = \frac{S_{ij}}{\sqrt{S_{ii} S_{jj}}}$, $i, j = 1, ..., p$ and y.

Then the matrix **R** is

$$\mathrm{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} & r_{1y} \\ r_{21} & r_{22} & \cdots & r_{2p} & r_{2y} \\ \vdots & \vdots & \ddots & \vdots & \\ r_{p1} & r_{2p} & \cdots & r_{pp} & r_{py} \\ r_{y1} & r_{y2} & \cdots & r_{yp} & r_{yy} \end{bmatrix} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} \\ \mathrm{R}_{12}^{\mathrm{T}} & R_{22} \end{bmatrix}$$

If the sweep operations are repeatedly applied to each row of $\mathbf{R}_{11}$, where $\mathbf{R}_{11}$ contains the predictors in the model at the current step, the result is

$$\tilde{\mathbf{R}} = \begin{bmatrix} \mathbf{R}_{11}^{-1} & \mathbf{R}_{11}^{-1}\mathbf{R}_{12} \\ -\mathbf{R}_{12}^{\mathrm{T}}\mathbf{R}_{11}^{-1} & R_{22} - \mathbf{R}_{12}^{\mathrm{T}}\mathbf{R}_{11}^{-1}\mathbf{R}_{12} \end{bmatrix}$$

The last column $\mathrm{R}_{11}^{-1}\mathrm{R}_{12}$ contains the standardized coefficient estimates; that is, $\mathbf{b} = \mathbf{R}_{11}^{-1}\mathbf{R}_{12}$. Then the coefficient estimates, except the intercept estimate if there is an intercept in the model, are:

$$\hat{\beta}_j = b_j \sqrt{\frac{S_{yy}}{S_{jj}}}$$

Model selection

The following model selection methods are supported:

- None, in which no selection method is used and effects are force entered into the model. For this method, the singularity tolerance is set to 1e−12 during the sweep operation.

- Forward stepwise, which starts with no effects in the model and adds and removes effects one step at a time until no more can be added or removed according to the stepwise criteria.
- Best subsets, which checks “all possible” models, or at least a larger subset of the possible models than forward stepwise, to choose the best according to the best subsets criterion.

Forward stepwise

The basic idea of the forward stepwise method is to add effects one at a time as long as these additions are worthy. After an effect has been added, all effects in the current model are checked to see if any of them should be removed. Then the process continues until a stopping criterion is met. The traditional criterion for effect entry and removal is based on their *F*-statistics and corresponding *p*-values, which are compared with some specified entry and removal significance levels; however, these statistics may not actually follow an *F* distribution so the results might be questionable. Hence the following additional criteria for effect entry and removal are offered:

- Maximum adjusted R^2;
- Minimum corrected Akaike information criterion (AICC); and
- Minimum average squared error (ASE) over the overfit prevention data

Candidate statistics

Some additional notations are needed describe the addition or removal of a continuous effect X_{j} or categorical effect $\{X_{j_s}\}_{s=1}^{\ell}$, where ℓ is the number of categories.

ℓ^* The number of non-redundant parameters of the eligible effect $\mathbf{X}_{\mathrm{j}}$ or $\{X_{j_s}\}_{s=1}^{\ell}$.

p^c The number of non-redundant parameters in the current model (including the intercept).

p^r The number of non-redundant parameters in the resulting model (including the intercept). Note that $p^r = \begin{cases} p^c + \ell^* \text{for entering an effect} \\ p^c - \ell^* \text{for removing an effect} \end{cases}$

SSe_p The weighted residual sum of squares for the current model.

$SSe_{p+\ell}$ The weighted residual sum of squares for the resulting model after entering the effect.

$SSe_{p-\ell}$ The weighted residual sum of squares for the resulting model after removing the effect.

r_{yy} The last diagonal element in the current **R** matrix.

$\tilde{r}_{yy}$ The last diagonal element in the resulting $\tilde{\mathbf{R}}$ matrix.

F statistics. The *F* statistics for entering or removing an effect from the current model are:

$$F_{enter_j} = \frac{\left(SSe_p - SSe_{p+\ell}\right)/\ell^*}{SSe_{p+\ell}/\left(N - p^r\right)} = \frac{\left(r_{yy} - \tilde{r}_{yy}\right)\left(N - p^r\right)}{\tilde{r}_{yy} \times \ell^*}$$

$$F_{remove_j} = \frac{\left(SSe_{p-\ell} - SSe_p\right)/\ell^*}{SSe_p/(N - p^c)} = \frac{(\tilde{r}_{yy} - r_{yy})(N - p^c)}{r_{yy} \times \ell^*}$$

and their corresponding *p*-values are:

$$p_{enter_j} = P\big(F_{\ell^*,N-p^r} \geq F_{enter_j}\big) = 1 - P\big(F_{\ell^*,N-p^r} \leq F_{enter_j}\big)$$

$$p_{remove_j} = P\big(F_{\ell^*,N-p^c} \geq F_{remove_j}\big) = 1 - P\big(F_{\ell^*,N-p^c} \leq F_{remove_j}\big)$$

Adjusted R-squared. The adjusted R^2 value for entering or removing an effect from the current model is:

$$\text{adj.}R^2 = 1 - \frac{(N-1)\tilde{r}_{yy}}{N-p^r}$$

Corrected Akaike Information Criterion (AICC). The AICC value for entering or removing an effect from the current model is:

$$AICC = N \ln\left(\frac{(N-1)\,S_{yy} \times \tilde{r}_{yy}}{N}\right) + \frac{2p^r N}{N-p^r-1}$$

Average Squared Error (ASE). The ASE value for entering or removing an effect from the current model is:

$$ASE = \frac{1}{\sum_{t=1}^{T} f_t} \sum_{t=1}^{T} w_t (y_t - \hat{y}_t)^2$$

where $\hat{y}_t = \mathrm{x}_t\hat{\beta}$ are the predicted values of y_t and *T* is the number of distinct testing cases in the overfit prevention set.

The Selection Process

There are slight variations in the selection process, depending upon the model selection criterion:

- The *F* statistic criterion is to select an effect for entry (removal) with the minimum (maximum) *p*-value and continue doing it until the *p*-values of all candidates for entry (removal) are equal to or greater than (less than) a specified significance level.
- The other three criteria are to compare the statistic (adjusted R^2, AICC or ASE) of the resulting model after entering (removing) an effect with that of the current model. Selection stops at a local optimal value (a maximum for the adjusted R^2 criterion and a minimum for the AICC and ASE).

The following additional definitions are needed for the selection process:

FLAG	A $p^e \times 1$ index vector which records the status of each effect. $FLAG_{\mathrm{i}}$ = 1 means the effect *i* is in the current model, $FLAG_{\mathrm{i}}$ = 0 means it is not. $\lvert\{i \vert FLAG_i = 1\}\rvert$ denotes the number of effects with $FLAG_{\mathrm{i}}$ = 1.
MAXSTEP	The maximum number of iteration steps. The default value is $3 \times p^e$.
MAXEFFECT	The maximum number of effects (excluding intercept if exists). The default value is p^e.

P_{in} The significance level for effect entry when the F-statistic criterion is used. The default is 0.05.

P_{out} The significance level for effect removal when the F statistic criterion is used. The default is 0.1.

ΔF The F statistic change. It is F_{enter_j} or F_{remove_j} for entering or removing an effect X_{j} (here X_{j} could represent continuous or categorical for simpler notation).

$p_{\Delta F}$ The corresponding p-value for ΔF.

MSC_{current} The adjusted R^2, AICC, or ASE value for the current model.

1. Set $\{FLAG_i\}_{i=1}^{p^e} = 0$ and *iter* = 0. The initial model is $\hat{y} = \overline{y}$. If the adjusted R^2, AICC, or ASE criterion is used, compute the statistic for the initial model and denote it as MSC_{current}.

2. If $\{i|FLAG_i = 0\} \neq 0$, *iter* ≤ *MAXSTEP* and $|\{i|FLAG_i = 1\}| < MAXEFFECT$, go to the next step; otherwise stop and output the current model .

3. Based on the current model, for every effect *j* eligible for entry (see Condition below),

   If FC (the F statistic criterion) is used, compute F_{enter_j} and p_{enter_j};

   If MSC (the adjusted R^2, AICC, or ASE criterion) is used, compute MSC_{j}.

4. If FC is used, choose the effect $X_{j^*}, j^* = arg\,\min_j \left\{p_{enter_j}\right\}$ and if $p_{enter_{j^*}} < P_{\text{in}}$, enter X_{j^*} to the current model.

   If MSC is used, choose the effect $X_{j^*}, j^* = arg\,\min_j \{MSC_j\}$ and if $MSC_{j^*} < MSC_{current}$, enter X_{j^*} to the current model. (For the adjusted R^2 criterion, replace min with max and reverse the inequality)

   If the inequality is not satisfied, stop and output the current model.

5. If the model with the new effect is the same as any previously obtained model, stop and output the current model; otherwise update the current model by doing the sweep operation on corresponding row(s) and column(s) associated with X_{j^*} in the current **R** matrix. Set $FLAG_{j^*} = 1$ and *iter* = *iter* + 1.

   If FC is used, let $\Delta F = F_{enter_{j^*}}$ and $p_{\Delta F} = p_{enter_{j^*}}$;

   If MSC is used, let $MSC_{current} = MSC_{j^*}$.

6. For every effect *k* in the current model; that is, $FLAG_k = 1, \forall k$,

   If FC is used, compute F_{remove_k} and p_{remove_k};

   If MSC is used, compute MSC_{k}.

7. If FC is used, choose the effect $X_{k^*}, k^* = arg\,\max_k \{p_{remove_k}\}$ and if $p_{remove_{k^*}} > P_{\text{out}}$, remove X_{k^*} from the current model.

   If MSC is used, choose the effect $X_{k^*}, k^* = arg\,\min_k \{MSC_k\}$ and if $MSC_{j^*} < MSC_{current}$, remove X_{k^*} from the current model. (For the adjusted R^2 criterion, replace min with max and reverse the inequality)

   If the inequality is met, go to the next step; otherwise go back to step 2.

8. If the model with the effect removed is the same as any previously obtained model, stop and output the current model; otherwise update the current model by doing the sweep operation on corresponding row(s) and column(s) associated with X_{j^*} in the current **R** matrix. Set $FLAG_{j^*} = 0$ and *iter* = *iter* + 1.

   If FC is used, let $\Delta F = F_{remove_{k^*}}$ and $p_{\Delta F} = p_{remove_{k^*}}$;

   If AC is used, let $AICC_{current} = AICC_{k^*}$. Then go back to step 6.

Condition. In order for effect *j* to be eligible for entry into the model, the following conditions must be met:

For continuous a effect X_{j} , $r_{jj} \geq t$; (*t* is the singularity tolerance with a value of 1e−4)

For categorical effect $\{X_{j_s}\}_{s=1}^{\ell}$, $max\,\{r_{j_1j_1}, r_{j_2j_2}, \ldots, r_{j_\ell j_\ell}\} \geq t$;

where *t* is the singularity tolerance, and r_{jj} and $r_{j_sj_s}$, $s = 1, \ldots, \ell$, are diagonal elements in the current **R** matrix (before entering).

For each continuous effect X_{k} that is currently in the model, $\tilde{r}_{kk}t \leq 1$.

For each categorical effect $\{X_{k_s}\}_{s=1}^{\ell'}$ with ℓ' levels that is currently in the model, $max\left\{\tilde{r}_{k_1k_1}, \tilde{r}_{k_2k_2}, \ldots, \tilde{r}_{k_{\ell'}k_{\ell'}}\right\} t \leq 1$.

where $\tilde{r}_{kk}$ and $\tilde{r}_{k_sk_s}$, $s = 1, \ldots, \ell'$, are diagonal elements in the resulting **R** matrix; that is, the results after doing the sweep operation on corresponding row(s) and column(s) associated with X_{k} or $\{X_{k_s}\}_{s=1}^{\ell'}$ in the current R matrix. The above condition is imposed so that entry of the effect does not reduce the tolerance of other effects already in the model to unacceptable levels.

Best subsets

Stepwise methods search fewer combinations of sub-models and rarely select the best one, so another option is to check all possible models and select the “best” based upon some criterion. The available criteria are the maximum adjusted R^2, minimum AICC, and minimum ASE over the overfit prevention set.

Since there are p^e free effects, we do an exhaustive search over 2^{p^e} models, which include intercept-only model ($\hat{y} = \bar{y}$). Because the number of calculations increases exponentially with p^e, it is important to have an efficient algorithm for carrying out the necessary computations. However, if p^e is too large, it may not be practical to check all of the possible models.

We divide the problem into 2 tiers in terms of the number of effects:

- when $p^e \leq 20$, we search all possible subsets
- when $p^e > 20$, we apply a hybrid method which combines the forward stepwise method and the all possible subsets method.

Searching All Possible Subsets

An efficient method that minimizes the number of sweep operations on the **R** matrix (Schatzoff 1968), is applied to traverse all the models and outlined as follows:

Each sweep step(s) on an effect results in a model. So 2^{p^e} models can be obtained through a sequence of exactly 2^{p^e} sweeps on effects. Assuming that the all possible models on $p^e - 1$ effects can be obtained in a sequence S_{p^e-1} of exactly 2^{p^e-1} sweeps on the first 2^{p^e-1} pivotal effects, and sweeping on the last effect will produce a new model which adds the last effect to the model produced by the sequence S_{p^e-1}, then repeating the sequence S_{p^e-1} will produce another 2^{p^e-1} distinct models (including the last effect). It is a recursive algorithm for constructing the sequence; that is, $S_{p^e} = \left(\underline{S_{p^e-1}}, k, \underline{S_{p^e-1}}\right) = \left(\underline{S_{p^e-2}, k-1, S_{p^e-2}}, k, \underline{S_{p^e-2}, k-1, S_{p^e-2}}\right) = \ldots$, and so on.

The sequence of models produced is demonstrated in the following table:

k	*S*k	Sequence of models produced
0	0	Only intercept
1	1	(1)
2	121	(1),(12),(2)
3	1213121	(1),(12),(2),(23),(123),(13),(3)
4	121312141213121	(1),(12),(2),(23),(123),(13),(3),(34),(134),(1234),(234),(24),(124),(14),(4)
...	...	...
p^e	$S_{p^e-1}, p^e, S_{p^e-1}$	All 2^{p^e} models including the intercept model.

The second column indicates the indexes of effects which are pivoted on. Each parenthesis in the third column represents a regression model. The numbers in the parentheses indicate the effects which are included in that model.

Hybrid Method

If p^e>20, we apply a hybrid method by combining the forward stepwise method with the all possible subsets method as follows:

Select the effects using the forward stepwise method with the same criterion chosen for best subsets. Say that p^s is the number of effects chosen by the forward stepwise method.

Apply one of the following approaches, depending on the value of p^s, as follows:

- If $p^s \leq 20$, do an exhaustive search of all possible subsets on these selected effects, as described above.
- If $20 < p^s \leq 40$, select $p^s - 20$ effects based on the *p*-values of type III sum of squares tests from all p^s effects (see ANOVA in “Model evaluation ” on p. 255) and enter them into the model, then do an exhaustive search of the remaining 20 effects via the method described above.
- If $40 < p^s$, do nothing and assume the best model is the one with these p^s effects (with a warning message that the selected model is based on the forward stepwise method).

Model evaluation

The following output statistics are available.

ANOVA

Weighted total sum of squares

$$SS_t = \sum_{i=1}^{n} w_i (y_i - \overline{y})^2 = (N-1)\, S_{yy} \ \text{ with d.f. } = df_t = N-1$$

where d.f. means degrees of freedom. It is called “SS (sum of squares) for Corrected Total”.

Weighted residual sum of squares

$$SS_e = \sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2 = \tilde{r}_{yy}\, (N-1)\, S_{yy}$$

with d.f. = $df_e = N - p^c$. It is also called “SS for Error”.

Weighted regression sum of squares

$$SS_r = \sum_{i=1}^{n} w_i (\hat{y}_i - \overline{y})^2 = (1 - \tilde{r}_{yy})\, (N-1)\, S_{yy} = SS_t - SS_e$$

with d.f. = $df_r = p^*$. It is called “SS for Corrected Model” if there is an intercept.

Regression mean square error

$$SS_r / df_r$$

Residual mean square error

$$SS_e / df_e$$

F statistic for corrected model

$$F = \frac{SS_r / df_r}{SS_e / df_e} = \frac{SS_r \cdot df_e}{SS_e \cdot df_r}$$

which follows an F distribution with degrees of freedom df_r and df_e, and the corresponding p-value can be calculated accordingly.

Type III sum of squares for each effect

To compute type III SS for the effect j, $j = 1, \ldots, p^e$, the type III test matrix $\mathbf{L}_\mathrm{i}$ needs to be constructed first. Construction of $\mathbf{L}_\mathrm{i}$ is based on the generating matrix $\mathrm{H}_\omega = \left(\mathrm{X}^\mathrm{T}\mathrm{DX}\right)^{-}\mathrm{X}^\mathrm{T}\mathrm{DX}$, where $\mathrm{D} = diag\left(g_1, \ldots, g_n\right)$, such that $\mathbf{L}_\mathrm{i}\beta$ is estimable. It involves parameters only for the given effect and the effects containing the given effect. For type III analysis, $\mathbf{L}_\mathrm{i}$ doesn't depend on the order of effects specified in the model. If such a matrix cannot be constructed, the effect is not testable. For each effect j, the type III SS is calculated as follows

$$\mathbf{S}_j = \hat{\beta}^\mathrm{T}\mathbf{L}_j^\mathrm{T}\left(\mathbf{L}_j\mathbf{G}\mathbf{L}_j^\mathrm{T}\right)^{-1}\mathbf{L}_j\hat{\beta}$$

where $\mathbf{G} = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-}$.

F statistic for each effect

The SS for the effect j is also used to compute the F statistic for the hypothesis test H_0: $\mathbf{L}_\mathrm{i}\beta = \mathbf{0}$ as follows:

$$F_j = \frac{\mathbf{S}_j/r_j}{SS_e/df_e}$$

where r_j is the full row rank of $\mathbf{L}_i$. It follows an F distribution with degrees of freedom r_i and df_e, then the p-values can be calculated accordingly.

Model summary

Adjusted R square

$$\text{adj.}R^2 = 1 - \frac{SS_e/df_e}{SS_t/df_t} = R^2 - \frac{\left(1 - R^2\right)p^*}{df_e} = 1 - \frac{df_t \times \tilde{r}_{yy}}{df_e}$$

where

$$R^2 = \frac{SS_r}{SS_t} = 1 - \frac{SS_e}{SS_t} = 1 - \tilde{r}_{yy}.$$

Model information criteria

Corrected Akaike information criterion (AICC)

$$AICC = N\ln\left(\frac{SS_e}{N}\right) + \frac{2p^cN}{N - p^c - 1}$$

Coefficients and statistical inference

After the model selection process, we can get the coefficients and related statistics from the swept correlation matrix. The following statistics are computed based on the **R** matrix.

Unstandardized coefficient estimates

$$\hat{\beta}_j = b_j \sqrt{\frac{S_{yy}}{S_{jj}}} = \tilde{r}_{jy} \sqrt{\frac{S_{yy}}{S_{jj}}}$$

for $j = 1, \cdots, p^*$.

Standard errors of regression coefficients

The standard error of $\hat{\beta}_j$ is

$$\hat{\sigma}_{\hat{\beta}_j} = \sqrt{var\left(\hat{\beta}_j\right)} = \sqrt{\frac{\tilde{r}_{jj}\tilde{r}_{yy}S_{yy}}{S_{jj}df_e}}$$

Intercept estimation

The intercept is estimated by all other parameters in the model as

$$\hat{\beta}_0 = \overline{y} - \sum_{j=1}^{p} \hat{\beta}_j \overline{X}_j$$

The standard error of $\hat{\beta}_0$ is estimated by

$$\hat{\sigma}_{\hat{\beta}_0} = \sqrt{\hat{\sigma}^2_{\hat{\beta}_0}}$$

where

$$\begin{aligned} \hat{\sigma}^2_{\hat{\beta}_0} &= \tfrac{(N-1)\tilde{r}_{yy}S_{yy}}{N(N-p^*-1)} + \textstyle\sum_{j=1}^{p} \overline{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \overline{X}_k \overline{X}_j cov\left(\hat{\beta}_k, \hat{\beta}_j\right) \\ &= \tfrac{SS_e}{N \times df_e} + \textstyle\sum_{j=1}^{p} \overline{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \overline{X}_k \overline{X}_j \tfrac{\tilde{r}_{kj} \times SS_e}{\sqrt{S_{kk}S_{jj}} \times (N-1)df_e}. \end{aligned}$$

$\hat{\sigma}^2_{\hat{\beta}_0} = \frac{(N-1)\tilde{r}_{yy}S_{yy}}{N(N-p^*-1)} + \sum_{j=1}^{p} \overline{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \overline{X}_k \overline{X}_j cov\left(\hat{\beta}_k, \hat{\beta}_j\right)$ and $cov\left(\hat{\beta}_k, \hat{\beta}_j\right)$ is the *k*th row and *j*th column element in the parameter estimates covariance matrix.

t statistics for regression coefficients

$$t = \frac{\hat{\beta}_j}{\hat{\sigma}_{\hat{\beta}_j}} = \tilde{r}_{jy} \sqrt{\frac{df_e}{\tilde{r}_{yy}\tilde{r}_{jj}}}$$

for $j = 1, \cdots, p^*$, with degrees of freedom df_e and the *p*-value can be calculated accordingly.

100(1−α)% confidence intervals

$$\hat{\beta}_j \pm \hat{\sigma}_{\hat{\beta}_j} \times t_{\alpha/2, df_e}$$

Note: For redundant parameters, the coefficient estimates are set to zero and standard errors, t statistics, and confidence intervals are set to missing values.

Scoring

Predicted values

$$\hat{y}_k = \sum_{i=0}^{p} x_{ki}\hat{\beta}_i, k = 1, \ldots, n.$$

Diagnostics

The following values are computed to produce various diagnostic charts and tables.

Residuals

$$e_k = y_k - \hat{y}_k$$

Studentized residuals

This is the ratio of the residual to its standard error.

$$SRES_k = \frac{e_k}{s\sqrt{\frac{(1-h_k)}{g_k}}}$$

where *s* is the square root of the mean square error; that is, $s = \sqrt{SS_e/df_e}$, and h_k is the leverage value for the *k*th case (see below).

Cook's distance

$$COOK_k = \frac{e_k^2 h_k g_k}{s^2 (1-h_k)^2 p^c}$$

where the "leverage"

$$h_k = g_k \mathbf{x}_k \mathrm{G} \mathbf{x}_k^{\mathrm{T}}$$

is the *k*th diagonal element of the hat matrix

$$\mathrm{H} = \mathrm{W}^{1/2}\mathrm{X}\left(\mathrm{X}^{\mathrm{T}}\mathrm{W}\mathrm{X}\right)^{-}\mathrm{X}^{\mathrm{T}}\mathrm{W}^{1/2} = \mathrm{W}^{1/2}\mathrm{X}\mathbf{G}\mathrm{X}^{\mathrm{T}}\mathrm{W}^{1/2}$$

A record with Cook's distance larger than $\frac{4}{N-p^c}$ is considered influential (Fox, 1997).

Predictor importance

We use the leave-one-out method to compute the predictor importance, based on the residual sum of squares (SSe) by removing one predictor at a time from the final full model.

If the final full model contains *p* predictors, $X_1, X_2, \cdots, X_p$, then the predictor importance can be calculated as follows:

1. *i*=1
2. If *i* > *p*, go to step 5.
3. Do a sweep operation on the corresponding row(s) and column(s) associated with X_i in the $\tilde{\mathbf{R}}$ matrix of the full final model.
4. Get the last diagonal element in the current $\tilde{\mathbf{R}}$ and denote it $\tilde{r}_{yy}^{(i)}$. Then the predictor importance of X_i is $VI_i = \left(\tilde{r}_{yy}^{(i)} - \tilde{r}_{yy}\right)(N-1)\,SS_{yy}$. Let *i* = *i* + 1, and go to step 2.
5. Compute the normalized predictor importance of X_i:

   $NormVI_i = \frac{VI_i}{\Sigma_{i=1}^{p} VI_i}$

References

Belsley, D. A., E. Kuh, and R. E. Welsch. 1980. *Regression diagnostics: Identifying influential data and sources of collinearity*. New York: John Wiley and Sons.

Dempster, A. P. 1969. *Elements of Continuous Multivariate Analysis*. Reading, MA: Addison-Wesley.

Fox, J. 1997. *Applied Regression Analysis, Linear Models, and Related Methods*. Thousand Oaks, CA: SAGE Publications, Inc..

Fox, J., and G. Monette. 1992. Generalized collinearity diagnostics. *Journal of the American Statistical Association*, 87, 178–183.

Schatzoff, M., R. Tsao, and S. Fienberg. 1968. Efficient computing of all possible regressions. *Technometrics*, 10, 769–779.

Velleman, P. F., and R. E. Welsch. 1981. Efficient computing of regression diagnostics. *American Statistician*, 35, 234–242.

Linear Regression Algorithms

Overview

This procedure performs ordinary least squares multiple linear regression with four methods for entry and removal of variables (Neter, Wasserman, and Kutner, 1990).

Primary Calculations

Notation

The following notation is used throughout this chapter unless otherwise stated:

y_i	Output field for record *i* with variance $\frac{\sigma^2}{g_i}$
c_i	Case weight for record *i*; in IBM® SPSS® Modeler, $c_i \equiv 1$
g_i	Regression weight for record i; $g_i = 1$ if regression weight is not specified
l	Number of distinct records
w_i	$c_i \cdot g_i$
W	The sum of weights across records, $\sum_{i=1}^{l} w_i$
p	Number of input fields
C	Sum of case weights, $\sum_{i=1}^{l} c_i$
x_{ki}	The value of the *k*th input field for record *i*
$\overline{X}_k$	Sample mean for the *k*th input field, $\frac{\sum_{i=1}^{l} w_i x_{ki}}{W}$
$\overline{Y}$	Sample mean for the output field, $\frac{\sum_{i=1}^{l} w_i y_i}{W}$
S_{kj}	Sample covariance for input fields X_k and X_j
S_{yy}	Sample variance for output field *Y*
S_{ky}	Sample covariance for X_k and *Y*
p^*	Number of coefficients in the model. $p^* = p$ if the intercept is not included; otherwise $p^* = p + 1$
$\mathbf{R}$	Sample correlation matrix for $X_1 ... X_p$ and *Y*

Model Parameters

The summary statistics $\overline{X}_i$ and covariance S_{ij} are computed using provisional means algorithms to update the values as each record is read:

$$\overline{X}_{i(k)} = \overline{X}_{i(k-1)} + \left(x_{ik} - \overline{X}_{i(k-1)}\right)\frac{w_k}{W_k}$$

and

$$S_{ij} = \frac{C_{ij}}{C-1}$$

where, if the intercept is included,

$$C_{ij(k)} = C_{ij(k-1)} + \left(x_{ik} - \overline{X}_{i(k-1)}\right)\left(x_{jk} - \overline{X}_{j(k-1)}\right)\left(w_k - \frac{w_k^2}{W_k}\right)$$

or if the intercept is not included,

$$C_{ij(k)} = C_{ij(k-1)} + w_k x_{ik} x_{jk}$$

where W_k is the cumulative weight up to record *k*, and $\overline{X}_{i(k)}$ is the estimate of $\overline{X}_i$ up to record *k*.

For a regression model of the form

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + ... + \beta_p X_{pi} + e_i$$

sweep operations are used to compute the least squares estimates **b** of β and the associated regression statistics (Dempster, 1969). The sweeping starts with the correlation matrix **R**,

$$\mathbf{R} = \begin{bmatrix} r_{11} & ... & r_{1p} & r_{1y} \\ r_{21} & ... & r_{2p} & r_{2y} \\ . & ... & . & . \\ r_{y1} & ... & r_{yp} & r_{yy} \end{bmatrix}$$

where

$$r_{kj} = \frac{S_{kj}}{\sqrt{S_{kk}S_{jj}}}$$

and

$$r_{yk} = r_{ky} = \frac{S_{ky}}{\sqrt{S_{kk}S_{yy}}}$$

Let $\tilde{\mathbf{R}}$ be the new matrix produced by sweeping on the *k*th row and column of **R**. The elements of $\tilde{\mathbf{R}}$ are

$$\tilde{r}_{kk} = \frac{1}{r_{kk}}$$

$$\tilde{r}_{ik} = \frac{r_{ik}}{r_{kk}}, i \neq k$$

$$\tilde{r}_{kj} = \frac{r_{kj}}{r_{kk}}, j \neq k$$

and

$$\tilde{r}_{ij} = \frac{r_{ij}r_{kk} - r_{ik}r_{kj}}{r_{kk}}, i \neq k, j \neq k$$

If the above sweep operations are repeatedly applied to each row of $\mathbf{R}_{11}$ in

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{pmatrix}$$

where $\mathbf{R}_{11}$ contains the input fields in the equation at the current step, the result is

$$\tilde{\mathbf{R}} = \begin{pmatrix} \mathbf{R}_{11}^{-1} & -\mathbf{R}_{11}^{-1}\mathbf{R}_{12} \\ \mathbf{R}_{21}\mathbf{R}_{11}^{-1} & \mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12} \end{pmatrix}$$

The last row of

$$\mathbf{R}_{21}\mathbf{R}_{11}^{-1}$$

contains the standardized coefficients (also called beta), and

$$\mathbf{R}_{22} - \mathbf{R}_{21}\mathbf{R}_{11}^{-1}\mathbf{R}_{12}$$

can be used to obtain the partial correlations for the variables not in the equation, controlling for the variables already in the equation. Note that this routine is its own inverse; that is, exactly the same operations are performed to remove an input field as to enter it.

The unstandardized coefficient estimates $b_1 ... b_p$ are calculated as

$$b_k = \frac{r_{yk}\sqrt{S_{yy}}}{\sqrt{S_{kk}}}$$

and the intercept b_0, if included in the model, is calculated as

$$b_0 = \overline{y} - \sum_{k=1}^{p} b_k \overline{X}_k$$

Automatic Field Selection

Let r_{ij} be the element in the current swept matrix associated with X_i and X_j. Variables are entered or removed one at a time. X_k is eligible for entry if it is an input field not currently in the model such that

$$r_{kk} \geq t$$

and

$$\left(r_{jj} - \frac{r_{jk}r_{kj}}{r_{kk}} \right) t \leq 1$$

where *t* is the tolerance, with a default value of 0.0001.

The second condition above is imposed so that entry of the variable does not reduce the tolerance of variables already in the model to unacceptable levels.

The F-to-enter value for X_k is computed as

$$F - to - enter_k = \frac{(C - p^* - 1)V_k}{r_{yy} - V_k}$$

with 1 and $C - p^* - 1$ degrees of freedom, where p^* is the number of coefficients currently in the model and

$$V_k = \frac{r_{yk} r_{ky}}{r_{kk}}$$

The F-to-remove value for X_k is computed as

$$F - to - remove_k = \frac{(C - p^*) \left| V_k \right|}{r_{yy}}$$

with 1 and $C - p^*$ degrees of freedom.

Methods for Variable Entry and Removal

Four methods for entry and removal of variables are available. The selection process is repeated until no more independent variables qualify for entry or removal. The algorithms for these four methods are described below.

Enter

The selected input fields are all entered in the model, with no field selection applied.

Stepwise

If there are independent variables currently entered in the model, choose X_k such that $F - to - remove_k$ is minimum. X_k is removed if $F - to - remove_k < F_{out}$ (default = 2.71) or, if probability criteria are used, $P\left(F - to - remove_k\right) > P_{out}$ (default = 0.1). If the inequality does not hold, no variable is removed from the model.

If there are no independent variables currently entered in the model or if no entered variable is to be removed, choose X_k such that $F - to - enter_k$ is maximum. X_k is entered if $F - to - enter_k > F_{in}$ (default = 3.84) or, $P\left(F - to - enter_k\right) < P_{in}$ (default = 0.05). If the inequality does not hold, no variable is entered.

At each step, all eligible variables are considered for removal and entry.

Forward

This procedure is the entry phase of the stepwise procedure.

Backward

This procedure starts with all input fields in the model and applies the removal phase of the stepwise procedure.

Blank Handling

By default, a case that has a missing value for any input or output field is deleted from the computation of the correlation matrix on which all consequent computations are based. If the Only use complete records option is deselected, each correlation in the correlation matrix **R** is computed based on records with complete data for the two fields associated with the correlation, regardless of missing values on other fields. For some datasets, this approach can lead to a non-positive definite **R** matrix, so that the model cannot be estimated.

Secondary Calculations

Model Summary Statistics

The multiple correlation coefficient R is calculated as

$$R = \sqrt{1 - r_{yy}}$$

R-square, the proportion of variance in the output field accounted for by the input fields, is calculated as

$$R^2 = 1 - r_{yy}$$

The adjusted R-square, which takes the complexity of the model relative to the size of the training data into account, is calculated as

$$R^2_{adj} = R^2 - \frac{\left(1 - R^2\right)p}{C - p^*}$$

Field Statistics and Other Calculations

The statistics shown in the advanced output for the regression equation node are calculated in the same manner as in the REGRESSION procedure in IBM® SPSS® Statistics. For more details, see the SPSS Statistics Regression algorithm document, available at *http://www.ibm.com/support*.

Generated Model/Scoring

Predicted Values

The predicted value for a new record is calculated as

$$\hat{y} = b_0 + \sum_{i=1}^{p} b_i X_i$$

Blank Handling

Records with missing values for any input field in the final model cannot be scored, and are assigned a predicted value of $null$.

Logistic Regression Algorithms

Logistic Regression Models

Logistic regression is a well-established statistical method for predicting binomial or multinomial outcomes. IBM® SPSS® Modeler now offers two distinct algorithms for logistic regression modeling:

Multinomial Logistic. This is the original logistic regression algorithm used in SPSS Modeler, introduced in version 6.0. It can produce models when the target field is a set field with more than two possible values. See below for more information. It can also produce models for flag or binary outcomes, though it doesn't give the same level of statistical detail for such models as the newer binomial logistic algorithm.

Binomial Logistic. This algorithm, introduced in SPSS Modeler 11, is limited to models where the target field is a flag, or binary field. This algorithm provides some enhanced statistical output, relative to the output of the multinomial algorithm, and is less susceptible to problems when the number of cells (unique combinations of predictor values) is large relative to the number of records. For more information, see the topic "Binomial Logistic Regression" on p. 279.

For models with a flag output field, selection of a logistic algorithm is controlled in the modeling node by the Procedure option.

Multinomial Logistic Regression

The purpose of the Multinomial Logistic Regression procedure is to model the dependence of a nominal (symbolic) output field on a set of symbolic and/or numeric predictor (input) fields.

Primary Calculations

Field Encoding

In logistic regression, each symbolic (set) field is recoded as a group of numeric fields, with one numeric field for each category or value of the original field, except the last category, which is defined as a reference category. For each record, the value of the derived field corresponding to the category of the record is set to 1.0, and all of the other derived field values are set to 0.0. For records which have the value of the reference category, all derived fields are set to 0.0. Such derived fields are sometimes called **dummy fields**, and this recoding is called **dummy coding**.

For example, consider the following data, where *x* is a symbolic field with possible values A, B, and C:

Record #	X	X_1'	X_2'
1	B	0	1
2	A	1	0
3	C	0	0

In this data, the original set field x is recoded into two derived fields x_1' and x_2'. x_1' is an indicator for category A, and x_2' is an indicator for category B. The last category, category C, is the reference category; records belonging to this category have both x_1' and x_2' set to 0.0.

Notation

The following notation is used throughout this chapter unless otherwise stated:

Y	The output field, which takes integer values from 1 to *J*.
J	The number of categories of the output field.
m	The number of subpopulations.
$\mathbf{X}^A$	$m \times p^A$ matrix with vector-element x_i^A , the observed values at the *i*th subpopulation, determined by the input fields specified in the command.
$\mathbf{X}$	$m \times p$ matrix with vector-element x_i^A, the observed values of the location model's input fields at the *i*th subpopulation.
n_{ij}	The sum of frequency weights of the observations that belong to the cell corresponding to $Y = j$ at subpopulation *i*.
N	The sum of all n_{ij}'s.
π_{ij}	The cell probability corresponding to $Y = j$ at subpopulation *i*.
$\log\left(\pi_{ij}/\pi_{ik}\right)$	The logit of response category *j* relative to response category *k*.
$\beta_j = (\beta_{j1}, ..., \beta_{jp})'$	$p \times 1$ vector of unknown parameters in the *j*th logit (that is, logit of response category *j* to response category *J*).
p	Number of parameters in each logit. $p \geq 1$.
p_j^{nr}	Number of non-redundant parameters in logit *j* after maximum likelihood estimation. $p \geq p_j^{nr} \geq 0$.
p^{nr}	The total number of non-redundant parameters after maximum likelihood estimation. $p^{nr} = \sum_{j=1}^{k-1} p_j^{nr}$.
$\mathbf{B} = \left({\beta'}_1, ..., {\beta'}_{J-1}\right)'$	$(k-1)p \times 1$ vector of unknown parameters in the model.
$\hat{\mathbf{B}} = \left(\hat{\beta}'_1, ..., \hat{\beta}'_{J-1}\right)'$	The maximum likelihood estimate of $\mathbf{B}$.
$\hat{\pi}_{ij}$	The maximum likelihood estimate of π_{ij}.

Data Aggregation

Observations are aggregated by the definition of subpopulations. Subpopulations are defined by the cross-classifications of the set of input fields.

Let n_i be the marginal count of subpopulation *i*,

$$n_i = \sum_{j=1}^{k} n_{ij}$$

If there is no observation for the cell of $Y = j$ at subpopulation i, it is assumed that $n_{ij} = 0$, provided that $n_i \neq 0$. A non-negative scalar $\delta \in [0, 1)$ may be added to any zero cell (that is, cell with $n_{ij} = 0$) if its marginal count n_i is nonzero. The value of δ is zero by default.

Generalized Logit Model

In a generalized logit model, the probability π_{ij} of response category *j* at subpopulation *i* is

$$\pi_{ij} = \frac{\exp\left(\mathbf{x'}_i\beta_j\right)}{1 + \sum_{k=1}^{J-1} \exp\left(\mathbf{x'}_i\beta_k\right)}$$

where the last category *J* is assumed to be the reference category.

In terms of logits, the model can be expressed as

$$\log\left(\frac{\pi_{ij}}{\pi_{iJ}}\right) = \mathbf{x'}_i\beta_j$$

for *j* = 1, …, *J*-1.

When *J* = 2, this model is equivalent to the binary logistic regression model. Thus, the above model can be thought of as an extension of the binary logistic regression model from binary response to polytomous nominal response.

Log-Likelihood

The log-likelihood of the model is given by

$$\begin{aligned} l(\mathbf{B}) \qquad\qquad &= \sum_{i=1}^{m}\sum_{j=1}^{J} n_{ij}\log\left(\pi_{ij}\right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{J} n_{ij}\log\left(\frac{\exp\left(\mathbf{x'}_i\beta_j\right)}{1 + \sum_{k=1}^{J-1}\exp\left(\mathbf{x'}_i\beta_l\right)}\right) \end{aligned}$$

A constant that is independent of parameters has been excluded here. The value of the constant is $c = \sum_{i=1}^{m} \log\left(n_i!/(n_{i1}!...n_{iJ}!)\right)$.

Model Parameters

Derivatives of the Log-Likelihood

For any *j* = 1, …, *J*-1, *s* = 1, …, *p*, the first derivative of *l* with respect to β_{js} is

$$\frac{\partial l}{\partial \beta_{js}} = \sum_{i=1}^{m} x_{is}(n_{ij} - n_i\pi_{ij}).$$

For any j, j'= 1, …, J-1 and s, t = 1, …, p, the second derivative of l with respect to β_{js} and $\beta_{j't}$ is

$$\frac{\partial^2 l}{\partial\beta_{js}\partial\beta_{j't}} = -\sum_{i=1}^{m} n_i x_{is} x_{it} \pi_{ij} (\delta_{jj'} - \pi_{ij'})$$

where $\delta_{jj'} = 1$ if $j = j'$, 0 otherwise.

Maximum Likelihood Estimate

To obtain the maximum likelihood estimate of $\mathbf{B}$, a Newton-Raphson iterative estimation method is used. Notice that this method is the same as Fisher-Scoring iterative estimation method in this model, since the expectation of the second derivative of l with respect to $\mathbf{B}$ is the same as the observed one.

Let $\partial l/\partial\mathbf{B}$ be the $(J-1)p \times 1$ vector of the first derivative of l with respect to $\mathbf{B}$. Moreover, let $\left[\partial^2 l/\partial\mathbf{B}\partial\mathbf{B}\right]$ be the $(J-1)p \times (J-1)p$ matrix of the second derivative of l with respect to $\mathbf{B}$. Notice that $-\left[\partial^2 l/\partial\mathbf{B}\partial\mathbf{B}\right] = \sum_{i=1}^{m} \mathbf{X}_i^* \Delta_i \mathbf{X}_i^{*\prime}$ where Δ_i is a $(J-1) \times (J-1)$ matrix as

$$\Delta_i = n_i\Big(Diag\Big(\pi_i^{(-J)}\Big) - \pi_i^{(-J)}\pi_i^{(-J)\prime}\Big)$$

in which $\pi_i^{(-J)} = (\pi_{i1}, ..., \pi_{iJ-1})'$ and $Diag\Big(\pi_i^{(-J)}\Big)$ is a β_{js} diagonal matrix of $\pi_i^{(-J)}$. Let $B^{(\nu)}$ be the parameter estimate at iteration ν, the parameter estimate $B^{(\nu+1)}$ at iteration $\nu + 1$ is updated as

$$\mathbf{B}^{(\nu+1)} = \mathbf{B}^{(\nu)} + \xi\left(\sum_{i=1}^{m} \mathbf{X}_i^* \Delta_i^{(\nu)} \mathbf{X}_i^{*\prime}\right)\frac{\partial l}{\partial\mathbf{B}^{(\nu)}}$$

and $\xi > 0$ is a stepping scalar such that $l\big(\mathbf{B}^{(\nu+1)}\big) - l\big(\mathbf{B}^{(\nu)}\big) \geq 0$, $\mathbf{X}^*$ is a $(J-1)p \times (J-1)$ matrix of independent vectors,

$$\mathbf{X}_i^* = \begin{pmatrix} \mathbf{x}_i & 0 & ... & 0 \\ 0 & \mathbf{x}i & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & ... & 0 & \mathbf{x}i \end{pmatrix}$$

and $\Delta_i^{(\nu)}$ is Δ_i and $\partial l/\partial\mathbf{B}^{(\nu)}$ is $\partial l/\partial\mathbf{B}$, both evaluated at $\mathbf{B} = \mathbf{B}^{(\nu)}$.

Stepping

Use step-halving method if $l\big(\mathbf{B}^{(\nu+1)}\big) - l\big(\mathbf{B}^{(\nu)}\big) < 0$. Let V be the maximum number of steps in step-halving, the set of values of ξ is $\{1/2^{\nu} : \nu = 0, ..., V-1\}$.

Starting Values of the Parameters

If intercepts are included in the model, set $\beta_j^{(0)} = \Big(\beta_{j1}^{(0)}, 0, ..., 0\Big)'$ where

$$\beta_{j1}^{(0)} = \log\left(\frac{\tilde{\pi}_{ij}}{\tilde{\pi}_{iJ}}\right) = \log\left(\frac{\sum_{i=1}^{m} n_{ij}}{\sum_{i=1}^{m} n_{iJ}}\right)$$

for j = 1, …, J-1.

If intercepts are not included in the model, set

$$\beta_{j}^{(0)} = (0, ..., 0)'$$

for j = 1, …, J-1.

Convergence Criteria

Given two convergence criteria $\epsilon_k > 0$ and $\epsilon_p > 0$, the iteration is considered to be converged if one of the following criteria are satisfied:

1. $\left|l\left(\mathbf{B}^{(\nu+1)}\right) - l\left(\mathbf{B}^{(\nu)}\right)\right| < \epsilon_k$.
2. $\max_i \left|\mathbf{B}_i^{(\nu+1)} - \mathbf{B}_i^{\nu}\right| < \epsilon_p$.
3. The maximum above element in $\partial l / \partial \mathbf{B}^{(\nu+1)}$ is less than $\min\left(\epsilon_k, \epsilon_p\right)$.

Checking for Separation

The algorithm checks for separation in the data starting with iteration ν^{chksep} (20 by default). To check for separation:

1. For each subpopulation i , find $j^* : \hat{\pi}_{ij^*} = \max_j \left(\hat{\pi}_{ij}\right)$.
2. If $n_{ij^*} = n_i$, then there is a perfect prediction for subpopulation i.
3. If all subpopulations have perfect prediction, then there is complete separation. If some patterns have perfect prediction and the Hessian of $\hat{\mathbf{B}}$ is singular, then there is quasi-complete separation.

Blank Handling

All records with missing values for any input or output field are excluded from the estimation of the model.

Secondary Calculations

Model Summary Statistics

Log-Likelihood

Initial model with intercepts. If intercepts are included in the model, the predicted probability for the initial model (that is, the model with intercepts only) is

$$\tilde{\pi}_{ij} = \frac{\sum_{i=1}^{m} n_{ij}}{N}$$

and the value of –2 log-likelihood of the initial model is

$$-2l(\tilde{\pi}) = -2\sum_{i=1}^{m}\sum_{j=1}^{J} n_{ij}\log\left(\tilde{\pi}_{ij}\right).$$

Initial model with no intercepts. If intercepts are not included in the model, the predicted probability for the initial model is

$$\tilde{\pi}_{ij} = \frac{1}{J}$$

and the value of –2 log-likelihood of the initial model is

$$-2l(\tilde{\pi}) = -2N\log\left(\frac{1}{J}\right).$$

Final model. The value of –2 log-likelihood of the final model is

$$-2l(\tilde{\pi}) = -2\sum_{i=1}^{m}\sum_{j=1}^{J} n_{ij}\log\left(\hat{\pi}_{ij}\right).$$

Model Chi-Square

The model chi-square is given by

$$-2l(\tilde{\pi}) - \{-2l(\hat{\pi})\}$$

If the final model includes intercepts, then the initial model is an intercept-only model. Under the null hypothesis that $H_0 : \beta^{intercepts} = \mathbf{0}$, the model chi-square is asymptotically chi-squared distributed with $p^{nr} - (J - 1)$degrees of freedoms.

If the model does not include intercepts, then the initial model is an empty model. Under the null hypothesis that $H_0 : \beta = \mathbf{0}$, the Model Chi-square is asymptotically chi-squared distributed with p^{nr} degrees of freedoms.

Pseudo R-Square Measures

Cox and Snell. Cox and Snell's R^2 is calculated as

$$R_{CS}^2 = 1 - \left(\frac{L(\tilde{\pi})}{L(\hat{\pi})}\right)^{\frac{2}{n}}.$$

Nagelkerke. Nagelkerke's R^2 is calculated as

$$R_N^2 = \frac{R_{CS}^2}{1 - L(\tilde{\pi})^{2/n}}.$$

McFadden. McFadden's R^2 is calculated as

$$R_M^2 = 1 - \left(\frac{l(\hat{\pi})}{l(\tilde{\pi})}\right).$$

Goodness-of-Fit Measures

Pearson. The Pearson goodness-of-fit measure is

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{J}\frac{(n_{ij} - n_i\hat{\pi}_{ij})^2}{n_i\hat{\pi}_{ij}}.$$

Under the null hypothesis, the Pearson goodness-of-fit statistic is asymptotically chi-squared distributed with $m(J-1) - p^{nr}$ degrees of freedom.

Deviance. The deviance goodness-of-fit measure is

$$D = 2\sum_{i=1}^{m}\sum_{j=1}^{J} n_{ij}\log\left(\frac{n_{ij}}{n_i\hat{\pi}_{ij}}\right).$$

Under the null hypothesis, the deviance goodness-of-fit statistic is asymptotically chi-squared distributed with $m(J-1) - p^{nr}$ degrees of freedom.

Field Statistics and Other Calculations

The statistics shown in the advanced output for the logistic equation node are calculated in the same manner as in the NOMREG procedure in IBM® SPSS® Statistics. For more details, see the SPSS Statistics Nomreg algorithm document, available at *http://www.ibm.com/support*.

Stepwise Variable Selection

Several methods are available for selecting independent variables. With the forced entry method, any variable in the variable list is entered into the model. The forward stepwise, backward stepwise, and backward entry methods use either the Wald statistic or the likelihood ratio statistic for variable removal. The forward stepwise, forward entry, and backward stepwise use the score statistic or the likelihood ratio statistic to select variables for entry into the model.

Forward Stepwise (FSTEP)

1. Estimate the parameter and likelihood function for the initial model and let it be our current model.
2. Based on the MLEs of the current model, calculate the score statistic or likelihood ratio statistic for every variable eligible for inclusion and find its significance.
3. Choose the variable with the smallest significance (p-value). If that significance is less than the probability for a variable to enter, then go to step 4; otherwise, stop FSTEP.
4. Update the current model by adding a new variable. If this results in a model which has already been evaluated, stop FSTEP.
5. Calculate the significance for each variable in the current model using LR or Wald's test.
6. Choose the variable with the largest significance. If its significance is less than the probability for variable removal, then go back to step 2. If the current model with the variable deleted is the same as a previous model, stop FSTEP; otherwise go to the next step.
7. Modify the current model by removing the variable with the largest significance from the previous model. Estimate the parameters for the modified model and go back to step 5.

Forward Only (FORWARD)

1. Estimate the parameter and likelihood function for the initial model and let it be our current model.
2. Based on the MLEs of the current model, calculate the score or LR statistic for every variable eligible for inclusion and find its significance.
3. Choose the variable with the smallest significance. If that significance is less than the probability for a variable to enter, then go to step 4; otherwise, stop FORWARD.
4. Update the current model by adding a new variable. If there are no more eligible variable left, stop FORWARD; otherwise, go to step 2.

Backward Stepwise (BSTEP)

1. Estimate the parameters for the full model that includes the final model from previous method and all eligible variables. Only variables listed on the BSTEP variable list are eligible for entry and removal. Let current model be the full model.
2. Based on the MLEs of the current model, calculate the LR or Wald's statistic for every variable in the BSTEP list and find its significance.

3. Choose the variable with the largest significance. If that significance is less than the probability for a variable removal, then go to step 5. If the current model without the variable with the largest significance is the same as the previous model, stop BSTEP; otherwise go to the next step.
4. Modify the current model by removing the variable with the largest significance from the model. Estimate the parameters for the modified model and go back to step 2.
5. Check to see any eligible variable is not in the model. If there is none, stop BSTEP; otherwise, go to the next step.
6. Based on the MLEs of the current model, calculate LR statistic or score statistic for every variable not in the model and find its significance.
7. Choose the variable with the smallest significance. If that significance is less than the probability for the variable entry, then go to the next step; otherwise, stop BSTEP.
8. Add the variable with the smallest significance to the current model. If the model is not the same as any previous models, estimate the parameters for the new model and go back to step 2; otherwise, stop BSTEP.

Backward Only (BACKWARD)

1. Estimate the parameters for the full model that includes all eligible variables. Let the current model be the full model.
2. Based on the MLEs of the current model, calculate the LR or Wald's statistic for all variables eligible for removal and find its significance.
3. Choose the variable with the largest significance. If that significance is less than the probability for a variable removal, then stop BACKWARD; otherwise, go to the next step.
4. Modify the current model by removing the variable with the largest significance from the model. Estimate the parameters for the modified model. If all the variables in the BACKWARD list are removed then stop BACKWARD; otherwise, go back to step 2.

Stepwise Statistics

The statistics used in the stepwise variable selection methods are defined as follows.

Score Function and Information Matrix

The score function for a model with parameter *B* is:

$$U\left(B\right)=\tfrac{\partial l(B)}{\partial B}$$

The (*j*,*s*)th element of the score function can be written as

$$\begin{aligned}\left[U\left(B\right)\right]_{js} &= \tfrac{\partial l(B)}{\partial \beta_{js}} \\ &= \sum_{i=1}^{m} x_{is}\left(n_{ij}-n_i\pi_{ij}\right)\end{aligned}$$

Similarly, elements of the information matrix are given by

$$\begin{aligned}[I(B)]_{js,j't} &= \tfrac{\partial^2 l(B)}{\partial \beta_{js} \partial \beta_{jt}} \\ &= -\sum_{i=1}^{m} n_i x_{is} x_{it} \pi_{ij} \left(\delta_{jj'} - \pi_{ij'}\right)\end{aligned}$$

where $\delta_{jj'} = 1$ if $j = j^{'}$, 0 otherwise.

(Note that π_{ij} in the formula are functions of B)

Block Notations

By partitioning the parameter B into two parts, B_1 and B_2, the score function, information matrix, and inverse information matrix can be written as partitioned matrices:

$$\begin{aligned}U(B_1, B_2) &= \begin{pmatrix} U_1(B_1, B_2) \\ U_2(B_1, B_2) \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial l(B_1,B_2)}{\partial B_1} \\ \frac{\partial l(B_1,B_2)}{\partial B_2} \end{pmatrix}\end{aligned}$$

where $l(B_1, B_2) = l(B)$

$$\begin{aligned}I(B) &= I(B_1, B_2) \\ &= \begin{pmatrix} I_{11}(B_1, B_2) & I_{12}(B_1, B_2) \\ I_{21}(B_1, B_2) & I_{22}(B_1, B_2) \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial^2 l(B_1,B_2)}{\partial B_1 \partial B_1} & \frac{\partial^2 l(B_1,B_2)}{\partial B_1 \partial B_2} \\ \frac{\partial^2 l(B_1,B_2)}{\partial B_2 \partial B_1} & \frac{\partial^2 l(B_1,B_2)}{\partial B_2 \partial B_2} \end{pmatrix}\end{aligned}$$

$$J(B) = I(B_1, B_2)^{-} = \begin{pmatrix} J_{11}(B_1, B_2) & J_{12}(B_1, B_2) \\ J_{21}(B_1, B_2) & J_{22}(B_1, B_2) \end{pmatrix}$$

where

$$\begin{aligned}J_{11} &= I_{11}^{-} + I_{11}^{-} I_{12} J_{22} I_{21} I_{11}^{-} \\ J_{12} &= -I_{11}^{-} I_{12} J_{22} \\ J_{21} &= J_{12}^{T} \\ J_{22} &= \left[I_{22} - I_{21} I_{11}^{-} I_{12}\right]^{-}\end{aligned}$$

Typically, B_1 and B_2 are parameters corresponding to two different sets of effects. The dimensions of the 1st and 2nd partition in U, I and J are equal to the numbers of parameters in B_1 and B_2 respectively.

Score Test

Suppose a base model with parameter vector B_{base} with the corresponding maximum likelihood estimate $\hat{B}_{base}$. We are interested in testing the significance of an extra effect E if it is added to the base model. For convenience, we will call the model with effect E the augmented model. Let B_E be the vector of extra parameters associated with the effect E, then the hypothesis can be written as

$H_0 : B_E = 0$ v.s. $H_1 : B_E \neq 0$

Using the block notations, the score function, information matrix and inverse information of the augmented model can be written as

$$U\left(B_{base}, B_E\right) = \begin{pmatrix} U_{base}\left(B_{base}, B_E\right) \\ U_E\left(B_{base}, B_E\right) \end{pmatrix}$$

$$I\left(B_{base}, B_E\right) = \begin{pmatrix} I_{base,base}\left(B_{base}, B_E\right) & I_{base,E}\left(B_{base}, B_E\right) \\ I_{E,base}\left(B_{base}, B_E\right) & I_{E,E}\left(B_{base}, B_E\right) \end{pmatrix}$$

$$J\left(B_{base}, B_E\right) = \begin{pmatrix} J_{base,base}\left(B_{base}, B_E\right) & J_{base,E}\left(B_{base}, B_E\right) \\ J_{E,base}\left(B_{base}, B_E\right) & J_{E,E}\left(B_{base}, B_E\right) \end{pmatrix}$$

Then the score statistic for testing our hypothesis will be

$$s = U_E\left(\hat{B}_{base}, 0\right)^T J_{E,E}\left(\hat{B}_{base}, 0\right) U_E\left(\hat{B}_{base}, 0\right)$$

where $U_E\left(\hat{B}_{base,}0\right)$ and $J_{E,E}\left(\hat{B}_{base}, 0\right)$ are the 2nd partition of score function and inverse information matrix evaluated at $B_{base} = \hat{B}_{base}$ and $B_E = 0$.

Under the null hypothesis, the score statistic s has a chi-square distribution with degrees of freedom equal to the rank of $J_{E,E}\left(B_1, B_2\right)$. If the rank of $J_{E,E}\left(B_1, B_2\right)$ is zero, then the score statistic will be set to 0 and the p-value will be 1. Otherwise, if the rank of $J_{E,E}\left(B_1, B_2\right)$ is $r_E : r_E > 0$, then the p-value of the test is equal to $1 - F\left(s; r_E\right)$, where $F\left(\cdot, r_E\right)$ is the cumulative distribution function of a chi-square distribution with r_E degrees of freedom.

Computational Formula for Score Statistic

When we compute the score statistic s, it is not necessary to re-compute $U\left(\hat{B}_{base,}0\right)$ and $I\left(\hat{B}_{base}, 0\right)$ from scratch. The score function and information matrix of the base model can be reused in the calculation. Using the block notations introduced earlier, we have

$$U\left(\hat{B}_{base}, 0\right) = \begin{pmatrix} U_{base}\left(\hat{B}_{base}, 0\right) \\ U_E\left(\hat{B}_{base}, 0\right) \end{pmatrix} = \begin{pmatrix} U\left(\hat{B}_{base}\right) \\ U_E\left(\hat{B}_{base}, 0\right) \end{pmatrix}$$

and

$$I\left(\hat{B}_{base}, 0\right) = \begin{pmatrix} I\left(\hat{B}_{base}\right) & I_{base,E}\left(\hat{B}_{base}, 0\right) \\ I_{E,base}\left(\hat{B}_{base}, 0\right) & I_{E,E}\left(\hat{B}_{base}, 0\right) \end{pmatrix}$$

In stepwise logistic regression, it is necessary to compute one score test for each effect that are not in the base model. Since the 1st partition of $U\left(\hat{B}_{base}, 0\right)$ and $I\left(\hat{B}_{base}, 0\right)$ depend only on the base model, we only need to compute $U_E\left(\hat{B}_{base}, 0\right)$, $I_{base,E}\left(\hat{B}_{base}, 0\right)$ and $I_{E,E}\left(\hat{B}_{base}, 0\right)$ for each new effect.

If β_{js} is the *s*-th parameter of the *j*-th logit in B_{base} and β_{kt} is the *t*-th parameter of *k*-th logit in B_E, then the elements of $U_E\left(\hat{B}_{base},0\right)$, $I_{base,E}\left(\hat{B}_{base},0\right)$ and $I_{E,E}\left(\hat{B}_{base},0\right)$ can be expressed as follows:

$$\left[U_E\left(\hat{B}_{base},0\right)\right]_{kt}=\sum_{i=1}^{m}x_{it}\left(n_{ik}-n_i\hat{\pi}_{ik}\right)$$

$$\left[I_{E,E}\left(\hat{B}_{base},0\right)\right]_{kt,k't'}=-\sum_{i=1}^{m}n_ix_{it}x_{it'}\hat{\pi}_{ik}\left(\delta_{kk'}-\hat{\pi}_{ik'}\right)$$

$$\left[I_{base,E}\left(\hat{B}_{base},0\right)\right]_{js,kt}=-\sum_{i=1}^{m}n_ix_{is}x_{it}\hat{\pi}_{ij}\left(\delta_{jk}-\hat{\pi}_{ik}\right)$$

where $\hat{\pi}_{ik}$, $\hat{\pi}_{ik'}$ are computed under the base model.

Wald's Test

In backward stepwise selection, we are interested in removing an effect *F* from an already fitted model. For a given base model with parameter vector B_{base}, we want to use Wald's statistic to test if effect *F* should be removed from the base model. If the parameter vector for the effect *F* is B_F, then the hypothesis can be formulated as

$H_0 : B_F = 0$ vs. $H_1 : B_F \neq 0$

In order to write down the expression of the Wald's statistic, we will partition our parameter vector (and its estimate) into two parts as follows:

$$B_{base}=\begin{pmatrix}B_{base\backslash F}\\ B_F\end{pmatrix}\text{ and }\hat{B}_{base}=\begin{pmatrix}\hat{B}_{base\backslash F}\\ \hat{B}_F\end{pmatrix}$$

The first partition contains parameters that we intended to keep in the model and the 2nd partition contains the parameters of the effect *F*, which may be removed from the model. The information matrix and inverse information will be partitioned accordingly,

$$I\left(B_{base}\right)=\begin{pmatrix}I_{base\backslash F,base\backslash F}\left(B_{base\backslash F},B_{base\backslash F}\right) & I_{base\backslash F,F}\left(B_{base\backslash F},B_F\right)\\ I_{F,base\backslash F}\left(B_{base\backslash F},B_F\right) & I_{F,F}\left(B_{base\backslash F},B_F\right)\end{pmatrix}$$

and

$$J\left(B_{base}\right)=\begin{pmatrix}J_{base\backslash F,base\backslash F}\left(B_{base\backslash F},B_{base\backslash F}\right) & J_{base\backslash F,F}\left(B_{base\backslash F},B_F\right)\\ J_{F,base\backslash F}\left(B_{base\backslash F},B_F\right) & J_{F,F}\left(B_{base\backslash F},B_F\right)\end{pmatrix}$$

Using the above notations, the Wald's statistic for effect *F* can be expressed as

$$w=\hat{B}_F\left[J_{F,F}\left(B_{base\backslash F},B_F\right)\right]^{-}\hat{B}_F$$

Under the null hypothesis, *w* has a chi-square distribution with degrees of freedom equal to the rank of $J_{F,F}\left(B_{base\backslash F},B_F\right)$. If the rank of $J_{F,F}\left(B_{base\backslash F},B_F\right)$ is zero, then Wald's statistic will be set to 0 and the *p*-value will be 1. Otherwise, if the rank of $J_{F,F}\left(B_{base\backslash F},B_F\right)$ is $r_F : r_F > 0$, then the p-value of the test is equal to $1 - F\left(w; r_F\right)$, where $F\left(w; r_F\right)$ is the cumulative distribution function of a chi-square distribution with r_F degrees of freedom.

Generated Model/Scoring

Predicted Values

The predicted value for a record *i* is the output field category *j* with the largest logit value r_{ij},

$$r_{ij} = \log\left(\frac{\pi_{ij}}{\pi_{iJ}}\right) = \mathbf{x'}_i\beta_j$$

for *j* = 1, ..., *J*-1. The logit for reference category *J*,r_{iJ}, is 1.0.

Predicted Probability

The probability for the predicted category j^* for scored record *i* is derived from the logit for category j^*,

$$\hat{\pi}_{ij} = \frac{\exp\left(r_{ij'}\right)}{1 + \sum_{k=1}^{J-1} \exp\left(r_{ij'}\right)} = \frac{\exp\left(\mathbf{x'}_i\beta_j\right)}{1 + \sum_{k=1}^{J-1} \exp\left(\mathbf{x'}_i\beta_k\right)}$$

If the Append all probabilities option is selected, the probability is calculated for all *J* categories in a similar manner.

Blank Handling

Records with missing values for any input field cannot be scored and are assigned a predicted value and probability value(s) of $null$.

Binomial Logistic Regression

For binomial models (models with a flag field as the target), IBM® SPSS® Modeler uses an algorithm optimized for such models, as described here.

Notation

The following notation is used throughout this chapter unless otherwise stated:

n	The number of observed cases
p	The number of parameters
$\mathbf{y}$	$n \times 1$ vector with element y_i, the observed value of the *i*th case of the dichotomous dependent variable
$\mathbf{X}$	$n \times p$ matrix with element x_{ij}, the observed value of the *i*th case of the *j*th parameter

β	$p \times 1$ vector with element β_j, the coefficient for the *j*th parameter
w	$n \times 1$ vector with element w_i, the weight for the *i*th case
l	Likelihood function
L	Log-likelihood function
I	Information matrix

Model

The linear logistic model assumes a dichotomous dependent variable *Y* with probability π, where for the *i*th case,

$$\pi_i = \frac{\exp(\eta_i)}{1+\exp(\eta_i)}$$

or

$$\ln\left(\frac{\pi_i}{1-\pi_i}\right) = \eta_i = \mathbf{X}_i^{'}\beta$$

Hence, the likelihood function *l* for *n* observations $y_1, \ldots, y_n$, with probabilities $\pi_1, \ldots, \pi_n$ and case weights $w_1, \ldots, w_n$, can be written as

$$l = \prod_{i=1}^{n} \pi_i^{w_i y_i}(1-\pi_i)^{w_i(1-y_i)}$$

It follows that the logarithm of *l* is

$$L = \ln(l) = \sum_{i=1}^{n} (w_i y_i \ln(\pi_i) + w_i(1-y_i)\ln(1-\pi_i))$$

and the derivative of L with respect to β_j is

$$L_{X_j}^{*} = \frac{\partial L}{\partial \beta_j} = \sum_{i=1}^{n} w_i(y_i - \pi_i)x_{ij}$$

Maximum Likelihood Estimates (MLE)

The maximum likelihood estimates for β satisfy the following equations

$$\sum_{i=1}^{n} w_i(y_i - \hat{\pi}_i)x_{ij} = 0 \text{, for the } j\text{th parameter}$$

where $x_{i0} = 1$ for $i = 1, \ldots, n$.

Note the following:

1. A Newton-Raphson type algorithm is used to obtain the MLEs. Convergence can be based on
   - Absolute difference for the parameter estimates between the iterations

- Percent difference in the log-likelihood function between successive iterations
- Maximum number of iterations specified

2. During the iterations, if $\hat{\pi}_i(1-\hat{\pi}_i)$ is smaller than 10^{-8} for all cases, the log-likelihood function is very close to zero. In this situation, iteration stops and the message "All predicted values are either 1 or 0" is issued.

   After the maximum likelihood estimates $\hat{\beta}$ are obtained, the asymptotic covariance matrix is estimated by I^{-1}, the inverse of the information matrix *I*, where

   $$I = -\left[E\left(\frac{\partial^2 L}{\partial\beta_i \partial\beta_j}\right)\right] = \mathbf{X}'\mathbf{W}\hat{\mathbf{V}}\mathbf{X},$$

   $$\hat{\mathbf{V}} = Diag\{\hat{\pi}_1(1-\hat{\pi}_1), \ldots, \hat{\pi}_n(1-\hat{\pi}_n)\},$$

   $$\mathbf{W} = Diag\{w_1, \ldots, w_n\},$$

   $$\hat{\pi}_i = \frac{\exp(\hat{\eta}_i)}{1+\exp(\hat{\eta}_i)},$$

   and

   $$\hat{\eta}_i = \mathbf{X}_i{}'\hat{\beta}$$

Stepwise Variable Selection

Several methods are available for selecting independent variables. With the forced entry method, any variable in the variable list is entered into the model. There are two stepwise methods: forward and backward. The stepwise methods can use either the Wald statistic, the likelihood ratio, or a conditional algorithm for variable removal. For both stepwise methods, the score statistic is used to select variables for entry into the model.

Forward Stepwise (FSTEP)

1. If FSTEP is the first method requested, estimate the parameter and likelihood function for the initial model. Otherwise, the final model from the previous method is the initial model for FSTEP. Obtain the necessary information: MLEs of the parameters for the current model, predicted probability, likelihood function for the current model, and so on.
2. Based on the MLEs of the current model, calculate the score statistic for every variable eligible for inclusion and find its significance.
3. Choose the variable with the smallest significance. If that significance is less than the probability for a variable to enter, then go to step 4; otherwise, stop FSTEP.
4. Update the current model by adding a new variable. If this results in a model which has already been evaluated, stop FSTEP.
5. Calculate LR or Wald statistic or conditional statistic for each variable in the current model. Then calculate its corresponding significance.

6. Choose the variable with the largest significance. If that significance is less than the probability for variable removal, then go back to step 2; otherwise, if the current model with the variable deleted is the same as a previous model, stop FSTEP; otherwise, go to the next step.

7. Modify the current model by removing the variable with the largest significance from the previous model. Estimate the parameters for the modified model and go back to step 5.

Backward Stepwise (BSTEP)

1. Estimate the parameters for the full model which includes the final model from previous method and all eligible variables. Only variables listed on the BSTEP variable list are eligible for entry and removal. Let the current model be the full model.

2. Based on the MLEs of the current model, calculate the LR or Wald statistic or conditional statistic for every variable in the model and find its significance.

3. Choose the variable with the largest significance. If that significance is less than the probability for a variable removal, then go to step 5; otherwise, if the current model without the variable with the largest significance is the same as the previous model, stop BSTEP; otherwise, go to the next step.

4. Modify the current model by removing the variable with the largest significance from the model. Estimate the parameters for the modified model and go back to step 2.

5. Check to see any eligible variable is not in the model. If there is none, stop BSTEP; otherwise, go to the next step.

6. Based on the MLEs of the current model, calculate the score statistic for every variable not in the model and find its significance.

7. Choose the variable with the smallest significance. If that significance is less than the probability for variable entry, then go to the next step; otherwise, stop BSTEP.

8. Add the variable with the smallest significance to the current model. If the model is not the same as any previous models, estimate the parameters for the new model and go back to step 2; otherwise, stop BSTEP.

Stepwise Statistics

The statistics used in the stepwise variable selection methods are defined as follows.

Score Statistic

The score statistic is calculated for each variable not in the model to determine whether the variable should enter the model. Assume that there are r_1 variables, namely, $\alpha_1, \ldots, \alpha_{r_1}$ in the model and r_2 variables, $\gamma_1, \ldots, \gamma_{r_2}$, not in the model. The score statistic for γ_i is defined as

$$\mathbf{S}_i = \left(\mathbf{L}^*_{\gamma_i}\right)^2 \mathbf{B}_{22,i}$$

if γ_i is not a categorical variable. If γ_i is a categorical variable with *m* categories, it is converted to a $(m-1)$-dimension dummy vector. Denote these new $m-1$ variables as $\tilde{\gamma}_i, \ldots, \tilde{\gamma}_{i+m-2}$. The score statistic for γ_i is then

$$\mathbf{S}_i = \left(\mathbf{L}^*_{\tilde{\gamma}}\right)' \mathbf{B}_{22,i} \mathbf{L}^*_{\tilde{\gamma}}$$

where $\left(\mathbf{L}^*_{\tilde{\gamma}}\right)' = \left(L^*_{\tilde{\gamma}_i}, \ldots, L^*_{\tilde{\gamma}_{i+m-2}}\right)$ and the $(m-1) \times (m-1)$ matrix $\mathbf{B}_{22,i}$ is

$$\mathbf{B}_{22,i} = \left(\mathbf{A}_{22,i} - \mathbf{A}_{21,i}\mathbf{A}_{11}^{-1}\mathbf{A}_{12,i}\right)^{-1}$$

with

$$\begin{aligned} \mathbf{A}_{11} &= \underset{\sim}{\alpha}'\hat{\mathbf{V}}\underset{\sim}{\alpha}, \\ \mathbf{A}_{12,i} &= \underset{\sim}{\alpha}'\hat{\mathbf{V}}\underset{\sim}{\gamma_i}, \\ \mathbf{A}_{22,i} &= \underset{\sim i}{\gamma_i'}\hat{\mathbf{V}}\underset{\sim}{\gamma_i} \end{aligned}$$

in which $\underset{\sim}{\alpha}$ is the design matrix for variables $\alpha_1, \ldots, \alpha_{r_1}$ and $\underset{\sim}{\gamma_i}$ is the design matrix for dummy variables $\tilde{\gamma}_i, \ldots, \tilde{\gamma}_{i+m-2}$. Note that $\underset{\sim}{\alpha}$ contains a column of ones unless the constant term is excluded from η. Based on the MLEs for the parameters in the model, $\mathbf{V}$ is estimated by $\hat{\mathbf{V}} = Diag\{\hat{\pi}_1(1-\hat{\pi}_1), \ldots, \hat{\pi}_n(1-\hat{\pi}_n)\}$. The asymptotic distribution of the score statistic is a chi-square with degrees of freedom equal to the number of variables involved.

Note the following:

1. If the model is through the origin and there are no variables in the model, $\mathbf{B}_{22,i}$ is defined by $\mathbf{A}_{22,i}^{-1}$ and $\hat{\mathbf{V}}$ is equal to $\frac{1}{4}\mathbf{I}_n$.
2. If $\mathbf{B}_{22,i}$ is not positive definite, the score statistic and residual chi-square statistic are set to be zero.

Wald Statistic

The Wald statistic is calculated for the variables in the model to determine whether a variable should be removed. If the *i*th variable is not categorical, the Wald statistic is defined by

$$Wald_i = \frac{\hat{\beta}_i^2}{\hat{\sigma}^2_{\hat{\beta}_i}}$$

If it is a categorical variable, the Wald statistic is computed as follows:

Let $\hat{\beta}_i$ be the vector of maximum likelihood estimates associated with the $m-1$ dummy variables, and $\mathbf{C}$ the asymptotic covariance matrix for $\hat{\beta}_i$. The Wald statistic is

$$Wald_i = \hat{\beta}_i'\mathbf{C}^{-1}\hat{\beta}_i$$

The asymptotic distribution of the Wald statistic is chi-square with degrees of freedom equal to the number of parameters estimated.

Likelihood Ratio (LR) Statistic

The LR statistic is defined as two times the log of the ratio of the likelihood functions of two models evaluated at their MLEs. The LR statistic is used to determine if a variable should be removed from the model. Assume that there are r_1 variables in the current model which is referred to as a full model. Based on the MLEs of the full model, *l(full)* is calculated. For each of the variables removed from the full model one at a time, MLEs are computed and the likelihood function *l(reduced)* is calculated. The LR statistic is then defined as

$$LR = -2\ln\left(\tfrac{l(reduced)}{l(full)}\right) = -2(L(reduced) - L(full))$$

LR is asymptotically chi-square distributed with degrees of freedom equal to the difference between the numbers of parameters estimated in the two models.

Conditional Statistic

The conditional statistic is also computed for every variable in the model. The formula for the conditional statistic is the same as the LR statistic except that the parameter estimates for each reduced model are conditional estimates, not MLEs. The conditional estimates are defined as follows. Let $\hat{\beta} = \left(\hat{\beta}_1, \ldots, \hat{\beta}_{r_1}\right)'$ be the MLE for the r_1 variables in the model and **C** be the asymptotic covariance matrix for $\hat{\beta}$. If variable x_i is removed from the model, the conditional estimate for the parameters left in the model given $\hat{\beta}$ is

$$\tilde{\beta}_{(i)} = \hat{\beta}_{(i)} - \mathbf{c}_{12}^{(i)}\left(\mathbf{c}_{22}^{(i)}\right)^{-1}\hat{\beta}_i$$

where $\hat{\beta}_i$ is the MLE for the parameter(s) associated with x_i and $\hat{\beta_{(i)}}$ is $\hat{\beta}$ with $\hat{\beta}_i$ removed, $\mathbf{c}_{12}^{(i)}$ is the covariance between $\hat{\beta_{(i)}}$ and $\hat{\beta}_i$, and $\mathbf{c}_{22}^{(i)}$ is the covariance of $\hat{\beta}_i$. Then the conditional statistic is computed by

$$-2\left(L\left(\tilde{\beta}_{(i)}\right) - L(full)\right)$$

where $L\left(\tilde{\beta}_{(i)}\right)$ is the log-likelihood function evaluated at $\hat{\beta_{(i)}}$.

Statistics

The following output statistics are available.

Initial Model Information

If β_0 is not included in the model, the predicted probability is estimated to be 0.5 for all cases and the log-likelihood function $L(0)$ is

$$L(0) = W\ln(0.5) = -0.6931472W$$

with $W = \sum_{i=1}^{n} w_i$. If β_0 is included in the model, the predicted probability is estimated as

$$\hat{\pi}_0 = \frac{\sum_{i=1}^{n} w_i y_i}{W}$$

and β_0 is estimated by

$$\hat{\beta}_0 = \ln\left(\frac{\hat{\pi}_0}{1-\hat{\pi}_0}\right)$$

with asymptotic standard error estimated by

$$\hat{\sigma}_{\hat{\beta}_0} = \frac{1}{\sqrt{W\hat{\pi}_0(1-\hat{\pi}_0)}}$$

The log-likelihood function is

$$L(0) = W\left[\hat{\pi}_0 \ln\left(\frac{\hat{\pi}_0}{1-\hat{\pi}_0}\right) + \ln\left(1-\hat{\pi}_0\right).\right]$$

Model Information

The following statistics are computed if a stepwise method is specified.

–2 Log-Likelihood

$$-2\sum_{i=1}^{n}\left(w_i y_i \ln\left(\hat{\pi}_i\right) + w_i(1-y_i)\ln\left(1-\hat{\pi}_i\right)\right)$$

Model Chi-Square

2(log-likelihood function for current model − log-likelihood function for initial model)

The initial model contains a constant if it is in the model; otherwise, the model has no terms. The degrees of freedom for the model chi-square statistic is equal to the difference between the numbers of parameters estimated in each of the two models. If the degrees of freedom is zero, the model chi-square is not computed.

Block Chi-Square

2(log-likelihood function for current model − log-likelihood function for the final model from the previous method)

The degrees of freedom for the block chi-square statistic is equal to the difference between the numbers of parameters estimated in each of the two models.

Improvement Chi-Square

2(log-likelihood function for current model − log-likelihood function for the model from the last step)

The degrees of freedom for the improvement chi-square statistic is equal to the difference between the numbers of parameters estimated in each of the two models.

Goodness of Fit

$$\sum_{i=1}^{n} \frac{w_i(y_i - \hat{\pi}_i)^2}{\hat{\pi}_i(1-\hat{\pi}_i)}$$

Cox and Snell's R-Square (Cox and Snell, 1989; Nagelkerke, 1991)

$$R^2_{CS} = 1 - \left(\frac{l(0)}{l(\hat{\beta})}\right)^{\frac{2}{W}}$$

where $l\left(\hat{\beta}\right)$ is the likelihood of the current model and *l(0)* is the likelihood of the initial model; that is, $l(0) = W\log(0.5)$ if the constant is not included in the model; $l(0) = W[\hat{\pi}_o \log\{\hat{\pi}_o/(1-\hat{\pi}_o)\} + \log(1-\hat{\pi}_o)]$ if the constant is included in the model, where $\hat{\pi}_o = \Sigma_i^n w_i y_i / W$.

Nagelkerke's R-Square (Nagelkerke, 1981)

$$R^2_N = R^2_{CS}/\max\left(R^2_{CS}\right)$$

where $\max\left(R^2_{CS}\right) = 1 - \{l(0)\}^{2/W}$.

Hosmer-Lemeshow Goodness-of-Fit Statistic

The test statistic is obtained by applying a chi-square test on a $2 \times g$ contingency table. The contingency table is constructed by cross-classifying the dichotomous dependent variable with a grouping variable (with *g* groups) in which groups are formed by partitioning the predicted probabilities using the percentiles of the predicted event probability. In the calculation, approximately 10 groups are used (*g*=10). The corresponding groups are often referred to as the "deciles of risk" (Hosmer and Lemeshow, 2000).

If the values of independent variables for observation *i* and *i'* are the same, observations *i* and *i'* are said to be in the same block. When one or more blocks occur within the same decile, the blocks are assigned to this same group. Moreover, observations in the same block are not divided when they are placed into groups. This strategy may result in fewer than 10 groups (that is, $g \leq 10$) and consequently, fewer degrees of freedom.

Suppose that there are *Q* blocks, and the *q*th block has m_q number of observations, $q = 1, \ldots, Q$. Moreover, suppose that the *k*th group ($k = 1, \ldots, g$) is composed of the q_1th, …, q_kth blocks of observations. Then the total number of observations in the *k*th group is $s_k = \Sigma_{q_1}^{q_k} m_j$. The total observed frequency of events (that is, *Y*=1) in the *k*th group, call it O_{1k}, is the total number of observations in the *k*th group with *Y*=1. Let E_{1k} be the total expected frequency of the event in the *k*th group; then E_{1k} is given by $E_{1k} = s_k \xi_k$, where ξ_k is the average predicted event probability for the *k*th group.

$$\xi_k = \Sigma_{q_1}^{q_k} m_j \hat{\pi}_j / s_k$$

The Hosmer-Lemeshow goodness-of-fit statistic is computed as

$$\chi^2_{HL} = \sum_{k=1}^{g} \frac{(O_{1k} - E_{1k})^2}{E_{1k}\,(1-\xi_k)}$$

The *p* value is given by $\Pr\left(\chi^2 \geq \chi^2_{HL}\right)$ where χ^2 is the chi-square statistic distributed with degrees of freedom (*g*–2).

Information for the Variables Not in the Equation

For each of the variables not in the equation, the score statistic is calculated along with the associated degrees of freedom, significance and partial *R*. Let X_i be a variable not currently in the model and S_i the score statistic. The partial *R* is defined by

$$Partial\_R = \begin{cases} \sqrt{\frac{S_i - 2\times df}{-2L(initial)}} & \text{if } S_i > 2 \times df \\ 0 & \text{otherwise} \end{cases}$$

where *df* is the degrees of freedom associated with S_i, and $L(initial)$ is the log-likelihood function for the initial model.

The residual Chi-Square printed for the variables not in the equation is defined as

$$R_{CS} = \left(L^*_{\mathrm{g}}\right)' B_{22} L^*_{\mathrm{g}}$$

where $L^*_{\mathrm{g}} = \left(L^*_{\gamma_1}, \ldots, L^*_{\gamma_{r_2}}\right)'$

Information for the Variables in the Equation

For each of the variables in the equation, the MLE of the Beta coefficients is calculated along with the standard errors, Wald statistics, degrees of freedom, significances, and partial *R*. If X_i is not a categorical variable currently in the equation, the partial *R* is computed as

$$Partial\_R = \begin{cases} sign\left(\hat{\beta}_i\right)\sqrt{\frac{Wald_i - 2}{-2L(initial)}} & \text{if } Wald_i > 2 \\ 0 & \text{otherwise} \end{cases}$$

If X_i is a categorical variable with *m* categories, the partial *R* is then

$$Partial\_R = \begin{cases} \sqrt{\frac{Wald_i - 2(m-1)}{-2L(initial)}} & \text{if } Wald_i > 2(m-1) \\ 0 & \text{otherwise} \end{cases}$$

Casewise Statistics

The following statistics are computed for each case.

Individual Deviance

The deviance of the *i*th case, G_i, is defined as

$$G_i = \begin{cases} \sqrt{2(y_i \ln(\hat{\pi}_i) + (1-y_i)\ln(1-\hat{\pi}_i))} & \text{if } y_i > \hat{\pi}_i \\ -\sqrt{2(y_i \ln(\hat{\pi}_i) + (1-y_i)\ln(1-\hat{\pi}_i))} & \text{otherwise} \end{cases}$$

Leverage

The leverage of the *i*th case, h_i, is the *i*th diagonal element of the matrix

$$\hat{\mathbf{V}}^{\frac{1}{2}}\mathbf{X}\left(\mathbf{X}'\mathbf{C}\hat{\mathbf{V}}\mathbf{X}\right)^{-1}\mathbf{X}'\hat{\mathbf{V}}^{\frac{1}{2}}$$

where

$$\hat{\mathbf{V}} = Diag\{\hat{\pi}_1(1-\hat{\pi}_1),\, \ldots, \hat{\pi}_n(1-\hat{\pi}_n)\}$$

Studentized Residual

$$\tilde{G}_i^* = \frac{G_i}{\sqrt{1-h_i}}$$

Logit Residual

$$\tilde{e}_i = \frac{e_i}{\hat{\pi}_i(1-\hat{\pi}_i)}$$

where $e_i = y_i - \hat{\pi}_i$

Standardized Residual

$$z_i = \frac{e_i}{\sqrt{\hat{\pi}_i(1-\hat{\pi}_i)}}$$

Cook's Distance

$$D_i = \frac{z_i^2 h_i}{1-h_i}$$

DFBETA

Let $\Delta\beta_i$ be the change of the coefficient estimates from the deletion of case *i*. It is computed as

$$\Delta\beta_i = \frac{\left(\mathbf{X}'\mathbf{C}\hat{\mathbf{V}}\mathbf{X}\right)^{-1}\mathbf{X}'_{i} e_i}{1-h_i}$$

Predicted Group

If $\hat{\pi}_i \geq 0.5$, the predicted group is the group in which *y*=1.

Note the following:

For the unselected cases with nonmissing values for the independent variables in the analysis, the leverage $\left(\tilde{h}_i\right)$ is computed as

$$\tilde{h}_i = h_i - \frac{\hat{V}_i h_i^2}{1+\hat{V}_i h_i}$$

where

$$h_i = \hat{V}_i \mathbf{X}'_i \left(\mathbf{X}' \mathbf{C} \hat{\mathbf{V}} \mathbf{X} \right)^{-1} \mathbf{X}_i$$

For the unselected cases, the Cook's distance and DFBETA are calculated based on $\tilde{h}_i$.

Generated Model/Scoring

For each record passed through a generated binomial logistic regression model, a predicted value and confidence score are calculated as follows:

Predicted Value

The probability of the value *y* = 1 for record *i* is calculated as

$$\hat{\pi}_i = \frac{\exp(\hat{\eta}_i)}{1 + \exp(\hat{\eta}_i)}$$

where

$$\hat{\eta}_i = \mathbf{X}_i{}' \hat{\beta}$$

If $\hat{\pi} > 0.5$, the predicted value is 1; otherwise, the predicted value is 0.

Confidence

For records with a predicted value of *y* = 1, the confidence value is $\hat{\pi}$. For records with a predicted value of *y* = 0, the confidence value is $(1 - \hat{\pi})$.

Blank Handling (generated model)

Records with missing values for any input field in the final model cannot be scored, and are assigned a predicted value of $null$.

Neural Networks Algorithms

Neural networks predict a continuous or categorical target based on one or more predictors by finding unknown and possibly complex patterns in the data.

For algorithms on enhancing model accuracy, enhancing model stability, or working with very large datasets, see "Ensembles Algorithms" on p. 125.

Multilayer Perceptron

The multilayer perceptron (MLP) is a feed-forward, supervised learning network with up to two hidden layers. The MLP network is a function of one or more predictors that minimizes the prediction error of one or more targets. Predictors and targets can be a mix of categorical and continuous fields.

Notation

The following notation is used for multilayer perceptrons unless otherwise stated:

$X^{(m)} = \left(x_1^{(m)}, ..., x_P^{(m)}\right)$	Input vector, pattern *m*, *m*=1,...*M*.
$Y^{(m)} = \left(y_1^{(m)}, ..., y_R^{(m)}\right)$	Target vector, pattern *m*.
I	Number of layers, discounting the input layer.
J_i	Number of units in layer *i*. $J_0 = P$, $J_\mathrm{i} = R$, discounting the bias unit.
Γ^c	Set of categorical outputs.
Γ	Set of continuous outputs.
Γ_h	Set of subvectors of $Y^{(m)}$ containing 1-of-*c* coded *h*th categorical field.
$a_{i:j}^m$	Unit *j* of layer *i*, pattern *m*, $j = 0, ..., J_i; i = 0, ..., I$.
$w_{i:j,k}$	Weight leading from layer *i*−1, unit *j* to layer *i*, unit *k*. No weights connect $a_{i-1:j}^m$ and the bias $a_{i:0}^m$; that is, there is no $w_{i:j,0}$ for any *j*.
$c_{i:k}^m$	$\sum_{j=0}^{J_{i-1}} w_{i:j,k} a_{i-1:j}^m$, *i*=1,...,*I*.
$\gamma_i\left(c\right)$	Activation function for layer *i*.
$\mathbf{w}$	Weight vector containing all weights $\left(w_{1:0,1}, w_{1:0,2}, ..., w_{I:J_{I-1},J_I}\right)$.

Architecture

The general architecture for MLP networks is:

Input layer: J_0=*P* units, $a_{0:1}, \cdots, a_{0:J_0}$; with $a_{0:j} = x_j$.

ith hidden layer: J_i units, $a_{i:1}, \cdots, a_{i:J_i}$; with $a_{i:k} = \gamma_i\left(c_{i:k}\right)$ and $c_{i:k} = \sum_{j=0}^{J_{i-1}} w_{i:j,k} a_{i-1:j}$ where $a_{i-1:0} = 1$.

Output layer: J_{I}=R units, $a_{I:1}, \cdots, a_{I:J_I}$; with $a_{I:k} = \gamma_I\left(c_{I:k}\right)$ and $c_{I:k} = \sum_{j=0}^{J_1} w_{I:j,k}a_{i-1:j}$ where $a_{i-1:0} = 1$.

Note that the pattern index and the bias term of each layer are not counted in the total number of units for that layer.

Activation Functions

Hyperbolic Tangent

$$\gamma\left(c\right) = \tanh\left(c\right) = \frac{e^c - e^{-c}}{e^c + e^{-c}}$$

This function is used for hidden layers.

Identity

$$\gamma(c) = c$$

This function is used for the output layer when there are continuous targets.

Softmax

$$\gamma\left(c_k\right) = \frac{\exp\left(c_k\right)}{\sum\limits_{j\in\Gamma_h} \exp\left(c_j\right)}$$

This function is used for the output layer when all targets are categorical.

Error Functions

Sum-of-Squares

$$E_T\left(w\right) = \sum_{m=1}^{M} E_m\left(w\right)$$

where

$$E_m\left(w\right) = \frac{1}{2}\sum_{r=1}^{R}\left(y_r^{(m)} - a_{I:r}^m\right)^2$$

This function is used when there are continuous targets.

Cross-Entropy

$$E_T(w) = \sum_{m=1}^{M} E_m(w)$$

where

$$E_m(w) = -\sum_{r \in \Gamma^c} y_r^{(m)} \log\left(\frac{a_{I:r}^{m}}{y_r^{(m)}}\right)$$

This function is used when all targets are categorical.

Expert Architecture Selection

Expert architecture selection determines the “best” number of hidden units in a single hidden layer.

A random sample is taken from the entire data set and split into training (70%) and testing samples (30%). The size of random sample is $N = 1000$. If entire dataset has less than N records, use all of them. If training and testing data sets are supplied separately, the random samples for training and testing should be taken from the respective datasets.

Given $K_{\min}$ and $K_{\max}$, the algorithm is as follows.

1. Start with an initial network of k hidden units. The default is k=min(g(R,P),20,h(R,P)), where

   $$g(R,P) = \begin{cases} \lfloor 4.5 + \sqrt{P+R} \rfloor & R < 5, P \geq 8 \\ \lfloor 0.5 + 0.5(P+R) \rfloor & \text{otherwise} \end{cases}$$

   where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. $h(R,P) = \left\lfloor \frac{M-R}{P+R+1} \right\rfloor$ is the maximum number of hidden units that will not result in more weights than there are records in the entire training set.

   If $k < K_{\min}$, set $k = K_{\min}$. Else if $k > K_{\max}$, set $k = K_{\max}$. Train this network once via the alternated simulated annealing and training procedure (steps 1 to 5).

2. If $k > K_{\min}$, set *DOWN=TRUE*. Else if training error ratio > 0.01, *DOWN=FALSE*. Else stop and report the initial network.

3. If *DOWN=TRUE*, remove the weakest hidden unit (see below); k=k−1. Else add a hidden unit; k=k+1.

4. Using the previously fit weights as initial weights for the old weights and random weights for the new weights, train the old and new weights for the network once through the alternated simulated annealing and training procedure (steps 3 to 5) until the stopping conditions are met.

5. If the error on test data has dropped:

   If *DOWN=FALSE*, If k< $K_{\max}$ and the training error has dropped but the error ratio is still above 0.01, return to step 3. Else if k> $K_{\min}$, return to step 3. Else, stop and report the network with the minimum test error.

Else if *DOWN=TRUE*, If |k−k_0|>1, stop and report the network with the minimum test error. Else if training error ratio for k=k_0 is bigger than 0.01, set *DOWN=FALSE*, k=k_0 return to step 3. Else stop and report the initial network.

Else stop and report the network with the minimum test error.

If more than one network attains the minimum test error, choose the one with fewest hidden units.

If the resulting network from this procedure has training error ratio (training error divided by error from the model using average of an output field to predict that field) bigger than 0.1, repeat the architecture selection with different initial weights until either the error ratio is <=0.1 or the procedure is repeated 5 times, then pick the one with smallest test error.

Using this network with its weights as initial values, retrain the network on the entire training set.

The weakest hidden unit

For each hidden unit j, calculate the error on the test data when j is removed from the network. The weakest hidden unit is the one having the smallest total test error upon its removal.

Training

The problem of estimating the weights consists of the following parts:

- Initializing the weights. Take a random sample and apply the alternated simulated annealing and training procedure on the random sample to derive the initial weights. Training in step 3 is performed using all default training parameters.
- Computing the derivative of the error function with respect to the weights. This is solved via the error backpropagation algorithm.
- Updating the estimated weights. This is solved by the gradient descent or scaled conjugate gradient method.

Alternated Simulated Annealing and Training

The following procedure uses simulated annealing and training alternately up to K_1 times. Simulated annealing is used to break out of the local minimum that training finds by perturbing the local minimum K_2 times. If break out is successful, simulated annealing sets a better initial weight for the next training. We hope to find the global minimum by repeating this procedure K_3 times. This procedure is rather expensive for large data sets, so it is only used on a random sample to search for initial weights and in architecture selection. Let K_1=K_2=4, K_3=3.

1. Randomly generate K_2 weight vectors between [a_0−a, a_0+a], where a_0=0 and a=0.5. Calculate the training error for each weight vector. Pick the weights that give the minimum training error as the initial weights.
2. Set k_1=0.
3. Train the network with the specified initial weights. Call the trained weights **w**.

4. If the training error ratio <= 0.05, stop the k_1 loop and use **w** as the result of the loop. Else set $k_1 = k_1+1$.

5. If $k_1 < K_1$, perturb the old weight to form K_2 new weights $\mathbf{w}^{'} = \mathbf{w} + \mathbf{w}_n$ by adding K_2 different random noise $\mathbf{w}_n$ between $[a(k_1), a(k_1)]$ where $a\left(k_1\right) = (0.5)^{k_1}a$. Let $\mathbf{w}_{\min}$ be the weights that give the minimum training error among all the perturbed weights. If $E_T\left(\mathbf{w}_{\min}\right) < E_T\left(\mathbf{w}\right)$, set the initial weights to be $\mathbf{w}_{\min}$, return to step 3. Else stop and report **w** as the final result.

   Else stop the k_1 loop and use **w** as the result of the loop.

   If the resulting weights have training error ratio bigger than 0.1, repeat this algorithm until either the training error ratio is <=0.1 or the procedure is repeated K_3 times, then pick the one with smallest test error among the result of the k_1 loops.

Error Backpropagation

Error-backpropagation is used to compute the first partial derivatives of the error function with respect to the weights.

First note that $\gamma^{'}\left(c\right) = \begin{cases} 1 - \left[\gamma\left(c\right)\right]^2 & \text{tanh} \\ 1 & \text{identity} \end{cases}$

The backpropagation algorithm follows:

For each *i*,*j*,*k*, set $\frac{\partial E_T}{\partial w_{i:k,j}} = 0$.

For each *m* in group *T*; For each $p=1,\ldots,J_I$, let

$$\delta^m_{I:p} = \frac{\partial E_m}{\partial c^m_{I:p}} = \begin{cases} a^m_{I:p} - y^{(m)}_p & \text{if cross-entropy error is used} \\ \gamma^{'}_I\left(c^m_{I:p}\right)\left(a^m_{I:p} - y^{(m)}_p\right) & \text{otherwise} \end{cases}$$

For each $i=I,\ldots,1$ (start from the output layer); For each $j=1,\ldots,J_i$; For each $k=0,\ldots,J_{i-1}$

- Let $\frac{\partial E_m}{\partial w_{i:k,j}} = \delta^m_{i:j} a^m_{i-1:k}$, where $\delta^m_{i:j} = \frac{\partial E_m}{\partial c^m_{i:j}}$
- Set $\frac{\partial E_T}{\partial w_{i:k,j}} = \frac{\partial E_T}{\partial w_{i:k,j}} + \frac{\partial E_m}{\partial w_{i:k,j}}$
- If $k > 0$ and $i > 1$, set $\delta^m_{i-1:k} = \gamma^{'}_{i-1}\left(c^m_{i-1:k}\right)\sum_{j=1}^{J_i}\delta^m_{i:j}w_{i:k,j}$

This gives us a vector of $\sum_{i=0}^{I-1}\left(J_i+1\right)J_{i+1}$ elements that form the gradient of $E_T\left(w_k\right)$.

Gradient Descent

Given the learning rate parameter η_0 (set to 0.4) and momentum rate α (set to 0.9), the gradient descent method is as follows.

1. Let k=0. Initialize the weight vector to w_0, learning rate to η_0. Let $\Delta w_0 = 0$.

2. Read all data and find $E_T\left(w_k\right)$ and its gradient $g_k = \nabla E_T\left(w_k\right)$. If $|g_k| < 10^{-6}$, stop and report the current network.

3. If $\eta_k|g_k| \le \alpha|\Delta w_k|$, $\alpha = 0.9\eta_k\frac{|g_k|}{|\Delta w_k|}$. This step is to make sure that the steepest gradient descent direction dominates weight change in next step. Without this step, the weight change in next step could be along the opposite direction of the steepest descent and hence no matter how small η_k is, the error will not decrease.

4. Let $v = w_k - \eta_k g_k + \alpha\Delta w_k$

5. If $E_T\left(v\right) < E_T\left(w_k\right)$, then set $w_{k+1} = v$, $\Delta w_{k+1} = w_{k+1} - w_k$, and $\eta_{k+1} = \eta_k$, Else $\eta_k = .5\eta_k$ and return to step 3.

6. If a stopping rule is met, exit and report the network as stated in the stopping criteria. Else let *k*=*k*+1 and return to step 2.

Model Update

Given the learning rate parameters η_0 (set to 0.4) and η_{low} (set to 0.001), momentum rate α (set to 0.9), and learning rate decay factor β = (1/*pK*)*ln(η$_0$/η$_{low}$), the gradient descent method for online and mini-batch training is as follows.

1. Let *k*=0. Initialize the weight vector to w_0, learning rate to η_0. Let $\Delta w_0 = 0$.

2. Read records in T_k (T_k is randomly chosen) and find $E_{T_k}\left(w_k\right)$ and its gradient $g_k = \nabla E_{T_k}\left(w_k\right)$.

3. If $\eta_k|g_k| \le \alpha|\Delta w_k|$, $\alpha = 0.9\eta_k\frac{|g_k|}{|\Delta w_k|}$. This step is to make sure that the steepest gradient descent direction dominates weight change in next step. Without this step, the weight change in next step could be along the opposite direction of the steepest descent and hence no matter how small η_k is, the error will not decrease.

4. Let $v = w_k - \eta_k g_k + \alpha\Delta w_k$.

5. If $E_{T_k}\left(v\right) < E_{T_k}\left(w_k\right)$, then set $w_{k+1} = v$ and $\Delta w_{k+1} = w_{k+1} - w_k$, Else $w_{k+1} = w_k, \Delta w_{k+1} = \Delta w_k$.

6. $\eta_{k+1} = e^{-\beta}\eta_k$. If $\eta_{k+1} < \eta_{low}$, then set $\eta_{k+1} = \eta_{low}$.

7. If a stopping rule is met, exit and report the network as stated in the stopping criteria. Else let *k*=*k*+1 and return to step 2.

Scaled Conjugate Gradient

To begin, initialize the weight vector to $\mathbf{w}_0$, and let *N* be the total number of weights.

1. *k*=0. Set scalars $\lambda_0 = 5.0\mathrm{E}-7, \sigma = 5.0\mathrm{E}-5, \overline{\lambda_0} = 0$. Set $\mathbf{r}_0 = \mathbf{p}_0 = -\nabla E_T\left(\mathbf{w}_0\right)$, and *success=true*.

2. If *success=true*, find the second-order information: $\sigma_k = \frac{\sigma}{|\mathbf{p}_k|}$, $\mathbf{s}_k = \frac{\nabla E_T(\mathbf{w}_k+\sigma_k\mathbf{p}_k)-\nabla E_T(\mathbf{w}_k)}{\sigma_k}$, $\delta_k = \mathbf{p}_k^t\mathbf{s}_k$, where the superscript *t* denotes the transpose.

3. Set $\delta_k = \delta_k + \left(\lambda_k - \overline{\lambda_k}\right)|\mathbf{p}_k|^2$.
4. If $\delta_k \leq 0$, make the Hessian positive definite: $\overline{\lambda_k} = 2\left(\lambda_k - \frac{\delta_k}{|\mathbf{p}_k|^2}\right)$, $\delta_k = -\delta_k + \lambda_k|\mathbf{p}_k|^2$, $\lambda_k = \overline{\lambda_k}$.
5. Calculate the step size: $\mu_k = \mathbf{p}_k^t\mathbf{r}_k$, $\alpha_k = \frac{\mu_k}{\delta_k}$.
6. Calculate the comparison parameter: $\Delta_k = 2\delta_k\frac{[E_T(\mathbf{w}_k) - E_T(\mathbf{w}_k + \alpha_k\mathbf{p}_k)]}{\mu_k}$.
7. If $\Delta_k \geq 0$, error can be reduced. Set $\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k\mathbf{p}_k$, $\mathbf{r}_{k+1} = -\nabla E_T\left(\mathbf{w}_{k+1}\right)$, If $|\mathbf{r}_{k+1}| < 10^{-6}$, return $\mathbf{w}_{k+1}$ as the final weight vector and exit. Set $\overline{\lambda_k} = 0$, *success=true*. If *k* mod *N*=0, restart the algorithm: $\mathbf{p}_{k+1} = \mathbf{r}_{k+1}$, else set $\beta_k = \frac{|\mathbf{r}_{k+1}|^2 - \mathbf{r}_{k+1}^t\mathbf{r}_k}{\mu_k}$, $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$. If $\Delta_k \geq .75$, reduce the scale parameter: $\lambda_k = \frac{1}{4}\lambda_k$. else (if $\Delta_k < 0$): Set $\overline{\lambda_k} = \lambda_k$, *success=false*.
8. If $\Delta_k < .25$, increase the scale parameter: $\lambda_k = \lambda_k + \frac{\delta_k(1-\Delta_k)}{|\mathbf{p}_k|^2}$.
9. If *success=false*, return to step 2. Otherwise if a stopping rule is met, exit and report the network as stated in the stopping criteria. Else set *k*=*k*+1 , $\overline{\lambda_{k+1}} = \overline{\lambda_k}$, $\lambda_{k+1} = \lambda_k$ and return to step 2.

   Note: each iteration requires at least two data passes.

Stopping Rules

Training proceeds through at least one complete pass of the data. Then the search should be stopped according to following criteria. These stopping criteria should be checked in the listed order. When creating a new model, check after completing an iteration. During a model update, check criteria 1, 3, 4, 5 and 6 is after completing a data pass, and only check criterion 2 after an iteration. In the descriptions below, a "step" means an iteration when building a new model and a data pass when performing a model update. Let E_1 denote the current minimum error and K_1 denote the iteration where it occurs for the training set, E_2 and K_2 are that for the overfit prevention set, and *K*3=min(K_1,K_2).

1. At the end of each step compute the total error for the overfit prevention set. From step K_2, if the testing error does not decrease below E_2 over the next *n*=1 steps, stop. Report the weights at step K_2. If there is no overfit prevention set, this criterion is not used for building a new model; for a model update when there is no overfit prevention set, compute the total error for training data at the end of each step. From step K_1, if the training error does not decrease below E_1 over the next *n*=1 steps, stop. Report the weights at step K_1.
2. The search has lasted beyond some maximum allotted time. For building a new model, simply report the weights at step K_3. For a model update, even though training stops before the completion of current step, treat this as a complete step. Calculate current errors for training and testing datasets and update E_1, K_1, E_2, K_2 correspondingly. Report the weights at step K_3.
3. The search has lasted more than some maximum number of data passes. Report the weights at step K_3.
4. Stop if the relative change in training error is small: $\frac{|E_T(w_k) - E_T(w_{k-1})|}{\frac{1}{2}(E_T(w_k) + E_T(w_{k-1}) + \delta)} < \epsilon_1$ for $\delta = 10^{-10}$ and $\epsilon_1 = 10^{-4}$, where w_{k-1}, w_k are the weight vectors of two consecutive steps. Report weights at step K_3.

5. The current training error ratio is small compared with the initial error: $\left|\frac{E_T(w_k)}{\overline{E}_T+\delta}\right| < \epsilon_2$ for $\delta = 10^{-10}$ and $\epsilon_2 = 10^{-3}$, where $\overline{E}_T$ is the total error from the model using the average of an output field to predict that field; $\overline{E}_T$ is calculated by using $a^m_{I:r} = \frac{1}{M}\sum_{m=1}^{M} y_r^{(m)}$ in the error function, where w_k is the weight vector of one step. Report weights at step K_3.

6. The current accuracy meets a specified threshold. Accuracy is computed based on the overfit prevention set if there is one, otherwise the training set.

Note: In criteria 4 and 5, the total error for whole training data is needed. For model updates, these criteria will not be checked if there is an overfit prevention set.

Model Updates

When new records become available, the synaptic weights can be updated. The new records are split into groups of the size *R* = min(*M*,2*N*,1000), where *M* is the number of training records and *N* is the number of weights in the network. A single data pass is made through the new groups to update the weights. If the last of the new groups has more than one-quarter of the records of a normal group, then it is processed normally; otherwise, it remains in the internal buffer so that these records can be used during the next update. Thus, after the last update there may be some unused records remaining in the buffer that will be lost.

Radial Basis Function

A radial basis function (RBF) network is a feed-forward, supervised learning network with only one hidden layer, called the radial basis function layer. The RBF network is a function of one or more predictors that minimizes the prediction error of one or more targets. Predictors and targets can be a mix of categorical and continuous fields.

Notation

The following notation is used throughout this chapter unless otherwise stated:

$X^{(m)} = \left(x_1^{(m)}, ..., x_P^{(m)}\right)$	Input vector, pattern *m*, *m*=1,...*M*.
$Y^{(m)} = \left(y_1^{(m)}, ..., y_R^{(m)}\right)$	Target vector, pattern *m*.
I	Number of layers, discounting the input layer. For an RBF network, *I*=2.
J_i	Number of units in layer *i*. J_0 = *P*, J_{i} = *R*, discounting the bias unit. J_1 is the number of RBF units.
$\phi_j\left(X^{(m)}\right)$	*j*th RBF unit for input $X^{(m)}$, *j*=1, ...,J_1.
μ_j	center of ϕ_j, it is *P*-dimensional.
σ_j	width of ϕ_j, it is *P*-dimensional.
h	the RBF overlapping factor.

$a^m_{i:j}$	Unit *j* of layer *i*, pattern *m*, $j = 0, ..., J_i; i = 0, ..., I$.
w_{rj}	weight connecting *r*th output unit and *j*th hidden unit of RBF layer.

Architecture

There are three layers in the RBF network:

Input layer: J_0=*P* units, $a_{0:1}, \cdots, a_{0:J_0}$; with $a_{0:j} = x_j$.

RBF layer: J_1 units, , $a_{1:1}, \cdots, a_{1:J_1}$; with $a_{1:j} = \phi_j\left(X\right)$ and $\phi_j\left(X\right) = \exp\left(-\sum_{p=1}^{P} \frac{1}{2\sigma^2_{jp}}\left(x_p - \mu_{jp}\right)^2\right) / \sum_{j=1}^{J_1} \exp\left(-\sum_{p=1}^{P} \frac{1}{2\sigma^2_{jp}}\left(x_p - \mu_{jp}\right)^2\right)$.

Output layer: J_2=*R* units, $a_{I:1}, \cdots, a_{I:J_2}$; with $a_{I:r} = w_{r0} + \sum_{j=1}^{J_1} w_{rj}\phi_j\left(X\right)$.

Error Function

Sum-of-squares error is used:

$$E_T\left(w\right) = \sum_{m=1}^{M} E_m\left(w\right)$$

where

$$E_m\left(w\right) = \frac{1}{2}\sum_{r=1}^{R}\left(y_r^{(m)} - a^m_{I:r}\right)^2$$

The sum-of-squares error function with identity activation function for output layer can be used for both continuous and categorical targets. For continuous targets, $a^m_{I:r}$ approximates the conditional expectation of the target value $E\left(y_r|X^{(m)}\right)$. For categorical targets, $a^m_{I:r}$ approximates the posterior probability of class *k*: $P\left(y_r = 1|X^{(m)}\right)$.

Note: though $\Sigma a^m_{I:r} = 1$ (the sum is over all classes of the same categorical target field), $a^m_{I:r}$ may not lie in the range [0, 1].

Training

The network is trained in two stages:

1. **Determine the basis functions by clustering methods.** The center and width for each basis function is computed.
2. **Determine the weights given the basis functions.** For the given basis functions, compute the ordinary least-squares regression estimates of the weights.

The simplicity of these computations allows the RBF network to be trained very quickly.

Determining Basis Functions

The two-step clustering algorithm is used to find the RBF centers and widths. For each cluster, the mean and standard deviation for each continuous field and proportion of each category for each categorical field are derived. Using the results from clustering, the center of the *j*th RBF is set as:

$$\mu_{jp} = \begin{cases} \overline{x}_{jp} & \text{if pth field is continuous} \\ \pi_{jp} & \text{if pth field is a dummy field of a categorical field} \end{cases}$$

where $\overline{x}_{jp}$ is the *j*th cluster mean of the *p*th input field if it is continuous, and π_{jp} is the proportion of the category of a categorical field that the *p*th input field corresponds to. The width of the *j*th RBF is set as

$$\sigma_{jp} = h^{1/2} \begin{cases} s_{jp} & \text{if pth field is continuous} \\ \sqrt{p_{jp}\left(1-p_{jp}\right)} & \text{if pth field is a dummy field of a categorical field} \end{cases}$$

where s_{jp} is the *j*th cluster standard deviation of the *p*th field and *h*>0 is the RBF overlapping factor that controls the amount of overlap among the RBFs. Since some σ_{jp} may be zeros, we use spherical shaped Gaussian bumps; that is, a common width

$$\sigma_j = \sqrt{\frac{1}{P}\sum_{p=1}^{P} \sigma_{jp}^2}$$

in for all predictors. In the case that σ_j is zero for some *j*, set it to be $\min\left\{\sigma_j : \sigma_j \neq 0,\right\}_{j=1}^{J_1}$. If all σ_j are zero, set all of them to be $\sqrt{h}$.

When there are a large number of predictors, $\sum_{p=1}^{P}\left(x_p-\mu_{jp}\right)^2$ could be easily very large and hence $\exp\left(-\sum_{p=1}^{P}\frac{1}{2\sigma_j^2}\left(x_p-\mu_{jp}\right)^2\right)$ is practically zero for every record and every RBF unit if σ_j is relatively small. This is especially bad for ORBF because there would be only a constant term in the model when this happens. To avoid this, σ_j is increased by setting the default overlapping factor *h* proportional to the number of inputs: *h*=1 + 0.1 *P*.

Automatic Selection of Number of Basis Functions

The algorithm tries a reasonable range of numbers of hidden units and picks the “best”. By default, the reasonable range [K_1, K_2] is determined by first using the two-step clustering method to automatically find the number of clusters, *K*. Then set K_1 = min(*K*, *R*) for ORBF and K_1 =max{2, min(*K*, *R*)} for NRBF and K_2=max(10, 2*K*, *R*).

If a test data set is specified, then the “best” model is the one with the smaller error in the test data. If there is no test data, the BIC (Bayesian information criterion) is used to select the “best” model. The BIC is defined as

$$BIC = MR\ln\left(MSE\right) + k\ln\left(M\right)$$

where $MSE = \frac{1}{MR}\sum_{m=1}^{M}\sum_{r=1}^{R}\left(y_r^{(m)} - a_{I:r}^{m}\right)^2$ is the mean squared error and k= (P+1+R)J_1 for NRBF and (P+1+R)J_1+R for ORBF is the number of parameters in the model.

Model Updates

When new records become available, you can update the weights connecting the RBF layer and output layer. Again, given the basis functions, updating the weights is a least-squares regression problem. Thus, it is very fast.

For best results, the new records should have approximately the same distribution as the original records.

Missing Values

The following options for handling missing values are available:

- Records with missing values are excluded listwise.
- Missing values are imputed. Continuous fields impute the average of the minimum and maximum observed values; categorical fields impute the most frequently occurring category.

Output Statistics

The following output statistics are available. Note that, for continuous fields, output statistics are reported in terms of the rescaled values of the fields.

Accuracy

For continuous targets, it is

$$R^2 = 1 - \frac{\Sigma_{k=1}^{K} f_k\left(y_k - \hat{y}_k\right)^2}{\Sigma_{k=1}^{K} f_k\left(y_k - \overline{y}\right)^2}$$

where $\overline{y} = \frac{1}{N}\Sigma_{k=1}^{K} f_k y_k$

Note that R^2 can never be greater than one, but can be less than zero.

For the naïve model, $\hat{y}_k$ is the modal category for categorical targets and the mean for continuous targets.

For each categorical target, this is the percentage of records for which the predicted value matches the observed value.

Predictor Importance

For more information, see the topic "Predictor Importance Algorithms" on p. 311.

Confidence

Confidence values for neural network predictions are calculated based on the type of output field being predicted. Note that no confidence values are generated for numeric output fields.

Difference

The difference method calculates the confidence of a prediction by comparing the best match with the second-best match as follows, depending on output field type and encoding used.

- **Flag fields.** Confidence is calculated as $c = 2 \cdot |0.5 - o|$, where o is the output activation for the output unit.
- **Set fields.** With the standard encoding, confidence is calculated as $c = o_1 - o_2$, where o_1 is the output unit in the fields group of units with the highest activation, and o_2 is the unit with the second-highest activation.

  With binary set encoding, the sum of the errors comparing the output activation and the encoded set value is calculated for the closest and second-closest matches, and the confidence is calculated as $c = e_2 - e_1$, where e_2 is the error for the second-best match and e_1 is the error for the best match.

Simplemax

Simplemax returns the highest predicted probability as the confidence.

References

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*, 3rd ed. Oxford: Oxford University Press.

Fine, T. L. 1999. *Feedforward Neural Network Methodology*, 3rd ed. New York: Springer-Verlag.

Haykin, S. 1998. *Neural Networks: A Comprehensive Foundation*, 2nd ed. New York: Macmillan College Publishing.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Tao, K. K. 1993. A closer look at the radial basis function (RBF) networks. In: *Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers,* A. Singh, ed. Los Alamitos, Calif.: IEEE Comput. Soc. Press, 401–405.

Uykan, Z., C. Guzelis, M. E. Celebi, and H. N. Koivo. 2000. Analysis of input-output clustering for determining centers of RBFN. *IEEE Transactions on Neural Networks*, 11, 851–858.

OPTIMAL BINNING Algorithms

The Optimal Binning procedure performs MDLP (minimal description length principle) discretization of scale variables. This method divides a scale variable into a small number of intervals, or bins, where each bin is mapped to a separate category of the discretized variable.

MDLP is a univariate, supervised discretization method. Without loss of generality, the algorithm described in this document only considers one continuous attribute in relation to a categorical guide variable — the discretization is "optimal" with respect to the categorical guide. Therefore, the input data matrix *S* contains two columns, the scale variable *A* and categorical guide *C*.

Optimal binning is applied in the Binning node when the binning method is set to Optimal.

Notation

The following notation is used throughout this chapter unless otherwise stated:

S	The input data matrix, containing a column of the scale variable *A* and a column of the categorical guide *C*. Each row is a separate observation, or instance.
A	A scale variable, also called a continuous attribute.
$S(i)$	The value of *A* for the *i*th instance in *S*.
N	The number of instances in *S*.
D	A set of all distinct values in *S*.
S_{i}	A subset of *S*.
C	The categorical guide, or class attribute; it is assumed to have *k* categories, or classes.
T	A cut point that defines the boundary between two bins.
T_{A}	A set of cut points.
$\mathrm{Ent}(S)$	The class entropy of *S*.
$\mathrm{E}(A, T, S)$	The class entropy of partition induced by *T* on *A*.
$\mathrm{Gain}(A, T, S)$	The information gain of the cut point *T* on *A*.
n	A parameter denoting the number of cut points for the equal frequency method.
W	A weight attribute denoting the frequency of each instance. If the weight values are not integer, they are rounded to the nearest whole numbers before use. For example, 0.5 is rounded to 1, and 2.4 is rounded to 2. Instances with missing weights or weights less than 0.5 are not used.

Simple MDLP

This section describes the supervised binning method (MDLP) discussed in Fayyad and Irani (1993).

Class Entropy

Let there be *k* classes $C_1, ..., C_{\mathrm{k}}$ and let $P(C_{\mathrm{i}}, S)$ be the proportion of instances in *S* that have class C_{i}. The class entropy Ent(*S*) is defined as

$$Ent\left(S\right) = -\sum_{i=1}^{k} P\left(C_i, S\right)\log_2\left(P\left(C_i, S\right)\right)$$

Class Information Entropy

For an instance set S, a continuous attribute A, and a cut point T, let $S_1 \subset S$ be the subset of instances in S with the values of $A \leq T$, and $S_2 = S - S_1$. The class information entropy of the partition induced by T, E(A, T; S), is defined as

$$E\left(A, T; S\right) = \frac{\left|S_1\right|}{\left|S\right|}Ent\left(S_1\right) + \frac{\left|S_2\right|}{\left|S\right|}Ent\left(S_2\right)$$

Information Gain

Given a set of instances S, a continuous attribute A, and a cut point T on A, the information gain of a cut point T is

$$Gain\left(A, T; S\right) = Ent\left(S\right) - E\left(A, T; S\right)$$

MDLP Acceptance Criterion

The partition induced by a cut point T for a set S of N instances is accepted if and only if

$$Gain\left(A, T; S\right) > \frac{\log_2\left(N-1\right)}{N} + \frac{\Delta\left(A, T; S\right)}{N}$$

and it is rejected otherwise.

Here $\Delta\left(A, T; S\right) = \log_2\left(3^k - 2\right) - \left[k \cdot Ent\left(S\right) - k_1 Ent\left(S_1\right) - k_2 Ent\left(S_2\right)\right]$ in which k_i is the number of classes in the subset S_i of S.

Note: While the MDLP acceptance criterion uses the association between A and C to determine cut points, it also tries to keep the creation of bins to a small number. Thus there are situations in which a high association between A and C will result in no cut points. For example, consider the following data:

D	*Class*	
	2	3
1	1	0
2	0	6

Then the potential cut point is $T = 1$. In this case:

$$Gain\left(A, T; S\right) = 0.5916728$$

$$\frac{\log_2(N-1)}{N}+\frac{\Delta(A,T;S)}{N}=0.6530774$$

Since 0.5916728 < 0.6530774, T is not accepted as a cut point, even though there is a clear relationship between A and C.

Algorithm: BinaryDiscretization

1. Calculate E(A, d_i; S) for each distinct value $d_i \in D$ for which d_i and d_{i+1} do not belong to the same class. A distinct value belongs to a class if all instances of this value have the same class.
2. Select a cut point T for which E(A, T; S) is minimum among all the candidate cut points, that is,

   $T = arg\,\min_{d_i} E(A, d_i; S)$

Algorithm: MDLPCut

1. BinaryDiscretization(A, T; D, S).
2. Calculate Gain(A, T; S).
3. If $Gain(A,T;S) > \frac{\log_2(N-1)}{N}+\frac{\Delta(A,T;S)}{N}$ then

   a) $T_A = T_A \cup T$.

   b) Split D into D_1 and D_2, and S into S_1 and S_2.

   c) MDLPCut(A, T_A; D_1, S_1).

   d) MDLPCut(A, T_A; D_2, S_2). where $S_1 \subset S$ be the subset of instances in S with A-values ≤ T, and $S_2 = S–S_1$. D_1 and D_2 are the sets of all distinct values in S_1 and S_2, respectively.

Also presented is the iterative version of MDLPCut(A, T_A; D, S). The iterative implementation requires a stack to store the D and S remaining to be cut.

First push D and S into *stack*. Then, while (*stack*≠∅) do

1. Obtain D and S by popping *stack*.
2. BinaryDiscretization(A, T; D, S).
3. Calculate Gain(A, T; S).
4. If $Gain(A,T;S) > \frac{\log_2(N-1)}{N}+\frac{\Delta(A,T;S)}{N}$ then

   i) $T_A = T_A \cup T$.

   ii) Split D into D_1 and D_2, and S into S_1 and S_2.

   iii) Push D_1 and S_1 into *stack*.

   iv) Push D_2 and S_2 into *stack*.

Note: In practice, all operations within the algorithm are based on a global matrix M. Its element, m_{ij}, denotes the total number of instances that have value $d_i \in D$ and belong to the jth class in S. In addition, D is sorted in ascending order. Therefore, we do not need to push D and S into *stack*, but only two integer numbers, which denote the bounds of D, into *stack*.

Algorithm: SimpleMDLP

1. Sort the set S with N instances by the value A in ascending order.
2. Find a set of all distinct values, D, in S.
3. $T_A = \varnothing$.
4. MDLPCut(A, T_A; D, S)
5. Sort the set T_A in ascending order, and output T_A.

Hybrid MDLP

When the set D of distinct values in S is large, the computational cost to calculate E(A, d_i; S) for each $d_i \in D$ is large. In order to reduce the computational cost, the unsupervised equal frequency binning method is used to reduce the size of D and obtain a subset $D_{ef} \in D$. Then the MDLPCut(A, T_A; D_s, S) algorithm is applied to obtain the final cut point set T_A.

Algorithm: EqualFrequency

It divides a continuous attribute A into n bins where each bin contains N/n instances. n is a user-specified parameter, where $1 < n < N$.

1. Sort the set S with N instances by the value A in ascending order.
2. $D_{ef} = \varnothing$.
3. j=1.
4. Use the aempirical percentile method to generate the $d_{p,i}$ which denote the $\left(\frac{i \cdot N}{n} \times 100\right)$th percentiles.
5. $D_{ef} = D_{ef} \cup d_{p,i}$; i=i+1
6. If $i \le n$, then go to step 4.
7. Delete the duplicate values in the set D_{ef}.

Note: If, for example, there are many occurrences of a single value of A, the equal frequency criterion may not be met. In this case, no cut points are produced.

Algorithm: HybridMDLP

1. $D = \varnothing$;

2. EqualFrequency(A, n, D; S).
3. $T_A = \varnothing$.
4. MDLPCut(A, T_A; D, S).
5. Output T_A.

Model Entropy

The model entropy is a measure of the predictive accuracy of an attribute A binned on the class variable C. Given a set of instances S, suppose that A is discretized into I bins given C, where the ith bin has the value A_i. Letting $S_i \subset S$ be the subset of instances in S with the value A_i, the model entropy is defined as:

$$E_m = \sum_{i=1}^{I} P\left(A_i\right)\left(-\sum_{j=1}^{J} P\left(C_j|A_i\right)\log_2 P\left(C_j|A_i\right)\right)$$

where $P\left(A_i\right) = \frac{|S_i|}{|S|}$ and $P\left(C_j|A_i\right) = \frac{P(C_j, A_i)}{P(A_i)} = P\left(C_j, S_i\right)$.

Merging Sparsely Populated Bins

Occasionally, the procedure may produce bins with very few cases. The following strategy deletes these pseudo cut points:

- For a given variable, suppose that the algorithm found n_{final} cut points, and thus n_{final}+1 bins. For bins i = 2, ..., n_{final} (the second lowest-valued bin through the second highest-valued bin), compute

  $\frac{sizeof(b_i)}{\min(sizeof(b_{i-1}), sizeof(b_{i+1}))}$

  where sizeof(bin) is the number of cases in the bin.

- When this value is less than a user-specified merging threshold, b_i is considered sparsely populated and is merged with b_{i-1} or b_{i+1}, whichever has the lower class information entropy. For more information, see the topic “Class Information Entropy ” on p. 306.

The procedure makes a single pass through the bins.

Blank Handling

In optimal binning, blanks are handled in pairwise fashion. That is, for every pair of fields {binning field, target field}, all records with valid values for both fields are used to bin that specific binning field, regardless of any blanks that may exist in other fields to be binned.

References

Fayyad, U., and K. Irani. 1993. Multi-interval discretization of continuous-value attributes for classification learning. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* San Mateo, CA: Morgan Kaufmann, 1022–1027.

Dougherty, J., R. Kohavi, and M. Sahami. 1995. Supervised and unsupervised discretization of continuous features. In: *Proceedings of the Twelfth International Conference on Machine Learning,* Los Altos, CA: Morgan Kaufmann, 194–202.

Liu, H., F. Hussain, C. L. Tan, and M. Dash. 2002. Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, 6, 393–423.

Predictor Importance Algorithms

Predictor importance can be determined by computing the reduction in variance of the target attributable to each predictor, via a sensitivity analysis. This method of computing predictor importance is used in the following models:

- Neural Networks
- C5.0
- C&RT
- QUEST
- CHAID
- Regression
- Logistic
- Discriminant
- GenLin
- SVM
- Bayesian Networks

Notation

The following notation is used throughout this chapter unless otherwise stated:

Y	Target
X_j	Predictor, where *j*=1,...,*k*
k	The number of predictors
$Y = f(X_1, X_2, \ldots, X_k)$	Model for *Y* based on predictors X_1 through X_k

Variance Based Method

Predictors are ranked according to the sensitivity measure defined as follows.

$$S_i = \frac{V_i}{V(Y)} = \frac{V(E(Y|X_i))}{V(Y)}$$

where *V*(*Y*) is the unconditional output variance. In the numerator, the expectation operator *E* calls for an integral over X_{-i}; that is, over all factors but X_i, then the variance operator *V* implies a further integral over X_i.

Predictor importance is then computed as the normalized sensitivity.

$$VI_i = \frac{S_i}{\sum_{j=1}^{k} S_j}$$

Saltelli et al (2004) show that S_i is the proper measure of sensitivity to rank the predictors in order of importance for any combination of interaction and non-orthogonality among predictors.

The importance measure S_{i} is the first-order sensitivity measure, which is accurate if the set of the input factors (X_1 , X_2 ,…, X_{k}) is orthogonal/independent (a property of the factors), and the model is additive; that is, the model does not include interactions (a property of the model) between the input factors. For any combination of interaction and non-orthogonality among factors, Saltelli (2004) pointed out that S_{i} is still the proper measure of sensitivity to rank the input factors in order of importance, but there is a risk of inaccuracy due to the presence of interactions or/and non-orthogonality. For better estimation of S_{i}, the size of the dataset should be a few hundred at least. Otherwise, S_{i} may be biased heavily. In this case, the importance measure can be improved by bootstrapping.

Computation

In the orthogonal case, it is straightforward to estimate the conditional variances V_i by computing the multidimensional integrals in the space of the input factors, via Monte Carlo methods as follows.

Let us start with two input sample matrices $\mathbf{M}_1$ and $\mathbf{M}_2$, each of dimension $N \times k$:

$$\mathbf{M}_1 = \begin{matrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_k^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_k^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_k^{(N)} \end{matrix}$$

and

$$\mathbf{M}_2 = \begin{matrix} x_1^{(1')} & x_2^{(1')} & \cdots & x_k^{(1')} \\ x_1^{(2')} & x_2^{(2')} & \cdots & x_k^{(2')} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N')} & x_2^{(N')} & \cdots & x_k^{(N')} \end{matrix}$$

where N is the sample size of the Monte Carlo estimate which can vary from a few hundred to one thousand. Each row is an input sample. From $\mathbf{M}_1$ and $\mathbf{M}_2$, we can build a third matrix $\mathbf{N}_j$.

$$\mathbf{N}_j = \begin{matrix} x_1^{(1')} & x_2^{(1')} & \cdots & x_j^{(1)} & \cdots & x_k^{(1')} \\ x_1^{(2')} & x_2^{(2')} & \cdots & x_j^{(2)} & \cdots & x_k^{(2')} \\ \vdots & \vdots & \ddots & & \ddots & \vdots \\ x_1^{(N')} & x_2^{(N')} & \cdots & x_j^{(N)} & \cdots & x_k^{(N')} \end{matrix}$$

We may think of $\mathbf{M}_1$ as the "sample" matrix, $\mathbf{M}_2$ as the "resample" matrix, and $\mathbf{N}_j$ as the matrix where all factors except X_j are resampled. The following equations describe how to obtain the variances (Saltelli 2002). The 'hat' denotes the numeric estimates.

$$\hat{V}(Y) = \frac{1}{N-1}\sum_{r=1}^{N} f^2\left(x_1^{(r)}, x_2^{(r)}, \dots, x_k^{(r)}\right) - \hat{E}^2(Y)$$

where

$$\hat{E}^2(Y) = \left[\frac{1}{N}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \dots, x_k^{(r)}\right)\right]^2$$

$$\hat{V}(E(Y|X_j)) = \hat{U}_j - \hat{E}^2(Y)$$

where

$$\hat{U}_j = \frac{1}{N-1}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \dots, x_k^{(r)}\right) f\left(x_1^{(r')}, x_2^{(r')}, \dots, , x_{(j-1)}^{(r')}, x_j^{(r)}, x_{(j+1)}^{(r')}, \dots, x_k^{(r')}\right)$$

and

$$\hat{E}^2(Y) = \frac{1}{N}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \dots, x_k^{(r)}\right) f\left(x_1^{(r')}, x_2^{(r')}, \dots, x_k^{(r')}\right)$$

When the target is continuous, we simply follow the accumulation steps of variance and expectations. For a categorical target, the accumulation steps are for each category of *Y*. For each input factor, S_i is a vector with an element for each category of *Y*. The average of elements of S_i is used as the estimation of importance of the *i*th input factor on *Y*.

Convergence. In order to improve scalability, we use a subset of the records and predictors when checking for convergence. Specifically, the convergence is judged by the following criteria:

$$\underset{i \in I}{\cap} \frac{1}{D} \sum_{j=t-D+1}^{t} \frac{\left|S_i(j) - \overline{S}_i\right|}{\overline{S}_i} < \epsilon$$

where $I = \{i|S_i(t) > 1/num\}$, *D*=100 and denotes the width of interest, $\overline{S}_i = \frac{1}{D} \sum_{j=t-D+1}^{t} S_i(j)$, and $\epsilon = 0.005$ defines the desired average relative error.

This specification focuses on “good” predictors; those whose importance values are larger than average.

Record order. This method of computing predictor importance is desirable because it scales well to large datasets, but the results are dependent upon the order of records in the dataset. However, with large, randomly ordered datasets, you can expect the predictor importance results to be consistent.

References

Saltelli, A., S. Tarantola, F. , F. Campolongo, and M. Ratto. 2004. *Sensitivity Analysis in Practice – A Guide to Assessing Scientific Models*. : John Wiley.

Saltelli, A. 2002. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145:2, 280–297.

QUEST Algorithms

Overview of QUEST

QUEST stands for Quick, Unbiased, Efficient Statistical Tree. It is a relatively new binary tree-growing algorithm (Loh and Shih, 1997). It deals with split field selection and split-point selection separately. The univariate split in QUEST performs approximately unbiased field selection. That is, if all predictor fields are equally informative with respect to the target field, QUEST selects any of the predictor fields with equal probability.

QUEST affords many of the advantages of C&RT, but, like C&RT, your trees can become unwieldy. You can apply automatic cost-complexity pruning (see "Pruning" on p. 323) to a QUEST tree to cut down its size. QUEST uses surrogate splitting to handle missing values. For more information, see the topic "Blank Handling" on p. 319.

Primary Calculations

The calculations directly involved in building the model are described below.

Frequency Weight Fields

A **frequency field** represents the total number of observations represented by each record. It is useful for analyzing aggregate data, in which a record represents more than one individual. The sum of the values for a frequency field should always be equal to the total number of observations in the sample. Note that output and statistics are the same whether you use a frequency field or case-by-case data. The table below shows a hypothetical example, with the predictor fields *sex* and *employment* and the target field *response*. The frequency field tells us, for example, that 10 employed men responded *yes* to the target question, and 19 unemployed women responded *no*.

Table 30-1
Dataset with frequency field

Sex	Employment	Response	Frequency
M	Y	Y	10
M	Y	N	17
M	N	Y	12
M	N	N	21
F	Y	Y	11
F	Y	N	15
F	N	Y	15
F	N	N	19

The use of a frequency field in this case allows us to process a table of 8 records instead of case-by-case data, which would require 120 records.

QUEST does not support the use of case weights.

Model Parameters

QUEST deals with field selection and split-point selection separately. Note that you can specify the alpha level to be used in the Expert Options for QUEST—the default value is $\alpha_{\text{nominal}} = 0.05$.

Field Selection

1. For each predictor field *X*, if *X* is a symbolic (categorical), or nominal, field, compute the *p* value of a Pearson chi-square test of independence between *X* and the dependent field. If *X* is scale-level (continuous), or ordinal field, use the *F* test to compute the *p* value.

2. Compare the smallest *p* value to a prespecified, Bonferroni-adjusted alpha level α_B.
   - If the smallest *p* value is less than α_B, then select the corresponding predictor field to split the node. Go on to step 3.
   - If the smallest *p* value is *not* less thanα_B, then for each *X* that is scale-level (continuous), use Levene's test for unequal variances to compute a *p* value. (In other words, test whether *X* has unequal variances at different levels of the target field.)
   - Compare the smallest *p* value from Levene's test to a new Bonferroni-adjusted alpha level α_L.
   - If the *p* value is less than α_L, select the corresponding predictor field with the smallest *p* value from Levene's test to split the node.
   - If the *p* value is greater than α_L, the node is not split.

Split Point Selection—Scale-Level Predictor

1. If *Y* has only two categories, skip to the next step. Otherwise, group the categories of *Y* into two superclasses as follows:
   - Compute the mean of *X* for each category of *Y*.
   - If all means are the same, the category with the largest weighted frequency is selected as one superclass and all other categories are combined to form the other superclass. (If all means are the same and there are multiple categories tied for largest weighted frequency, select the category with the smallest index as one superclass and combine the other categories to form the other.)
   - If the means are not all the same, apply a two-mean clustering algorithm to those means to obtain two superclasses of *Y*, with the initial cluster centers set at the two most extreme class means. (This is a special case of *k*-means clustering, where $k = 2$. For more information, see the topic "Overview" on p. 227.)

2. Apply quadratic discriminant analysis (QDA) to determine the split point. Notice that QDA usually produces two cut-off points—choose the one that is closer to the sample mean of the first superclass.

Split Point Selection—Symbolic (Categorical) Predictor

QUEST first transforms the symbolic field into a continuous field ξ by assigning discriminant coordinates to categories of the predictor. The derived field ξ is then split as if it were any other continuous predictor as described above.

Chi-Square Test

The Pearson chi-square statistic is calculated as

$$X^2 = \sum_{j=1}^{J} \sum_{i=1}^{I} \frac{(n_{ij} - \hat{m}_{ij})^2}{\hat{m}_{ij}}$$

where $n_{ij} = \sum_n f_n I(x_n = i \wedge y_n = j)$ is the observed cell frequency and $\hat{m}_{ij}$ is the expected cell frequency for cell ($x_n = i$, $y_n = j$) from the independence model as described below. The corresponding p value is calculated as $p = \Pr\left(\chi^2_d > X^2\right)$, where χ^2_d follows a chi-square distribution with $d = (J - 1)(I - 1)$ degrees of freedom.

Expected Frequencies for Chi-Square Test

For models with no case weights, expected frequencies are calculated as

$$\hat{m}_{ij} = \frac{n_{i.} n_{.j}}{n_{..}}$$

where

$$n_{i.} = \sum_{j=1}^{J} n_{ij},\ n_{.j} = \sum_{i=1}^{I} n_{ij},\ n_{..} = \sum_{j=1}^{J} \sum_{i=1}^{I} n_{ij}.$$

F Test

Suppose for node t there are J_t classes of target field Y. The F statistic for continuous predictor X is calculated as

$$F_X = \frac{\sum_{j=1}^{J_t} N_{f,j}(t)\left(\overline{x}^{(j)}(t) - \overline{x}(t)\right)^2 / (J_t - 1)}{\sum_{i \in t} f_i\left(x_i - \overline{x}^{(y_n)}(t)\right)^2 / \left(N_f(t) - J_t\right)}$$

where

$$\overline{x}^{(j)}(t) = \frac{\sum_{i \in t} f_n x_n I(y_n = j)}{N_{f,j}(t)}, \overline{x}(t) = \frac{\sum_{i \in t} f_n x_n}{N_f(t)}$$

The corresponding p value is given by

$$p_X = \Pr\left(F\left(J_t - 1, N_f(t) - J_t\right) > F_X\right)$$

where $F(J_t - 1, N_f(t) - J_t)$ follows an F distribution with degrees of freedom $J_t - 1$ and $N_f(t) - J_t$.

Levene's Test

For continuous predictor *X*, calculate $z_n = \left|x_n - \overline{x}^{(y_n)}(t)\right|$, where is the mean of *X* for records in node *t* with target value y_n. Levene's *F* statistic for predictor *X* is the ANOVA *F* statistic for z_n.

Bonferroni Adjustment

The adjusted alpha level α_B is calculated as the nominal value divided by the number of possible comparisons.

For QUEST, the Bonferroni adjusted alpha level α_B for the initial predictor selection is

$$\alpha_B = \frac{\alpha_{nominal}}{m}$$

where *m* is the number of predictor fields in the model.

For the Levene test, the Bonferroni adjusted alpha level α_L is

$$\alpha_L = \frac{\alpha_{nominal}}{m + m_c}$$

where m_c is the number of continuous predictor fields.

Discriminant Coordinates

For categorical predictor *X* with values $\{b_1,...,b_I\}$, QUEST assigns a score value from a continuous variable ξ to each category of *X*. The scores assigned are chosen to maximize the ratio of between-class to within-class sum of squares of ξ for the target field classes:

For each record, transform *X* into a vector of dummy fields $\mathbf{g} = (g_1, ..., g_I)^{'}$, where

$$g_i = \begin{cases} 1 & x = b_i \\ 0 & \text{otherwise} \end{cases}$$

Calculate the overall and class *j* mean of ν:

$$\overline{\mathbf{g}} = \frac{\sum_n f_n \mathbf{g}_n}{N_f}, \overline{\mathbf{g}}^{(j)} = \frac{\sum_n f_n \mathbf{g}_n I(y_n = j)}{N_{f,j}}$$

where f_n is the frequency weight for record *n*, g_n is the dummy vector for record *n*, N_f is the total sum of frequency weights for the training data, and $N_{f,j}$ is the sum of frequency weights for records with category *j*.

Calculate the following $I \times I$ matrices:

$$\mathbf{B} = \sum_{j=1}^{J} N_{f,j} \left(\overline{\mathbf{g}}^{(j)} - \overline{\mathbf{g}}\right)\left(\overline{\mathbf{g}}^{(j)} - \overline{\mathbf{g}}\right)'$$

$$\mathbf{T} = \sum_n f_n(\mathbf{g}_n - \overline{\mathbf{g}})(\mathbf{g}_n - \overline{\mathbf{g}})^{'}$$

Perform singular value decomposition on $\mathbf{T}$ to obtain $\mathbf{T} = \mathbf{QDQ}^{'}$, where $\mathbf{Q}$ is an $I \times I$ orthogonal matrix, **D** = diag(d_1,...,d_I) such that $d_1 \geq ... \geq d_I \geq 0$. Let $\mathbf{D}^{-\frac{1}{2}} = diag\left(d_1^*, ..., d_I^*\right)$ where $d_i^* = d_i^{-\frac{1}{2}}$ if $d_i > 0$, 0 otherwise. Perform singular value decomposition on $\mathbf{D}^{-\frac{1}{2}}\mathbf{Q}^{'}\mathbf{BQD}^{-\frac{1}{2}}$ to obtain its eigenvector **a** which is associated with its largest eigenvalue.

The largest discriminant coordinate of **g** is the projection

$$\xi = \mathbf{a}^{'}\mathbf{D}^{-\frac{1}{2}}\mathbf{Q}^{'}\mathbf{g}$$

Quadratic Discriminant Analysis (QDA)

To determine the cutpoint for a continuous predictor, first group the categories of the target field *Y* to form two superclasses, *A* and *B*, as described above.

If min $\left(s_A^2, s_B^2\right) = 0$, order the two superclasses by their variance in increasing order and denote the variances by $s_1^2 \leq s_2^2$, and the corresponding means by $\overline{x}_1, \overline{x}_2$. Let ε be a very small positive number, say ε = 10^{-12}. Set the cutpoint *d* based on $\overline{x}_1$ and ε:

$$d = \begin{cases} \overline{x}_1(1+\epsilon) & \text{if } \overline{x}_1 < \overline{x}_2 \\ \overline{x}_1(1-\epsilon) & \text{otherwise} \end{cases}$$

Blank Handling

Records with missing values for the target field are ignored in building the tree model.

Surrogate splitting is used to handle blanks for predictor fields. If the best predictor field to be used for a split has a blank or missing value at a particular node, another field that yields a split similar to the predictor field in the context of that node is used as a surrogate for the predictor field, and its value is used to assign the record to one of the child nodes.

Note: If Surrogate splitting is used (where a particular rule does not fit into a node) the Confidence score is reduced by multiplying it by 0.9. This can result in multiple Confidence scores being present within a single node.

For example, suppose that *X** is the predictor field that defines the best split *s** at node *t*. The surrogate-splitting process finds another split *s*, the surrogate, based on another predictor field *X* such that this split is most similar to *s** at node *t* (for records with valid values for both predictors). If a new record is to be predicted and it has a missing value on *X** at node *t*, the surrogate split *s* is applied instead. (Unless, of course, this record also has a missing value on *X*. In such a situation, the next best surrogate is used, and so on, up to the limit of number of surrogates specified.)

In the interest of speed and memory conservation, only a limited number of surrogates is identified for each split in the tree. If a record has missing values for the split field and all surrogate fields, it is assigned to the child node with the higher weighted probability, calculated as

$$\frac{N_{f,j}(t)}{N_f(t)}$$

where $N_{\text{f,j}}(t)$ is the sum of frequency weights for records in category *j* for node *t*, and $N_{\text{f}}(t)$ is the sum of frequency weights for all records in node *t*.

If the model was built using equal or user-specified priors, the priors are incorporated into the calculation:

$$\frac{\pi(j)}{p_f(t)} \times \frac{N_{f,j}(t)}{N_f(t)}$$

where $\pi(j)$ is the prior probability for category *j*, and $p_{\text{f}}(t)$ is the weighted probability of a record being assigned to the node,

$$p_f(t) = \sum_j \frac{\pi(j)N_{f,j}(t)}{N_{f,j}}$$

where $N_{\text{f,j}}(t)$ is the sum of the frequency weights (or the number of records if no frequency weights are defined) in node *t* belonging to category *j*, and $N_{\text{f,j}}$ is the sum of frequency weights for records belonging to category in the entire training sample.

Predictive measure of association

Let $\hbar_{X^* \cap X}$ (resp. $\hbar_{X^* \cap X}(t)$) be the set of learning cases (resp. learning cases in node *t*) that has non-missing values of both X^* and *X*. Let $p\left(s^* \approx s_X | t\right)$ be the probability of sending a case in $\hbar_{X^* \cap X}(t)$ to the same child by both s^* and s_X, and $\tilde{s}_X$ be the split with maximized probability $p\left(s^* \approx \tilde{s}_X | t\right) = \max_{s_X}\left(p\left(s^* \approx s_X | t\right)\right)$.

The predictive measure of association $\lambda\left(s^* \approx \tilde{s}_X | t\right)$ between s^* and $\tilde{s}_X$ at node *t* is

$$\lambda\left(s^* \approx \tilde{s}_X | t\right) = \frac{\min\left(p_L, p_R\right) - \left(1 - p\left(s^* \approx \tilde{s}_X | t\right)\right)}{\min\left(p_L, p_R\right)}$$

where p_L (resp. p_R) is the relative probability that the best split s^* at node *t* sends a case with non-missing value of X^* to the left (resp. right) child node. And where

$$p\left(s^* \approx s_X | t\right) = \begin{cases} \displaystyle\sum_j \frac{\pi\left(j\right) N_{w,j}\left(s^* \approx s_X, t\right)}{N_{w,j}(X^* \cap X)} & \text{if}Y\text{is categorical} \\ \frac{N_w(s^* \approx s_X, t)}{N_w(X^* \cap X)} & \text{if}Y\text{is continuous} \end{cases}$$

with

$$N_w\left(X^* \cap X\right) = \sum_{n \in \hbar_{X^* \cap X}} w_n f_n,\ N_w\left(X^* \cap X, t\right) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n$$

$$N_w\left(s^* \approx s_X, t\right) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n I\left(n : s^* \approx s_X\right)$$

$$N_{w,j}\left(X^* \cap X\right) = \sum_{n \in \hbar_{X^* \cap X}} w_n f_n I\left(y_n = j\right), N_{w,j}\left(X^* \cap X\right) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n I\left(y_n = j\right)$$

$$N_{w,j}\left(s^* \approx s_X, t\right) = \sum_{n \in \hbar_{X^* \cap X}(t)} w_n f_n I\left(y_n = j\right) I\left(n : s^* \approx s_X\right)$$

and $I\left(n : s^* \approx s_X\right)$ being the indicator function taking value 1 when both splits s^* and s_X send the case *n* to the same child, 0 otherwise.

Effect of Options

Stopping Rules

Stopping rules control how the algorithm decides when to stop splitting nodes in the tree. Tree growth proceeds until every leaf node in the tree triggers at least one stopping rule. Any of the following conditions will prevent a node from being split:

- The node is pure (all records have the same value for the target field)
- All records in the node have the same value for all predictor fields used by the model
- The tree depth for the current node (the number of recursive node splits defining the current node) is the *maximum tree depth* (default or user-specified).
- The number of records in the node is less than the *minumum parent node size* (default or user-specified)
- The number of records in any of the child nodes resulting from the node's best split is less than the *minimum child node size* (default or user-specified)

Profits

Profits are numeric values associated with categories of a (symbolic) target field that can be used to estimate the gain or loss associated with a segment. They define the relative value of each value of the target field. Values are used in computing gains but not in tree growing.

Profit for each node in the tree is calculated as

$$\sum_j f_j(t) P_j$$

where *j* is the target field category, $f_j(t)$ is the sum of frequency field values for all records in node *t* with category *j* for the target field, and P_j is the user-defined profit value for category *j*.

Priors

Prior probabilities are numeric values that influence the misclassification rates for categories of the target field. They specify the proportion of records expected to belong to each category of the target field prior to the analysis. The values are involved both in tree growing and risk estimation.

There are three ways to derive prior probabilities.

Empirical Priors

By default, priors are calculated based on the training data. The prior probability assigned to each target category is the weighted proportion of records in the training data belonging to that category,

$$\pi(j) = \frac{N_{w,j}}{N_w}$$

In tree-growing and class assignment, the *N*s take both case weights and frequency weights into account (if defined); in risk estimation, only frequency weights are included in calculating empirical priors.

Equal Priors

Selecting equal priors sets the prior probability for each of the *J* categories to the same value,

$$\pi(j) = \frac{1}{J}$$

User-Specified Priors

When user-specified priors are given, the specified values are used in the calculations involving priors. The values specified for the priors must conform to the probability constraint: the sum of priors for all categories must equal 1.0. If user-specified priors do not conform to this constraint, adjusted priors are derived which preserve the proportions of the original priors but conform to the constraint, using the formula

$$\pi'(j) = \frac{\pi(j)}{\sum_J \pi(j)}$$

where *π'*(*j*) is the adjusted prior for category *j*, and *π*(*j*) is the original user-specified prior for category *j*.

Costs

If misclassification costs are specified, they are incorporated into split calculations by using altered priors. The altered prior is defined as

$$\pi^{'}(t) = \frac{C(j)\pi(j)}{\sum_J C(j)\pi(j)}$$

where $C(j) = \sum_i C(i|j)$.

Misclassification costs also affect risk estimates and predicted values, as described below (on p. 324 and on p. 325, respectively).

Pruning

Pruning refers to the process of examining a fully grown tree and removing bottom-level splits that do not contribute significantly to the accuracy of the tree. In pruning the tree, the software tries to create the smallest tree whose misclassification risk is not too much greater than that of the largest tree possible. It removes a tree branch if the cost associated with having a more complex tree exceeds the gain associated with having another level of nodes (branch).

It uses an index that measures both the misclassification risk and the complexity of the tree, since we want to minimize both of these things. This cost-complexity measure is defined as follows:

$$R_\alpha\left(T\right) = R\left(T\right) + \alpha\left|\tilde{T}\right|$$

R(*T*) is the misclassification risk of tree *T*, and $\left|\tilde{T}\right|$ is the number of terminal nodes for tree *T*. The term *α* represents the complexity cost *per terminal node* for the tree. (Note that the value of *α* is calculated by the algorithm during pruning.)

Any tree you might generate has a maximum size (T_{max}), in which each terminal node contains only one record. With no complexity cost (*α* = 0), the maximum tree has the lowest risk, since every record is perfectly predicted. Thus, the larger the value of *α*, the fewer the number of terminal nodes in *T*(*α*), where *T*(*α*) is the tree with the lowest complexity cost for the given *α*. As *α* increases from 0, it produces a finite sequence of subtrees (T_1, T_2, T_3), each with progressively fewer terminal nodes. Cost-complexity pruning works by removing the weakest split.

The following equations represent the cost complexity for $\{t\}$, which is any single node, and for T_t, the subbranch of $\{t\}$.

$$R_\alpha\left(\{t\}\right) = R\left(t\right) + \alpha$$

$$R_\alpha\left(T_t\right) = R\left(T_t\right) + \alpha\left|\tilde{T}_t\right|$$

If $R_\alpha(T_t)$ is less than $R_\alpha(\{t\})$, then the branch T_t has a smaller cost complexity than the single node $\{t\}$.

The tree-growing process ensures that $R_\alpha(\{t\}) \geq R_\alpha(T_t)$ for (*α* = 0). As *α* increases from 0, both $R_\alpha(\{t\})$ and $R_\alpha(T_t)$ grow linearly, with the latter growing at a faster rate. Eventually, you will reach a threshold *α'*, such that $R_\alpha(\{t\}) < R_\alpha(T_t)$ for all *α* > *α*'. This means that when *α* grows larger than *α'*, the cost complexity of the tree can be reduced if we cut the subbranch T_t under $\{t\}$. Determining the threshold is a simple computation. You can solve this first inequality, $R_\alpha(\{t\}) \geq R_\alpha(T_t)$, to find the largest value of *α* for which the inequality holds, which is also represented by *g*(*t*). You end up with

$$\alpha \leq g(t) = \frac{R(t) - R(T_t)}{\left|\tilde{T}_t\right| - 1}$$

You can define the weakest link ($\bar{t}$) in tree *T* as the node that has the smallest value of *g*(*t*):

$$g(\bar{t}) = \min_{t \in T} \; g(t)$$

Therefore, as α increases, $\bar{t}$ is the first node for which $R_\alpha(\{t\}) = R_\alpha(T_t)$. At that point, $\{\bar{t}\}$ becomes preferable to $T_{\bar{t}}$, and the subbranch is pruned.

With that background established, the pruning algorithm follows these steps:

- Set α_1 = 0 and start with the tree T_1 = *T*(0), the fully grown tree.
- Increase α until a branch is pruned. Prune the branch from the tree, and calculate the risk estimate of the pruned tree.
- Repeat the previous step until only the root node is left, yielding a series of trees, T_1, T_2, ... T_k.
- If the standard error rule option is selected, choose the smallest tree T_opt for which

$$R(T_{opt}) \leq \min_k \; R(T_k) + m \times SE(R(T))$$

- If the standard error rule option is not selected, then the tree with the smallest risk estimate *R*(*T*) is selected.

Secondary Calculations

Secondary calculations are not directly related to building the model but give you information about the model and its performance.

Risk Estimates

Risk estimates describe the risk of error in predicted values for specific nodes of the tree and for the tree as a whole.

Risk Estimates for Symbolic Target Field

For classification trees (with a symbolic target field), the risk estimate *r*(*t*) of a node *t* is computed as

$$r\,(t) = \frac{1}{N_f} \sum_j N_{f,j}(t) C(j^*(t)|j)$$

where $C(j^*(t)|j)$ is the misclassification cost of classifying a record with target value j as $j^*(t)$, $N_{\mathrm{f,j}}(t)$ is the sum of the frequency weights for records in node t in category j (or the number of records if no frequency weights are defined), and N_{f} is the sum of frequency weights for all records in the training data.

If the model uses user-specified priors, the risk estimate is calculated as

$$\sum_j \frac{\pi(j)N_{f,j}(t)}{N_{f,j}}C(j^*(t)|j)$$

Gain Summary

The **gain summary** provides descriptive statistics for the terminal nodes of a tree.

If your target field is continuous (scale), the gain summary shows the weighted mean of the target value for each terminal node,

$$g(t) = \sum_{i \in t} w_i f_i x_i$$

If your target field is symbolic (categorical), the gain summary shows the weighted percentage of records in a selected target category,

$$g\left(t, j\right) = \frac{\sum_{i \in t} f_i x_i(j)}{\sum_{i \in t} f_i}$$

where $x_{\mathrm{i}}(j) = 1$ if record x_{i} is in target category j, and 0 otherwise. If profits are defined for the tree, the gain is the average profit value for each terminal node,

$$g(t) = \sum_{i \in t} f_i P(x_i)$$

where $P(x_{\mathrm{i}})$ is the profit value assigned to the target value observed in record x_{i}.

Generated Model/Scoring

Calculations done by the QUEST generated model are described below.

Predicted Values

New records are scored by following the tree splits to a terminal node of the tree. Each terminal node has a particular predicted value associated with it, determined as follows:

For trees with a symbolic target field, each terminal node's predicted category is the category with the lowest weighted cost for the node. This weighted cost is calculated as

$$\min_{i} \sum_{j} C(i|j)p(j|t)$$

where $C(i|j)$ is the user-specified misclassification cost for classifying a record as category i when it is actually category j, and $p(j|t)$ is the conditional weighted probability of a record being in category j given that it is in node t, defined as

$$p(j|t) = \frac{p(j,t)}{\sum_j p(j,t)}, p(j,t) = \pi(j)\frac{N_{w,j}(t)}{N_{w,j}}$$

where $\pi(j)$ is the prior probability for category j, $N_{\mathrm{w,j}}(t)$ is the weighted number of records in node t with category j (or the number of records if no frequency or case weights are defined),

$$N_{w,j}(t) = \sum_{i \in t} w_i f_i j(i)$$

and $N_{\mathrm{w,j}}$ is the weighted number records in category j (any node),

$$N_{w,j} = \sum_{i \in T} w_i f_i j(i)$$

Confidence

Confidence for a scored record is the proportion of weighted records in the training data in the scored record's assigned terminal node that belong to the predicted category, modified by the Laplace correction:

$$\frac{N_{f,j}(t) + 1}{N_f(t) + k}$$

Note: If Surrogate Splitting is used (where a particular rule does not fit into a node) the Confidence score is reduced by multiplying it by 0.9. This can result in multiple Confidence scores being present within a single node.

Blank Handling

In classification of new records, blanks are handled as they are during tree growth, using surrogates where possible, and splitting based on weighted probabilities where necessary. For more information, see the topic "Blank Handling" on p. 319.

Self-Learning Response Model Algorithms

Self-Learning Response Models (SLRMs) use Naive Bayes classifiers to build models that can be easily updated to incorporate new data, without having to regenerate the entire model. The methods used for building, updating and scoring with SLRMs are described here.

Primary Calculations

The model-building algorithm used in SLRMs is Naive Bayes. A Bayesian Network consisting of a Naive Bayes model for each target field is generated.

Naive Bayes Algorithms

The Naive Bayes model is an old method for classification and predictor selection that is enjoying a renaissance because of its simplicity and stability.

Notation

The following notation is used throughout this chapter unless otherwise stated:

Table 31-1
Notation

Notation	Description
J_0	Total number of predictors.
X	Categorical predictor vector **X’** = (X_1, ..., X_J), where J is the number of predictors considered.
M_j	Number of categories for predictor X_j.
Y	Categorical target variable.
K	Number of categories of Y.
N	Total number of cases or patterns in the training data.
N_k	The number of cases with Y= k in the training data.
$N^j{}_{mk}$	The number of cases with Y= k and X_j=m in the training data.
π_k	The probability for Y= k.
$p^j{}_{mk}$	The probability of X_j=m given Y= k.

Naive Bayes Model

The Naive Bayes model is based on the conditional independence model of each predictor given the target class. The Bayesian principle is to assign a case to the class that has the largest posterior probability. By Bayes’ theorem, the posterior probability of Y given **X** is:

$$P\left(Y=k|\mathbf{X}=\mathbf{x}\right)=\frac{P(\mathbf{X}=\mathbf{x}|Y=k)P(Y=k)}{\sum_{i=1}^{K}P\left(\mathbf{X}=\mathbf{x}|Y=i\right)P\left(Y=i\right)}$$

Let X_1, ..., X_J be the J predictors considered in the model. The Naive Bayes model assumes that X_1, ..., X_J are conditionally independent given the target; that is:

$$P\Big(\mathbf{X} = \mathbf{x}\,|Y = k) = \prod_{j=1}^{J} P\left(X_j = x_j|\, Y = k\Big)\right.$$

These probabilities are estimated from training data by the following equations:

$$\pi_k = P\left(Y = k\right) = \frac{N_k+\lambda}{N+K\lambda}$$

$$p^j_{mk} = P\left(X_j = m|Y = k\right) = \frac{N^j_{mk}+f}{\Sigma_{l=1}^{M_j} N^j_{lk} + M_j f}$$

Where N_k is calculated based on all non-missing Y, N^j_{mk} is based on all non-missing pairs of X_J and Y, and the factors λ and *f* are introduced to overcome problems caused by zero or very small cell counts. These estimates correspond to Bayesian estimation of the multinomial probabilities with Dirichlet priors. Empirical studies suggest $\lambda = f = \frac{1}{N}$(Kohavi, Becker, and Sommerfield, 1997).

A single data pass is needed to collect all the involved counts.

For the special situation in which J = 0; that is, there is no predictor at all, $P\left(Y = k|\mathbf{X} = \mathbf{x}\right) = P\left(Y = k\right)$. When there are empty categories in the target variable or categorical predictors, these empty categories should be removed from the calculations.

Secondary Calculations

In addition to the model parameters, a model assessment is calculated.

Model Assessment

For a trained model, we need to assess how reliable it is. Given this problem, we face two conditions which will result with different solutions:

- A sample of test data (not used in training or updating the model) is available. In this case we can directly feed these data into the model, and observe the outcome.
- No extra testing data are available. This is more common since users normally apply all available data to train the model. In this case, we have to simulate data first based on the calibrated model parameters, such as π_k and p^j_{mk}, then assess the trained model by scoring these pseudo random data.

Testing with Simulated Data

In our simulation, $n_{round} \times n_{sample\_per\_round}$ data are generated. For each round, we can determine the corresponding accuracy; across all rounds, average accuracy and variance can be calculated, and they are explained as reliability statistics.

- For each round, we generate n_{sample} random cases as follows:
  - y is assigned a random value based on the prior probabilities π_k.
  - Each X_i is randomly assigned based on conditional probabilities $\hat{P}\left(X_j|Y=y\right)$
- The accuracy of each round is calculated by comparing the model's predicted value for each case to the case's generated outcome y, $P_{accuracy} = \frac{n_{correct}}{n_{sample}}$
- The mean, variance, minimum and maximum of the accuracy estimates are calculated across rounds.

Blank Handling

If the target is missing, or all J_0 predictors for a case are missing, the case is ignored. If every value for a predictor is missing, or all non-missing values for a predictor are the same, that predictor is ignored.

Updating the Model

The model can be updated by updating the cell counts N_k, N^j_{mk} to account for the new records and recalculating the probabilities π_k and p^j_{mk} as described in "Naive Bayes Model" on p. 327. Updating the model only requires a data pass of new records.

Generated Model/Scoring

Scoring with a generated SLRM model is described below.

Predicted Values and Confidences

By default, the first *M* offers with highest predicted value will be returned. However, sometimes low-probability offers are of interest for marketing strategy. Model settings allow you to bias the results toward particular offers, or include random components to the offers.

Some notation for scoring offers:

N	Number of offers modeled already
$P = \{P_1, P_2, ..., P_N\}$	Scores for each offer
$P_r = \{P_{r1}, P_{r2}, ..., P_{rN}\}$	Randomly generated scores for offers
α	Randomization factor, ranging from 0.0 (offer based only on model prediction) to 1.0 (offer is completely random)
$W = \{W_1, W_2, ..., W_N\}$	Number of cases used for training each offer
W_{emp}	Empirical value of the amount of training cases that will result in a reliable model. When "Take account of model reliability" is selected in the Settings tab, this is set to 500; otherwise 0.

$S = \{S_1, S_2, ..., S_N\}$	User's preferences for offers, or the ratings of the offers. Can be any non-negative value, where larger values means stronger recommendations for the corresponding offers. The default setting is $S = \{1, 1, ..., 1\}$
$F = \{F_1, F_2, ..., F_N\}$	Mandatory inclusion/exclusion filters. $F_i \in \{0, 1\}$, where 0 indicates an excluded offer.

The final score for each offer is calculated as

$$P_i = \left[\alpha P_{ri} + (1-\alpha)\left(\frac{W_i}{W_i + W_{emp}}P + \frac{W_{emp}}{W_i + W_{emp}}0.5\right)\right] \cdot \frac{S_i}{\max(S)} \cdot F_i$$

The outcomes P_i are ordered in specified order, ascending or descending, and the first *M* offers in the list are recommended. The calculated score is reported as the confidence for the score.

Variable Assessment

Among all the features modeled, some are definitely more important to the accuracy of the model than others. Two different approaches to measuring importance are proposed here: Predictor Importance and Information Measure.

Predictor Importance

The variance of predictive error can be used as the measure of importance. With this method, we leave out one predictor variable at a time, and observe the performance of remaining model. A variable is regarded as more important than another if it adds more variance compared to that of the complete model (with all variables).

When test data are available, they can be used for predictor importance calculations in a direct way. When test data are not available, they are simulated based on the model parameters π_k and p^j_{mk}.

In our simulation, $n_{round} \times n_{sample\_per\_round}$ data are generated. For each round, we determine the corresponding accuracy for each submodel, excluding X_j for each of the *j* predictors; across all rounds, average accuracy and variance can be calculated.

- For each round, we generate n_{sample} random cases as follows:
  - *y* is assigned a random value based on the prior probabilities π_k.
  - Each X_i is randomly assigned based on conditional probabilities $\hat{P}\left(X_j | Y = y\right)$

Within a round, each of the X_j predictors is excluded from the model, and the accuracy is calculated based on the generated test data for each submodel in turn.

- The accuracies for each round are calculated by comparing the submodel's predicted value for each case to the case's generated outcome *y*, $P_{accuracy\_without\_xj} = \frac{n_{correct\_without\_xj}}{n_{sample}}$, for each of the *j* submodels.

- The mean and variance of the accuracy estimates are calculated across rounds for each submodel. For each variable, the importance is measured as the difference between the accuracy of the full model and the mean accuracy for the submodels that excluded the variable.

Information Measure

The importance of an explanatory variable *X* for a response variable *Y* is the extent to which the use of *X* reduces uncertainty in predicting outcomes of *Y*. The uncertainty about predicting an outcome *Y* is measured by the entropy of its distribution (Shannon 1948):

$$H_Y \equiv -\sum_i P\left(Y=i\right)\log P\left(Y=i\right)$$

Based on a value *x* of the explanatory variable, the probability distribution of the outcomes *Y* is the conditional distribution $f_{y|x}$. The information value of using the value *x* for the prediction is assessed by comparing the concentrations of the marginal distribution f_y and the conditional distribution $f_{y|x}$. The difference between the conditional and marginal distribution entropy is:

$$\Delta H\left(x_j\right) = H_Y - H_{Y|x_j}$$

where $H_{Y|x_j}$ denotes the entropy of the conditional distribution $f_{y|x_j}$. The value x_j is informative about *Y* if the conditional distribution $f_{y|x_j}$ is more concentrated than f_y.

The importance of a random variable *X* for predicting *Y* is measured by the expected uncertainty reduction, referred to as the *mutual information* between two variables:

$$\begin{array}{rcl} M\left(Y,X\right) & \equiv & \sum_i f_x\left(x_j\right)\Delta H\left(x_j\right) \\ & = & H_Y - H_{Y|X} \\ & = & H_Y + H_X - H_{Y,X} \end{array}$$

The expected fraction of uncertainty reduction due to *X* is a mutual information index given by

$$I_{y,x} = 1 - \frac{H_{Y|X}}{H_Y} = \frac{M_{Y,X}}{H_Y}$$

This index ranges from zero to one: $I_{y,x} = 0$ if and only if the two variables are independent, and $I_{y,x} = 100\%$ if and only if the two variables are functionally related in some form, linearly or nonlinearly.

Sequence Algorithm

Overview of Sequence Algorithm

The sequence node in IBM® SPSS® Modeler detects patterns in sequential data, such as purchases over time. The sequence node algorithm uses the following two-stage process for sequential pattern mining (Agrawal and Srikant, 1995):

- **Mine for the frequent sequences.** This part of the process extracts the information needed for quick responses to the pattern queries, yielding an adjacency lattice of the frequent sequences. This structure provides an optimal configuration for the second stage.
- **Generate sequential patterns online.** This stage uses a pre-computed adjacency lattice. You can extract the patterns according to specified criteria, such as support and confidence bounds, or place restrictions on the antecedent sequence.

Primary Calculations

Itemsets, Transactions, and Sequences

A group of items associated at a single point in time constitutes an **itemset**, which will be identified here using braces "{ }". Consider the hypothetical data below representing sales at a gourmet store.

Table 32-1
Example data - product purchases

Customer	Time 1	Time 2	Time 3	Time 4
1	cheese & crackers	wine	beer	-
2	wine	beer	cheese	-
3	bread	wine	cheese & beer	-
4	crackers	wine	beer	cheese
5	beer	cheese & crackers	bread	-
6	crackers	bread	-	-

Customer 1 yields three itemsets: {*cheese* & *crackers*}, {*wine*}, and {*beer*}. The ampersand denotes items appearing in a single itemset. In this case, items separated by an ampersand appear in the same purchase. Notice that some itemsets may contain a single item only.

The complete group of itemsets for a single object, in this case a customer, constitutes a **transaction**. *Time* refers to a purchase occasion for a particular customer and does not represent a specific time across all customers. For example, the first purchase occasion for customer 1 may have been on January 23 while the first occasion for customer 4 was February 12. Although the dates are not identical, each itemset was the first for that customer. The analysis focuses on time relative to a specific customer instead of on absolute time.

Ordering the itemsets by time yields **sequences**. The symbol ">" denotes an ordering of itemsets, with the itemset on the right occurring after the itemset on the left. For example, customer 6 yields a sequence of [{*crackers*} > {*bread*}].

Two common characteristics used to describe sequences are **size** and **length**. The number of items contained in a sequence corresponds to the sequence size. The number of itemsets in the sequence equals its length. For example, the three timepoints for customer 5 correspond to a sequence having a length of three and a size of four.

A sequence is a **subsequence** of another sequence if the first can be derived by deleting itemsets from the second. Consider the sequence:

[{*wine*} > {*beer*} > {*cheese*}]

Deleting the itemset *cheese* results in the sequence of length two [{*wine*} > {*beer*}]. This two itemset sequence is a subsequence of the original sequence. Similar deletions reveal that the three itemset sequence can be decomposed into three singleton subsequences ({*wine*}, {*beer*}, {*cheese*}) and three subsequences involving two itemsets ([{*wine*} > {*beer*}], [{*beer*} > {*cheese*}], [{*wine*} > {*cheese*}]). A sequence that is not a subsequence of another sequence is referred to as a **maximal sequence**.

Support

The **support** for a sequence equals the proportion of transactions that contain the sequence. The table below shows support values for sequences that appear in at least one transaction for a set of gourmet store sales data (note that this is a different data set from the one shown previously).

For example, the support for sequence [{*wine*} > {*beer*}] is 0.67 because it occurs in four of the six transactions. Similarly, support for a sequential rule equals the proportion of transactions that contain both the antecedent and the consequent of the rule, in that order. The support for the sequential rule:

If [{cheese} >
{wine}] then [{beer}]

is 0.17 because only one of the six transactions contains these three itemsets in this order.

Sequences that do not appear in any transaction have support values of 0 and are excluded from the mining analysis.

Table 32-2
Nonzero support values

Sequence	**Support**	**Sequence**	**Support**
{cheese}	0.83	{crackers} > {cheese}	0.17
{crackers}	0.67	{beer} > {cheese & crackers}	0.17
{wine}	0.67	{cheese & crackers} > {wine}	0.17
{beer}	0.83	{cheese & crackers} > {beer}	0.17
{bread}	0.50	{bread} > {cheese & beer}	0.17
{cheese & crackers}	0.33	{wine} > {cheese & beer}	0.17
{cheese & beer}	0.17	{cheese & crackers} > {bread}	0.17
{cheese} > {wine}	0.17	{cheese} > {wine} > {beer}	0.17
{cheese} > {beer}	0.17	{crackers} > {wine} > {beer}	0.33
{wine} > {beer}	0.67	{wine} > {beer} > {cheese}	0.33
{crackers} > {wine}	0.33	{bread} > {wine} > {beer}	0.17

Sequence	Support	Sequence	Support
{crackers} > {beer}	0.33	{bread} > {wine} > {cheese}	0.17
{wine} > {cheese}	0.50	{beer} > {cheese} > {bread}	0.17
{beer} > {cheese}	0.50	{beer} > {crackers} > {bread}	0.17
{bread} > {wine}	0.17	{crackers} > {wine} > {cheese}	0.17
{bread} > {beer}	0.17	{crackers} > {beer} > {cheese}	0.17
{bread} > {cheese}	0.17	{cheese & crackers} > {wine} > {beer}	0.17
{beer} > {bread}	0.17	{bread} > {wine} > {cheese & beer}	0.17
{beer} > {crackers}	0.17	{beer} > {cheese & crackers} > {bread}	0.17
{cheese} > {bread}	0.17	{crackers} > {wine} > {beer} > {cheese}	0.17
{crackers} > {bread}	0.33		

Typically, the analysis focuses on sequences having support values greater than a minimum threshold, the **support level**. This value, defined by the user, determines the minimum level for which sequences will be kept. Sequences with support values exceeding the threshold, referred to as **frequent sequences**, form the basis of the adjacency lattice. For example, for a threshold of 0.40, sequence [{*wine*} > {*beer*}] is a frequent sequence because its support level is 0.67. By relaxing the threshold, more sequences are classified as frequent.

Time Constraints

Defining the time at which events occur has a dramatic impact on sequences. For instance, each purchase occasion in the gourmet data yields a new timed itemset. However, suppose a customer bought wine and realized while walking to his car that beer was needed too. He immediately returns to the store and buys the forgotten item. Should these two purchases be considered separately?

One method for controlling for itemsets that occur very close in time is through a **timestamp tolerance** parameter. This tolerance defines the length of time covering a single itemset. Specifying a tolerance larger than the difference between two consecutive times results in a single itemset at one time, such as {*wine* & *beer*} in the scenario described above.

Another time issue commonly arising in the analysis of sequences is **gap**. This statistic measures the difference in time between two items and can be used to make time-based predictions of future behavior. Gap statistics can be based on the gap between the last and penultimate sets in sequences, or on the gaps between the last and first sets in sequences.

Sequential Patterns

Sequential patterns, or sequential association rules,identify items that frequently follow other items in transaction-based data. A sequential pattern is simply an ordered list of itemsets. All itemsets leading to the final itemset form the **antecedent** sequence, and the last itemset is the **consequent** sequence. These statements have the following form:

```
If [antecedent] then [consequent]
```

For example, a sequential pattern for *wine*, *beer*, and *cheese* is: "if a customer buys wine, then buys beer, he will buy cheese in the future". *Wine* and *beer* form the antecedent, and *cheese* is the consequent.

Notationally, the symbol "=>" separates the antecedent from the consequent in a sequential rule. The sequence to the left of this symbol corresponds to the antecedent; the sequence on the right is the consequent. For instance, the rule above is denoted:

[{*wine*} > {*beer* } => {*cheese*}]

The only notational difference between a sequence and a sequential rule is the identification of a subsequence as a consequent.

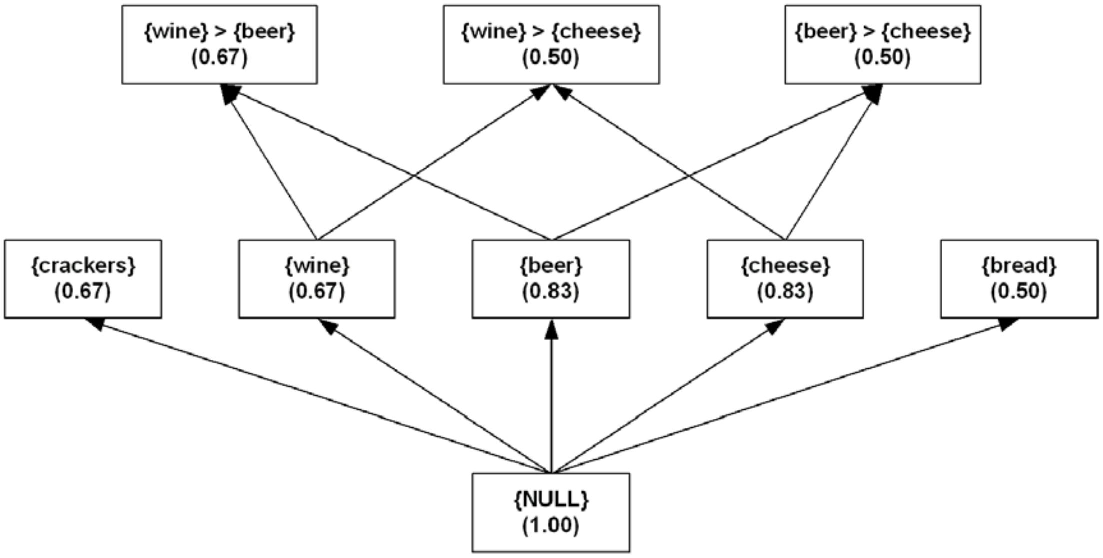
Adjacency Lattice

The number of itemsets and sequences for a collection of transactions grows very quickly as the number of items appearing in transactions gets larger. In practice, analyses typically involve many transactions and these transactions include a variety of itemsets. Larger datasets require complex methods to process the sequential patterns, particularly if rapid feedback is needed.

An adjacency lattice provides a structure for organizing sequences, permitting rapid generation of sequential patterns. Two sequences are **adjacent** if adding a single item to one yields the other, resulting in a hierarchical structure denoting which sequences are subsequences of other sequences. The lattice also includes sequence frequencies, as well as other information.

The adjacency lattice of all observed sequences is usually too large to be practical. It may be more useful to prune the lattice to frequent sequences in an effort to simplify the structure. All sequences contained in the resulting structure reach a specified support level. The adjacency lattice for the sample transactions using a support level of 0.40 is shown below.

Figure 32-1
Adjacency lattice for a threshold of 0.40 (support values in parentheses)



Mining for Frequent Sequences

IBM® SPSS® Modeler uses a non-sequential association rule mining approach that performs very well with respect to minimizing I/O costs, time, and space requirements. The **continuous association rule mining algorithm** (Carma), uses only two data passes and allows changes in the support level during execution (Hidber, 1999). The final guaranteed support level depends on the provided series of support values.

For the first stage of the mining process, the component uses a variation of Carma to apply the approach to the sequential case. The general order of operations is:

- Read the transaction data.
- Identify frequent sequences, discarding infrequent sequences.
- Build an adjacency lattice of frequent sequences.

Carma is based upon transactions and requires only two passes through the data. In the first data pass, referred to as Phase I, the algorithm generates the frequent sequence candidates. The second data pass, Phase II, computes the exact frequency counts for the candidate sequences from Phase I.

Phase I

Phase I corresponds to an estimation phase. In this phase, Carma generates candidate sequences successively for every transaction. Candidate sequences satisfy a version of the "apriori" principle where a sequence becomes a candidate only if all of its subsequences are candidates from the previous transactions. Therefore, the size of candidate sequences can grow with each transaction. To prevent the number of candidates from growing too large, Carma periodically prunes candidate sequences that have not reached a threshold frequency. Pruning may occur after processing any number of transactions. While pruning usually lowers the memory requirements, it increases the computational costs. At the end of the Phase I, the algorithm generates all sequences whose frequency exceeds the computed support level (which depends on the support series). Carma can use many support levels, up to one support level per transaction.

The table below represents support values during transaction processing with no pruning for the gourmet data. As the algorithm processes a transaction, support values adjust to account for items appearing in that transaction, as well as for the total number of processed transactions. For example, after the first transaction, the lattice contains *cheese*, *crackers*, *wine*, and *beer*, each having a support exceeding the threshold level. After processing the second transaction, the support for *crackers* drops from 1.0 to 0.50 because that item appears in only one of the two transactions. The support for the other items remains unchanged because both transactions contain the items. Furthermore, the sequences [{*wine*} > {*beer*}] and [{*beer*} > {*cheese*}] enter the lattice because their constituent subsequences already appear in the lattice.

Table 32-3
Carma transaction processing

	Transaction					
Sequence	**1**	**2**	**3**	**4**	**5**	**6**
{cheese}	1	1	1	1	1	0.83
{crackers}	1	0.50	0.33	0.50	0.60	0.67
{wine}	1	1	1	1	0.80	0.67
{beer}	1	1	1	1	1	0.83

	Transaction					
Sequence	**1**	**2**	**3**	**4**	**5**	**6**
{wine} > {beer}		1	1	1	0.80	0.67
{beer} > {cheese}		0.50	0.33	0.50	0.60	0.50
{bread}			0.33	0.25	0.40	0.50
{wine} > {cheese}			0.67	0.75	0.60	0.50
{cheese & beer}			0.33	0.25	0.20	0.17
{crackers} > {wine}				0.50	0.40	0.33
{crackers} > {beer}				0.50	0.40	0.33
{crackers} > {cheese}				0.25	0.20	0.17
{wine} > {beer} > {cheese}				0.50	0.40	0.33
{cheese & crackers}					0.40	0.33
{beer} > {crackers}					0.20	0.17
{beer} > {bread}					0.20	0.17
{cheese} > {bread}					0.20	0.17
{crackers} > {bread}					0.20	0.33

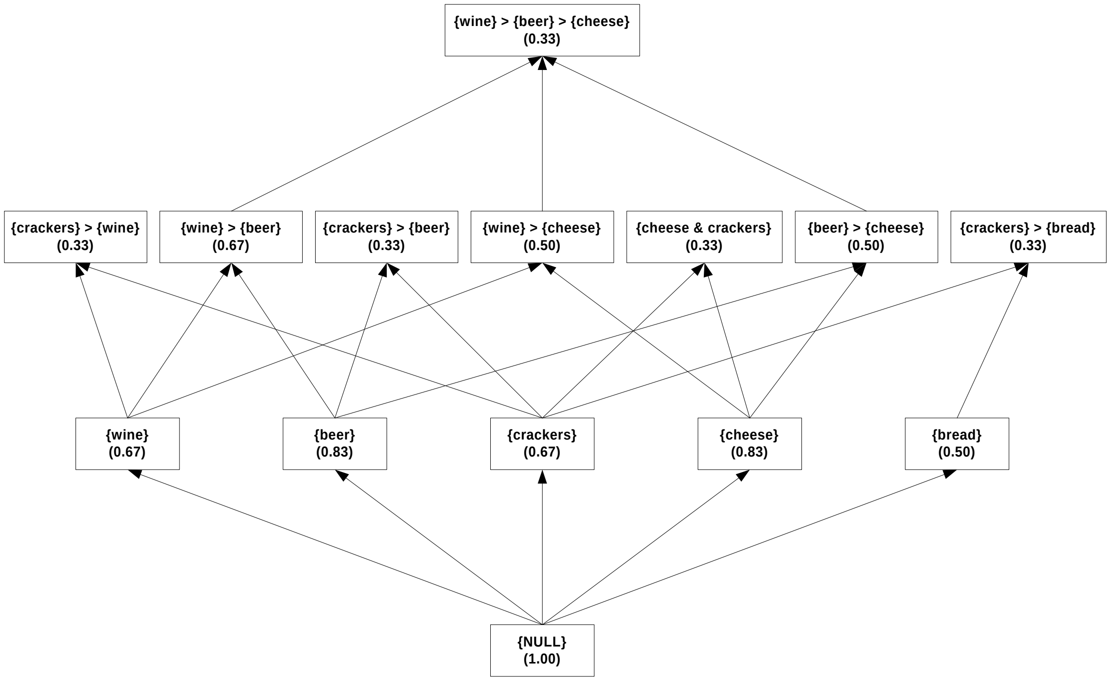
After completing the first data pass, the lattice contains five sequences containing one item, twelve sequences involving two items, and one sequence composed of three items.

Phase II

Phase II is a validation phase requiring a second data pass, during which the algorithm determines accurate frequencies for candidate sequences. In this phase, Carma does not generate any candidate sequences and prunes infrequent sequences only once, making Phase II faster than Phase I. Moreover, depending on the entry points of candidate sequences during Phase I, a complete data pass my not even be necessary. In an online application, Carma skips Phase II altogether.

Suppose the threshold level is 0.30 for the lattice. Several sequences fail to reach this level and subsequently get pruned during Phase II. The resulting lattice appears below.

Figure 32-2
Adjacency lattice for a threshold of 0.30 (support values in parentheses)



Notice that the lattice does not contain [{*crackers*} > {*wine*} > {*beer*}] although the support for this sequence exceeds the threshold. Although [{*crackers*} > {*wine*} > {*beer*}] occurs in one-third of the transactions, Carma cannot add this sequence to the lattice until all of its subsequences are included. The final two subsequences occur in the fourth transaction, after which the full three-itemset sequence is not observed. In general, however, the database of transactions will be much larger than the small example shown here, and exclusions of this type will be extremely rare.

Generating Sequential Patterns

The second stage in the sequential pattern mining process queries the adjacency lattice of the frequent sequences produced in the first stage for the actual patterns. Aggarwal and Yu (1998a) IBM® SPSS® Modeler uses a set of efficient algorithms for generating association rules online from the adjacency lattice (Aggarwal and Yu, 1998). Applying these algorithms to the sequential case takes advantage of the monotonic properties for rule support and confidence preserved by the adjacency lattice data structures. The lattice efficiently saves all the information necessary for generating the sequential patterns and is orders of magnitude smaller than all the patterns it could possibly generate.

The queries contain the constraints that the resulting set of sequential patterns needs to satisfy. These constraints fall into two categories:

- constraints on statistical indices
- constraints on the items contained in the antecedent of the patterns

Statistical index constraints involve support, confidence, or cause. These queries require returned patterns to have values for these statistics within a specified range. Usually, lower confidence bound is the primary criterion. The lower bound for the pattern support level is given by the support level for the sequences in the corresponding adjacency lattice. Often, however, the support specified for pattern generation exceeds the value specified for lattice creation.

For the lattice shown above, specifying a support range between 0.30 and 1.00, a confidence range from 0.30 to 1.0, and a cause range from 0 to 1.0 results in the following seven rules:

- `If [{`*crackers*`}] then [{`*beer*`}].`
- `If [{`*crackers*`}] then [{`*wine*`}].`
- `If [{`*crackers*`}] then [{`*bread*`}].`
- `If [{`*wine*`} > {`*beer*`}] then [{`*cheese*`}].`
- `If [{`*wine*`}] then [{`*beer*`}].`
- `If [{`*wine*`}] then [{`*cheese*`}].`
- `If [{`*beer*`}] then [{`*cheese*`}].`

Limiting the set to only maximal sequences omits the final three rules because they are subsequences of the fourth.

The second type of query requires the specification of the sequential rule antecedent. This type of query returns a new singleton itemset after the final itemset in the antecedent. For example, consider an online shopper who has placed items in a shopping cart. A future item query looks at only the past purchases to derive a recommended item for the next time the shopper visits the site.

Blank Handling

Blanks are ignored by the sequence rules algorithm. The algorithm will handle records containing blanks for input fields, but such a record will not be considered to match any rule containing one or more of the fields for which it has blank values.

Secondary Calculations

Confidence

Confidence is a measure of sequential rule accuracy and equals the proportion obtained by dividing the number of transactions that contain both the antecedent and consequent of the rule by the number of transactions containing the antecedent. In other words, confidence is the support for the rule divided by the support for the antecedent. For example, the confidence for the sequential rule:

If [{wine}] then
[{cheese}]

is 3/4, or 0.75. Three-quarters of the transactions that include *wine* also include *cheese* at a later time. In contrast, the sequential rule:

If [{cheese}] then
[{wine}]

includes the same itemsets but has a confidence of 0.20. Only one-fifth of the transactions that include *cheese* contain *wine* at a later time. In other words, *wine* is more likely to lead to *cheese* than *cheese* is to lead to *wine*.

displays the confidence for every sequential rule observed in the gourmet data. Rules with empty antecedents correspond to having no previous transaction history.

Table 32-4
Nonzero confidence values

Sequence	Confidence	Sequence	Confidence
{cheese}	1.00	{crackers} => {cheese}	0.25
{crackers}	1.00	{beer} => {cheese & crackers}	0.20
{wine}	1.00	{cheese & crackers} => {wine}	0.50
{beer}	1.00	{cheese & crackers} => {beer}	0.50
{bread}	1.00	{bread} => {cheese & beer}	0.33
{cheese & crackers}	1.00	{wine} => {cheese & beer}	0.25
{cheese & beer}	1.00	{cheese & crackers} => {bread}	0.50
{cheese} => {wine}	0.20	{cheese} > {wine} => {beer}	1.00
{cheese} => {beer}	0.20	{crackers} > {wine} => {beer}	1.00
{wine} => {beer}	1.00	{wine} > {beer} => {cheese}	0.50
{crackers} => {wine}	0.50	{bread} > {wine} => {beer}	1.00
{crackers} => {beer}	0.50	{bread} > {wine} => {cheese}	1.00
{wine} => {cheese}	0.75	{beer} > {cheese} => {bread}	0.33
{beer} => {cheese}	0.60	{beer} > {crackers} => {bread}	1.00
{bread} => {wine}	0.33	{crackers} > {wine} => {cheese}	0.50
{bread} => {beer}	0.33	{crackers} > {beer} => {cheese}	0.50
{bread} => {cheese}	0.33	{cheese & crackers} > {wine} => {beer}	1.00
{beer} => {bread}	0.20	{bread} > {wine} => {cheese & beer}	1.00
{beer} => {crackers}	0.20	{beer} > {cheese & crackers} => {bread}	1.00
{cheese} => {bread}	0.20	{crackers} > {wine} > {beer} => {cheese}	0.50
{crackers} => {bread}	0.50		

Generated Model/Scoring

Predicted Values

When you pass data records into a Sequence Rules model, the model handles the records in a time-dependent manner (or order-dependent, if no timestamp field was used to build the model). Records should be sorted by the ID field and timestamp field (if present).

For each record, the rules in the model are compared to the set of transactions processed for the current ID so far, including the current record and any previous records with the same ID and earlier timestamp. The *k* rules with the highest confidence values that apply to this set of transactions are used to generate the *k* predictions for the record, where *k* is the number of predictions specified when the model was built. (If multiple rules predict the same outcome for the transaction set, only the rule with the highest confidence is used.)

Note that the predictions for each record do not necessarily depend on that record's transactions. If the current record's transactions do not trigger a specific rule, rules will be selected based on the previous transactions for the current ID. In other words, if the current record doesn't add any useful predictive information to the sequence, the prediction from the last useful transaction for this ID is carried forward to the current record.

For example, suppose you have a Sequence Rule model with the single rule

```
Jam -> Bread (0.66)
```

and you pass it the following records:

ID	Purchase	Prediction
001	jam	bread
001	milk	bread

Notice that the first record generates a prediction of *bread*, as you would expect. The second record also contains a prediction of *bread*, because there's no rule for *jam* followed by *milk*; therefore the *milk* transaction doesn't add any useful information, and the rule Jam -> Bread still applies.

Confidence

The confidence associated with a prediction is the confidence of the rule that produced the prediction. For more information, see the topic "Confidence" on p. 340.

Blank Handling

Blanks are ignored by the sequence rules algorithm. The algorithm will handle records containing blanks for input fields, but such a record will not be considered to match any rule containing one or more of the fields for which it has blank values.

Note that the sequence algorithm generates rules that have a max length of the users in the dataset. For example, if you have transactions such as the following, the algorithm won't find a sequence of event codes A -> B -> C, because there are only two users in the dataset.

User	Event Code
1	A
1	B
1	C
1	A
1	B
1	C
2	A
2	B
2	C

Simulation algorithms

Simulation in IBM® SPSS® Modeler refers to simulating input data to predictive models using the Monte Carlo method and evaluating the model based on the simulated data. You do this by using the Simulation Generation (also known as SimGen) source node. The distribution of predicted target values can then be used to evaluate the likelihood of various outcomes.

Simulation algorithms

Creating a simulation includes specifying distributions for all inputs to a predictive model that are to be simulated. When historical data are present, the distribution that most closely fits the data for each input can be determined using the algorithms described in this section.

Notation

The following notation is used throughout this section unless otherwise stated:

Table 33-1
Notation

Notation	Description
x_i	Value of the input variable in the *i*th case of the historical data
w_i	Frequency weight associated with the *i*th case of the historical data
W	Total effective sample size accounting for frequency weights
$\overline{x}_{obs}$	Sample mean
s^2_{obs}	Sample variance
s_{obs}	Sample standard deviation

Distribution fitting

The historical data for a given input is denoted by:

$$\underset{x}{\rightarrow} = x_1,\ \ x_2,\ \ \ldots,\ \ x_n$$

The total effective sample size is:

$$W = \sum_{i=1}^{n} w_i$$

The observed sample mean, sample variance and sample standard deviation are:

$$\overline{x}_{obs} = \frac{1}{W} \sum_{i=1}^{n} w_i x_i$$

$$s^2_{obs} = \frac{1}{W-1} \sum_{i=1}^{n} w_i(x_i - \overline{x}_{obs})^2$$

$$s_{obs} = \sqrt{s^2_{obs}}$$

Parameter estimation for most distributions is based on the maximum likelihood (ML) method, and closed-form solutions for the parameters exist for many of the distributions. There is no closed-form ML solution for the distribution parameters for the following distributions: negative binomial, beta, gamma and Weibull. For these distributions, the Newton-Raphson method is used. This approach requires the following information: the log-likelihood function, the gradient vector, the Hessian matrix, and the initial values for the iterative Newton-Raphson process.

Discrete distributions

Distribution fitting is supported for the following discrete distributions: binomial, categorical, Poisson and negative binomial.

Binomial distribution: parameter estimation

The probability mass function for a random variable x with a binomial distribution is:

$$Bin\,(x; N, P) = \binom{N}{x} P^x (1-P)^{N-x}, for\ x = 0, 1, \ldots, N$$

where $0 \leq P \leq 1$ is the probability of success. The binomial distribution is used to describe the total number of successes in a sequence of N independent Bernoulli trials. The parameter estimates for the binomial distribution using the method of moments (see Johnson & Kotz (2005) for details) are:

$$\hat{P} = \begin{cases} 1 - \frac{s^2_{obs}}{\overline{x}_{obs}}, & \overline{x}_{obs} > s^2_{obs} \\ NaN, & \overline{x}_{obs} < s^2_{obs} \end{cases}$$

where *NaN* implies that the binomial distribution would not be an appropriate distribution to fit the data under this criterion, and where

$$\hat{N} = \frac{\overline{x}_{obs}}{\hat{P}}$$

If $\hat{N}$ is not an integer, then the parameter estimates are:

$$\hat{N}^* = \left[\hat{N} + 0.5\right]$$

$$\hat{P}^* = \frac{\overline{x}_{obs}}{\hat{N}^*}$$

where $[x]$ denotes the integer part of x.

Categorical distribution: parameter estimation

The categorical distribution can be considered a special case of the multinomial distribution in which $N = 1$. Suppose x_i, $i = 1, 2, \ldots, n$, has the categorical distribution and its categorical values are denoted as 1, 2, …, J. Then an indicator variable of x_i for category j can be denoted as

$$x_{i,j} = \begin{cases} 1 & \text{if } x_i = j \\ 0 & \text{otherwise} \end{cases}$$

and the corresponding probability is P_j. Then the probability mass function for a random variable x_i with the categorical distribution can be described based on $x_{i,j}$ and P_j as follows:

$$Categorical(x_i; P_1, \ldots, P_J) = \prod_{j=1}^{J} P_j^{x_{i,j}}, \text{ with} \sum_{j=1}^{J} P_j = 1$$

The parameter estimates for $P_j, j = 1, \ldots, J$, are:

$$\hat{P}_j = \frac{\sum_{i=1}^{n} w_i x_{i,j}}{W}, j = 1, \ldots, J$$

Poisson distribution: parameter estimation

The probability mass function for a random variable x with a Poisson distribution is:

$$Pois(x; \lambda) = \frac{e^{-\lambda}\lambda^x}{x!}, \ for\ x = 0, 1, \ldots$$

where $\lambda > 0$ is the rate parameter of the Poisson distribution. The parameter of the Poisson distribution can be estimated as:

$$\hat{\lambda} = \overline{x}_{obs}$$

Negative binomial distribution: parameter estimation

The distribution fitting component for simulation supports the parameterization of the negative binomial distribution that describes the distribution of the number of failures before the r^{th} success. For this parameterization, the probability mass function for a random variable x is:

$$NB(x; r, \theta) = \binom{x + r - 1}{x} \theta^r (1 - \theta)^x, \text{ for } x = 0, 1, \ldots$$

where $r \geq 0,\ 0 \leq \theta \leq 1$ are the two distribution parameters. There is no closed-form solution for the parameters *r* and θ, so the Newton-Raphson method with step-halving will be used. The method requires the following information:

(1) The log likelihood function

$$L = \sum_{i=1}^{n} w_i \ln\Gamma(x_i + r) - \sum_{i=1}^{n} w_i \mathrm{ln} x_i! - W \mathrm{ln}\Gamma(r) + Wr\mathrm{ln}(\theta) + \ln(1-\theta)\sum_{i=1}^{n} w_i x_i$$

(2) The gradient (1st derivative) vector with respect to *r* and θ

$$s = \begin{bmatrix} \frac{\partial L}{\partial \theta} \\ \frac{\partial L}{\partial r} \end{bmatrix} = \begin{bmatrix} \frac{Wr}{\theta} - \frac{1}{(1-\theta)} \sum_{i=1}^{n} w_i x_i \\ \sum_{i=1}^{n} w_i \psi(x_i + r) - W\psi(r) + W\mathrm{ln}(\theta) \end{bmatrix}$$

where $\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ is a digamma function, which is the derivative of the logarithm of the gamma function, evaluated at α.

(3) The Hessian (2nd derivative) matrix with respect to *r* and θ (since the Hessian matrix is symmetric, only the lower triangular portion is displayed)

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial \theta^2} & \\ \frac{\partial^2 L}{\partial r \partial \theta} & \frac{\partial^2 L}{\partial r^2} \end{bmatrix} = \begin{bmatrix} -\frac{Wr}{\theta^2} - \frac{1}{(1-\theta)^2} \sum_{i=1}^{n} w_i x_i & \\ \frac{W}{\theta} & \sum_{i=1}^{n} w_i \psi'(x_i + r) - W\psi'(r) \end{bmatrix}$$

where $\psi'(\alpha)$ is the trigamma function, or the derivative of the digamma function.

(4) The initial values of θ and *r* can be obtained from the closed-form estimates using the method of moments:

$$r^{(0)} = \begin{cases} \frac{\overline{x}_{obs}^2}{s_{obs}^2 - \overline{x}_{obs}} & \text{if } s_{obs}^2 > \overline{x}_{obs} \\ 1 & \text{otherwise} \end{cases}$$

$$\theta^{(0)} = \frac{r^{(0)}}{r^{(0)} + \overline{x}_{obs}}$$

Note

An alternative parameterization of the negative binomial distribution describes the distribution of the number of trials before the r^{th} success. Although it is not supported in distribution fitting, it is supported in simulation when explicitly specified by the user. The probability mass function for this parameterization, for a random variable x is:

$$NB(x; r, \theta) = \binom{x-1}{r-1} \theta^r (1-\theta)^{x-r}, \text{ for } x \geq r$$

where $r \geq 0,\ 0 \leq \theta \leq 1$ are the two distribution parameters.

Continuous distributions

Distribution fitting is supported for the following continuous distributions: triangular, uniform, normal, lognormal, exponential, beta, gamma and Weibull.

Triangular distribution: parameter estimation

The probability density function for a random variable x with a triangular distribution is:

$$Triag(x; a, m, b) = \begin{cases} \frac{2}{(b-a)}\frac{(x-a)}{(m-a)}, & x \in [a, m) \\ \frac{2}{(b-a)}, & x = m \\ \frac{2}{(b-a)}\frac{(b-x)}{(b-m)}, & x \in (m, b] \\ 0, & x \notin [a, b] \end{cases}$$

such that $a \leq m \leq b$. Parameter estimates of the triangular distribution are:

$$\hat{a} = \min\{x_1, x_2, \ldots, x_n\}$$

$$\hat{b} = \max\{x_1, x_2, \ldots, x_n\}$$

$$\hat{m} = mode\{x_1, x_2, \ldots, x_n\}$$

Since the calculation of the mode for continuous data may be ambiguous, we transform the parameter estimates and use the method of moments as follows (see Kotz and Rene van Dorp (2004) for details):

$$z_i = \frac{x_i - \hat{a}}{\hat{b} - \hat{a}}$$

$$\theta = \frac{m - \hat{a}}{\hat{b} - \hat{a}}$$

$$\overline{z} = \frac{1}{W}\sum_{i=1}^{n} w_i z_i$$

From the method of moments we obtain

$$\hat{\theta} = 3\overline{z} - 1$$

from which it follows that

$$\hat{m} = \hat{a} + \left(\hat{b} - \hat{a}\right) \times (3\overline{z} - 1)$$

Note: For very skewed data or if the actual mode equals a or b, the estimated mode, $\hat{m}$, may be less than â or greater than $\hat{\text{b}}$. In this case, the adjusted mode, defined as below, is used:

$$Adj.\ \hat{m} = \begin{cases} \hat{a} & \text{if } \hat{m} < \hat{a} \\ \hat{b} & \text{if } \hat{m} > \hat{b} \end{cases}$$

Uniform distribution: parameter estimation

The probability density function for a random variable x with a uniform distribution is:

$$U\left(x; a, b\right) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \\ 0, & x \notin [a, b] \end{cases}$$

where a is the minimum and b is the maximum among the values of $\underset{x}{\rightarrow}$. Hence, the parameter estimates of the uniform distribution are:

$$\hat{a} = \min\left\{x_1, x_2, \ldots, x_n\right\}$$

$$\hat{b} = \max\left\{x_1, x_2, \ldots, x_n\right\}$$

Normal distribution: parameter estimation

The probability density function for a random variable x with a normal distribution is:

$$Nor(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2},\ -\infty < x < \infty$$

Here, μ is the measure of centrality and σ is the measure of dispersion of the normal distribution. The parameter estimates of the normal distribution are:

$$\hat{\mu} = \overline{x}_{obs}$$

$$\hat{\sigma} = \sqrt{\frac{1}{W} \sum_{i=1}^{n} w_i (x_i - \overline{x}_{obs})^2} = \sqrt{\frac{(W-1)\, s_{obs}^2}{W}}$$

Lognormal distribution: parameter estimation

The lognormal distribution is a probability distribution where the natural logarithm of a random variable follows a normal distribution. In other words, if x has a lognormal(μ, σ) distribution, then ln(x) has a normal(ln(μ),σ) distribution. The probability density function for a random variable x with a lognormal distribution is:

$$LN(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma x}\ e^{-\frac{1}{2}\left(\frac{\ln x - \ln \mu}{\sigma}\right)^2}, 0 < x < \infty$$

Define $\overline{lx}_{obs} = \frac{1}{W}\sum_{i=1}^{n} w_i \ln x_i$

Parameter estimates for the lognormal distribution are:

$$\hat{\mu} = e^{\overline{lx}_{obs}}$$

$$\hat{\sigma} = \sqrt{\frac{1}{W} \sum_{i=1}^{n} w_i(\ln x_i - \ln \hat{\mu})^2}$$

Exponential distribution: parameter estimation

The probability density function for a random variable x with an exponential distribution is:

$$Exp(x;\lambda) = \lambda e^{-\lambda x}, \text{ for } x \geq 0 \text{ and } \lambda > 0$$

The estimate of the parameter for the exponential distribution is:

$$\hat{\lambda} = \frac{1}{\overline{x}_{obs}}$$

Beta distribution: parameter estimation

The probability density function for a random variable x with a beta distribution is:

$$Beta(x;\alpha,\beta) = \frac{1}{\mathrm{B}(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \;\; \alpha,\beta > 0$$

where,

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

There is no closed-form solution for the parameters α and β, so the Newton-Raphson method with step-halving will be used. The method requires the following information:

(1) The log likelihood function

$$L = W\ln(\Gamma(\alpha+\beta)) - W\ln(\Gamma(\alpha)) - W\ln(\Gamma(\beta))$$

$$+(\alpha-1)\sum_{i=1}^{n} w_i \ln x_i + (\beta-1)\sum_{i=1}^{n} w_i \ln(1-x_i)$$

(2) The gradient (1st derivative) vector with respect to α and β

$$s = \begin{bmatrix} \frac{\partial L}{\partial \alpha} \\ \frac{\partial L}{\partial \beta} \end{bmatrix} = \begin{bmatrix} W\psi(\alpha+\beta) - W\psi(\alpha) + \sum_{i=1}^{n} w_i \ln(x_i) \\ W\psi(\alpha+\beta) - W\psi(\beta) + \sum_{i=1}^{n} w_i \ln(1-x_i) \end{bmatrix}$$

where $\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ is a digamma function, which is the derivative of the logarithm of the gamma function, evaluated at α.

(3) The Hessian (2nd derivative) matrix with respect to *α* and *β* (since the Hessian matrix is symmetric, only the lower triangular portion is displayed)

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial \alpha^2} & \\ \frac{\partial^2 L}{\partial \alpha \partial \beta} & \frac{\partial^2 L}{\partial \beta^2} \end{bmatrix} = \begin{bmatrix} W(\psi'(\alpha+\beta) - \psi'(\alpha)) & \\ W\psi'(\alpha+\beta) & W(\psi'(\alpha+\beta) - \psi'(\beta)) \end{bmatrix}$$

where $\psi'(\alpha)$ is the trigamma function, or the derivative of the digamma function.

(4) The initial values of *α* and *β* can be obtained from the closed-form estimates using the method of moments:

$$\alpha^{(0)} = \overline{x}_{obs}\left(\frac{\overline{x}_{obs}(1-\overline{x}_{obs})}{{s_{obs}}^2} - 1\right)$$

$$\beta^{(0)} = (1-\overline{x}_{obs})\left(\frac{\overline{x}_{obs}(1-\overline{x}_{obs})}{{s_{obs}}^2} - 1\right)$$

Gamma distribution: parameter estimation

The probability density function for a random variable x with a gamma distribution is:

$$Gamma(x;\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \text{ for } x \geq 0 \text{ and } \alpha,\beta > 0$$

If α is a positive integer, then the gamma function is given by: $\Gamma(\alpha) = (\alpha - 1)!$

There is no closed-form solution for the parameters *α* and *β*, so the Newton-Raphson method with step-halving will be used. The method requires the following information:

(1) The log likelihood function

$$L = W\alpha \ln \beta - W \ln\Gamma(\alpha) + (\alpha - 1)\sum_{i=1}^{n} w_i \ln x_i - \beta \sum_{i=1}^{n} w_i x_i$$

(2) The gradient (1st derivative) vector with respect to *α* and *β*

$$s = \begin{bmatrix} \frac{\partial L}{\partial \alpha} \\ \frac{\partial L}{\partial \beta} \end{bmatrix} = \begin{bmatrix} W \ln \beta - W\psi(\alpha) + \sum_{i=1}^{n} w_i \ln x_i \\ \frac{W\alpha}{\beta} - \sum_{i=1}^{n} w_i x_i \end{bmatrix}$$

where $\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ is a digamma function, which is the derivative of the logarithm of the gamma function, evaluated at α.

(3) The Hessian (2nd derivative) matrix with respect to *α* and *β* (since the Hessian matrix is symmetric, only the lower triangular portion is displayed)

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial \alpha^2} & \\ \frac{\partial^2 L}{\partial \alpha \partial \beta} & \frac{\partial^2 L}{\partial \beta^2} \end{bmatrix} = \begin{bmatrix} -W\psi'(\alpha) & \\ \frac{W}{\beta} & -\frac{W\alpha}{\beta^2} \end{bmatrix}$$

where $\psi'(\alpha)$ is the trigamma function, or the derivative of the digamma function.

(4) The initial values of *α* and *β* can be obtained from the closed-form estimates using the method of moments:

$$\alpha^{(0)} = \left(\frac{\overline{x}_{obs}}{s_{obs}}\right)^2$$

$$\beta^{(0)} = \frac{\overline{x}_{obs}}{s_{obs}^2}$$

Weibull distribution: parameter estimation

Distribution fitting for the Weibull distribution is restricted to the two-parameter Weibull distribution, whose probability density function is given by:

$$Weib(x;\beta,\gamma) = \frac{\gamma}{\beta}\left(\frac{x}{\beta}\right)^{\gamma-1} e^{-\left(\frac{x}{\beta}\right)^{\gamma}}, \text{ for } x \geq 0 \text{ and } \beta,\gamma > 0$$

There is no closed-form solution for the parameters *β* and *γ*, so the Newton-Raphson method with step-halving will be used. The method requires the following information:

(1) The log likelihood function

$$L = W(\ln\gamma - \gamma\ln\beta) + (\gamma - 1)\sum_{i=1}^{n} w_i \ln(x_i) - \sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma}$$

(2) The gradient (1st derivative) vector with respect to *β* and *γ*

$$s = \begin{bmatrix} \frac{\partial L}{\partial \beta} \\ \frac{\partial L}{\partial \gamma} \end{bmatrix} = \begin{bmatrix} -\frac{W\gamma}{\beta} + \frac{\gamma}{\beta}\sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma} \\ \frac{W}{\gamma} - W\ln\beta + \sum_{i=1}^{n} w_i \ln(x_i) - \sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma} \ln\left(\frac{x_i}{\beta}\right) \end{bmatrix}$$

(3) The Hessian (2nd derivative) matrix with respect to *β* and *γ* (since the Hessian matrix is symmetric, only the lower triangular portion is displayed)

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial \beta^2} & \\ \frac{\partial^2 L}{\partial \beta \partial \gamma} & \frac{\partial^2 L}{\partial \gamma^2} \end{bmatrix}$$

where

$$\frac{\partial^2 L}{\partial \beta^2} = \frac{\gamma}{\beta^2}\left[W - (\gamma+1)\sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma}\right]$$

$$\frac{\partial^2 L}{\partial \beta \partial \gamma} = -\frac{1}{\beta}\left[W - \sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma} - \gamma \sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma} \ln\left(\frac{x_i}{\beta}\right)\right]$$

$$\frac{\partial^2 L}{\partial \gamma^2} = \ -\frac{W}{\gamma^2} - \sum_{i=1}^{n} w_i \left(\frac{x_i}{\beta}\right)^{\gamma}\left[\ln\left(\frac{x_i}{\beta}\right)\right]^2$$

(4) The initial values of β and γ are given by:

$$\gamma^{(0)} = 1$$

$$\beta^{(0)} = 1$$

Goodness of fit measures

Goodness of fit measures are used to determine the distribution that most closely fits the data. For discrete distributions, the Chi-Square test is used. For continuous distributions, the Anderson-Darling test or the Kolmogorov-Smirnov test is used.

Discrete distributions

The Chi-Square goodness of fit test is used for discrete distributions (Dirk P. Kroese, 2011). The Chi-Square test statistic has the following form:

$$T = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

where,

Table 33-2
Notation

Notation	Description
k	The number of classes, as defined in the table below for each discrete distribution
O_i	The total observed frequency for class *i*

Notation	**Description**
PDF(i)	Probability density function of the fitted distribution. For the Poisson and negative binomial distributions, the density function for the last class is computed as $\text{PDF}(k) = 1 - \sum_{i=1}^{k-1} \text{PDF}(i)$
E_i	Expected frequency for class *i*: E_i = *W*PDF(i)*
W	The total effective sample size

For large *W*, the above statistic follows the Chi-Square distribution:

$$T \sim \chi^2_{k-1-r}$$

where *r* = *number of parameters estimated from the data*. The following table provides the values of *k* and *r* for the various distributions. The value *Max* in the table is the observed maximum value.

Distribution	**Notation**	***k* (classes)**	***r* (parameters)**
Binomial	$Bin\,(x; N, P)$	*N+1*	2
Categorical	$Categorical\,(x; p_1, \ldots, p_J)$	*J*	*J*-1
Poisson	$Pois(x; \lambda)$	*Max* + 1	1
Negative binomial	$NB(x; r, \theta)$	*Max* + 1	2

This Chi-Square test is valid only if all values of $E_i \geq 5$.

The p-value for the Chi-Square test is then calculated as:

$$p = 1 - F\left(T, \chi^2_{k-1-r}\right)$$

where $F\left(T, \chi^2_{k-1-r}\right) =$ *CDF* of the Chi-Square distribution.

Note: The p-value cannot be calculated for the Categorical distribution since the number of degrees of freedom is zero.

Continuous distributions

For continuous distributions, the Anderson-Darling test or the Kolmogorov-Smirnov test is used to determine goodness of fit. The calculation consists of the following steps:

1. Transform the data to a Uniform(0,1) distribution
2. Sort the transformed data to generate the Order Statistics
3. Calculate the Anderson-Darling or Kolmogorov-Smirnov test statistic
4. Compute the approximate p-value associated with the test statistic

The first two steps are common to both the Anderson-Darling and Kolmogorov-Smirnov tests. The original data are transformed to a Uniform(0,1) distribution using the transformation:

$$y = F\left(x\right)$$

where the transformation function $F\left(x\right)$ is given in the table below for each of the supported distributions.

Distribution	**Transformation F(x)**
$Triag(x; a, m, b)$	$\begin{cases} \frac{1}{(b-a)}\frac{(x-a)^2}{(m-a)}, & x \in [a, m) \\ \frac{(m-a)}{(b-a)}, & x = m \\ 1 - \frac{1}{(b-a)}\frac{(b-x)^2}{(b-m)}, & x \in \ (m, b] \\ 0, & x \notin [a, b] \end{cases}$
$U\left(x; a, b\right)$	$\frac{x-a}{b-a}$
$Nor(x, \mu, \sigma)$	$\Phi\left(\frac{x-\mu}{\sigma}\right)$
$LN(x, \mu, \sigma)$	$\Phi\left(\frac{\ln x - \ln \mu}{\sigma}\right)$
$Exp(x; \lambda)$	$1 - e^{-\lambda x}$
$Beta(x; \alpha, \beta)$	$\int\limits_0^x \frac{1}{\mathrm{B}(\alpha,\beta)}\ t^{\alpha-1}(1-t)^{\beta-1}dt$
$Gamma(x; \alpha, \beta)$	$\int\limits_0^x \frac{\beta^\alpha}{\Gamma(\alpha)}t^{\alpha-1}e^{-\beta t}dt$
$Weib(x; \beta, \gamma)$	$1 - e^{-\left(\frac{x}{\beta}\right)^\gamma}$

The transformed data points y_i are sorted in ascending order to generate the Order Statistics:

$$y_{(1)} \leq y_{(2)} \leq \ldots \leq y_{(n-1)} \leq y_{(n)}$$

Define w_i^* to be the corresponding frequency weight for $y_{(i)}$. The cumulative frequency up to and including $y_{(i)}$ is defined as:

$$W_i^* = \sum_{k=1}^{i} w_i^*$$

and where we define $W_0^* = 0$.

Anderson-Darling test

The Anderson-Darling test statistic is given by:

$$z = -W_n^* - \frac{1}{W_n^*}\sum_{i=1}^{n} w_i^*\left(2W_{i-1}^* + w_i^*\right)\ln\left(y_{(i)}\right) + \frac{1}{W_n^*}\sum_{i=1}^{n} w_i^*\left(2W_{i-1}^* + w_i^*\right)\ln\left(1 - y_{(i)}\right)$$
$$-2\sum_{i=1}^{n} w_i^*\ln\left(1 - y_{(i)}\right)$$

For more information, see the topic “Anderson-Darling statistic with frequency weights” on p. 360.

The approximate p-value for the Anderson-Darling statistic can be computed for the following distributions: uniform, normal, lognormal, exponential, Weibull and gamma. The p-value is not available for the triangular and beta distributions.

Uniform distribution: p-value

The p-value for the Anderson-Darling statistic is computed based on the following result, provided by Marsaglia (2004):

$$p = \begin{cases} 1 - z^{-1/2}e^{-1.2337141/z}g\left(z\right) & for\ z \in (0,2) \\ 1 - e^{-e^{(1.0776-(2.30695-(0.43424-(0.082433-(0.008056-0.0003146z)z)z)z)z)}} & for\ z \in [2,\infty) \end{cases}$$

where

$$g\left(z\right) = (2.00012 + (0.247105 - (0.0649821 - (0.0347962 - (0.0116720 - 0.00168691z)z)z)z)z)$$

Normal and lognormal distributions: p-value

The p-value for the Anderson-Darling statistic is computed based on the following result, provided by D’Agostino and Stephens (1986):

$$p = \begin{cases} 1 - \exp\left(-13.436 + 101.14z^* - 223.73z^{*2}\right), & z^* \leq 0.2 \\ 1 - \exp\left(-8.318 + 42.796z^* - 59.938z^{*2}\right), & 0.2 < z^* \leq 0.34 \\ \exp\left(0.9177 - 4.279z^* - 1.38z^{*2}\right), & 0.34 < z^* \leq 0.6 \\ \exp\left(1.2937 - 5.709z^* + 0.0186z^{*2}\right), & 0.6 < z^* \leq 153.467 \\ 0 & z^* > 153.467 \end{cases}$$

where

$$z^* = z\left(1.0 + 0.75/W_n^* + 2.25/W_n^{*2}\right)$$

Exponential distribution: p-value

The p-value for the Anderson-Darling statistic is computed based on the following result, provided by D'Agostino and Stephens (1986):

$$p = \begin{cases} 1-\exp\left(-12.2204+67.459z^{*}-110.3z^{*2}\right), & z^{*}\leq 0.260 \\ 1-\exp\left(-6.1327+20.218z^{*}-18.663z^{*2}\right), & 0.260<z^{*}\leq 0.510 \\ \exp\left(0.9209-3.353z^{*}+0.3z^{*2}\right), & 0.510<z^{*}\leq 0.950 \\ \exp\left(0.731-3.009z^{*}+0.15z^{*2}\right), & 0.950<z^{*}\leq 10.03 \\ 0 & z^{*}>10.03 \end{cases}$$

where

$$z^{*} = z\,(1.0+0.6/W_n^{*})$$

Weibull distribution: p-value

The p-value for the Anderson-Darling statistic is computed based on Table 33-3 below, provided by D'Agostino and Stephens (1986). First, the adjusted Anderson-Darling statistic is computed from:

$$z^{*} = z\left(1+0.2/\sqrt{W_n^{*}}\right)$$

If the value of z^{*} is between two probability levels (in the table), then linear interpolation is used to estimate the p-value. For example, if $z^{*}=0.543$ which is between $z_1^{*}=0.474$ and $z_2^{*}=0.637$, then the corresponding probabilities of z_1^{*} and z_2^{*} are $\mathrm{p}_1=0.25$ and $\mathrm{p}_2=0.1$ respectively. Then the p-value of z^{*} is computed as

$$p = \frac{p_2-p_1}{z_2^{*}-z_1^{*}}(z^{*}-z_1^{*})+p_1 = \frac{0.1-0.25}{0.637-0.474}(0.543-0.474)+0.25 = 0.1865$$

If the value of z^{*} is less than the smallest critical value in the table, then the p-value is ≥ 0.25; and if z^{*} is greater than the largest critical value in the table, then the p-value is ≤ 0.01.

Table 33-3
Upper tail probability and corresponding critical values for the Anderson-Darling test, for the Weibull distribution

p-value	**0.25**	**0.10**	**0.05**	**0.025**	**0.01**
$z\left(1+0.2/\sqrt{W_n^{*}}\right)$	0.474	0.637	0.757	0.877	1.038

Gamma distribution: p-value

Table 33-4, which is provided by D'Agostino and Stephens (1986), is used to compute the p-value of the Anderson-Darling test for the gamma distribution. First, the appropriate row in the table is determined from the range of the parameter α. Then linear interpolation is used to compute the p-value, as done for the Weibull distribution. For more information, see the topic "Weibull distribution: p-value" on p. 356.

If the test statistic is less than the smallest critical value in the row, then the p-value is ≥ 0.25; and if the test statistic is greater than the largest critical value in the row, then the p-value is ≤ 0.005.

Table 33-4

Upper tail probability and corresponding critical values for the Anderson-Darling test, for the gamma distribution with estimated parameter α

p-value	**0.25**	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**
$\alpha \in (0,1]$	0.486	0.657	0.786	0.917	1.092	1.227
$\alpha \in (1,8]$	0.473	0.637	0.759	0.883	1.048	1.173
$\alpha \in (8,\infty)$	0.470	0.631	0.752	0.873	1.035	1.159

Kolmogorov-Smirnov test

The Kolmogorov-Smirnov test statistic, D_n, is given by:

$$D^+ = max_i \left| y_{(i)} - \frac{W_i^*}{W_n^*} \right| \qquad D^- = max_i \left| y_{(i)} - \frac{W_i^* - 1}{W_n^*} \right|$$

$$D_n = max\left(D^+, D^-\right)$$

Computation of the p-value is based on the modified Kolmogorov-Smirnov statistic, which is distribution specific.

Uniform distribution: p-value

The procedure proposed by Kroese (2011) is used to compute the p-value of the Kolmogorov-Smirnov statistic for the uniform distribution. First, the modified Kolmogorov-Smirnov statistic is computed as

$$D = \sqrt{W_n^*}\ D_n$$

The corresponding p-value is computed as follows:

1. Set k=100
2. Define $\alpha_k = \sum_{i=-k}^{k} (-1)^i e^{-2(iD)^2}$
3. Calculate α_k and α_{k+1}
4. If $|\alpha_k - \alpha_{k+1}| \geq 10^{-5}$ set k=k+1 and repeat step 2; otherwise, go to step 5.
5. p-value $= 1 - \alpha_k$

Normal and lognormal distributions: p-value

The modified Kolmogorov-Smirnov statistic is

$$D = D_n \left(\sqrt{W_n^*} - 0.01 + \frac{0.85}{\sqrt{W_n^*}} \right)$$

The p-value for the Kolmogorov-Smirnov statistic is computed based on Table 33-5 below, provided by D'Agostino and Stephens (1986). If the value of D is between two probability levels, then linear interpolation is used to estimate the p-value. For more information, see the topic "Weibull distribution: p-value" on p. 356.

If D is less than the smallest critical value in the table, then the p-value is ≥ 0.15; and if D is greater than the largest critical value in the table, then the p-value is ≤ 0.01.

Table 33-5
Upper tail probability and corresponding critical values for the Kolmogorov-Smirnov test, for the Normal and Lognormal distributions

***p*-value**	**0.15**	**0.10**	**0.05**	**0.025**	**0.01**
D	0.775	0.819	0.895	0.995	1.035

Exponential distribution: p-value

The modified Kolmogorov-Smirnov statistic is

$$D = (D_n - 0.2/W_n^*) \left(\sqrt{W_n^*} + 0.26 + 0.5/\sqrt{W_n^*} \right)$$

The p-value for the Kolmogorov-Smirnov statistic is computed based on Table 33-6 below, provided by D'Agostino and Stephens (1986). If the value of D is between two probability levels, then linear interpolation is used to estimate the p-value. For more information, see the topic "Weibull distribution: p-value" on p. 356.

If D is less than the smallest critical value in the table, then the p-value is ≥ 0.15; and if D is greater than the largest critical value in the table, then the p-value is ≤ 0.01.

Table 33-6
Upper tail probability and corresponding critical values for the Kolmogorov-Smirnov test, for the Exponential distribution

***p*-value**	**0.15**	**0.10**	**0.05**	**0.025**	**0.01**
D	0.926	0.995	1.094	1.184	1.298

Weibull distribution: p-value

The modified Kolmogorov-Smirnov statistic is

$$D = \sqrt{W_n^*} D_n$$

The p-value for the Kolmogorov-Smirnov statistic is computed based on Table 33-7 below, provided by D'Agostino and Stephens (1986). If the value of D is between two probability levels, then linear interpolation is used to estimate the p-value. For more information, see the topic "Weibull distribution: p-value" on p. 356.

If D is less than the smallest critical value in the table, then the p-value is ≥ 0.10; and if D is greater than the largest critical value in the table, then the p-value is ≤ 0.01.

Table 33-7
Upper tail probability and corresponding critical values for the Kolmogorov-Smirnov test, for the Weibull distribution

***p*-value**	**0.10**	**0.05**	**0.025**	**0.01**
D	1.372	1.477	1.557	1.671

Gamma distribution: p-value

The modified Kolmogorov-Smirnov statistic is

$$D = D_n \quad \left(\sqrt{W_n^*} + 0.3/\sqrt{W_n^*}\right)$$

The p-value for the Kolmogorov-Smirnov statistic is computed based on Table 33-8 below, provided by D'Agostino and Stephens (1986). If the value of D is between two probability levels, then linear interpolation is used to estimate the p-value. For more information, see the topic "Weibull distribution: p-value" on p. 356.

If D is less than the smallest critical value in the table, then the p-value is ≥ 0.25; and if D is greater than the largest critical value in the table, then the p-value is ≤ 0.005.

Table 33-8
Upper tail probability and corresponding critical values for the Kolmogorov-Smirnov test, for the Gamma distribution

***p*-value**	**0.25**	**0.20**	**0.15**	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**
D	0.74	0.780	0.800	0.858	0.928	0.990	1.069	1.13

Determining the recommended distribution

The distribution fitting module is invoked by the user, who may specify an explicit set of distributions to test or rely on the default set, which is determined from the measurement level of the input to be fit. For continuous inputs, the user specifies either the Anderson-Darling test (the default) or the Kolmogorov-Smirnov test for the goodness of fit measure (for ordinal and nominal inputs, the Chi-Square test is always used). The distribution fitting module then returns the values of the specified test statistic along with the calculated p-values (if available) for each of the tested distributions, which are then presented to the user in ascending order of the test statistic. The recommended distribution is the one with the minimum value of the test statistic.

The above approach yields the distribution that most closely fits the data. However, if the p-value of the recommended distribution is less than 0.05, then the recommended distribution may not provide a close fit to the data.

Anderson-Darling statistic with frequency weights

To obtain the expression for the Anderson-Darling statistic with frequency weights, we first give the expression where the frequency weight of each value is 1:

$$\begin{aligned} z &= -n - \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(2i-1)\big[\ln\big(y_{(i)}\big(1-y_{(n+1-i)}\big)\big)\big] \\ &= -n - \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(2i-1)\big[\ln\big(y_{(i)}\big)+\ln\big(1-y_{(n+1-i)}\big)\big] \\ &= -n - \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(2i-1)\big[\ln\big(y_{(i)}\big)\big] - \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left(2\left(n+1-i\right)-1\right)\big[\ln\big(1-y_{(i)}\big)\big] \\ &= -n - \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(2i-1)\big[\ln\big(y_{(i)}\big)\big] - \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(1-2i)\big[\ln\big(1-y_{(i)}\big)\big] - \textstyle\sum_{i=1}^{n}2\big[\ln\big(1-y_{(i)}\big)\big] \\ &= A+B+C+D \end{aligned}$$

If there is a frequency weight variable, then the corresponding four terms of the above expression are given by:

$$A = -W_n^*$$

$$\begin{aligned} B &= -\tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}\sum_{j=1}^{w_i^*}\left(2\big(W_{i-1}^*+j\big)-1\right)\left[\ln\big(y_{(i)}\big)\right] \\ &= -\tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}w_i^*\big(2W_{i-1}^*-1\big)\ln\big(y_{(i)}\big) - \tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}w_i^*(w_i^*+1)\ln\big(y_{(i)}\big) \\ &= -\tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}w_i^*\big(2W_{i-1}^*+w_i^*\big)\ln\big(y_{(i)}\big) \end{aligned}$$

$$\begin{aligned} C &= -\tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}\sum_{j=1}^{w_i^*}\left(1-2\big(W_{i-1}^*+j\big)\right)\left[\ln\big(1-y_{(i)}\big)\right] \\ &= -\tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}w_i^*\big(1-2W_{i-1}^*\big)\ln\big(1-y_{(i)}\big) + \tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}w_i^*(w_i^*+1)\ln\big(1-y_{(i)}\big) \\ &= \tfrac{1}{W_n^*}\textstyle\sum_{i=1}^{n}w_i^*\big(2W_{i-1}^*+w_i^*\big)\ln\big(1-y_{(i)}\big) \end{aligned}$$

$$D = -2\textstyle\sum_{i=1}^{n}w_i^*\ln\big(1-y_{(i)}\big)$$

where w_i^* and W_i^* are defined in the section on goodness of fit measures for continuous distributions. For more information, see the topic “Continuous distributions ” on p. 353.

References

D’Agostino, R., and M. Stephens. 1986. *Goodness-of-Fit Techniques*. New York: Marcel Dekker.

Johnson, N. L., S. Kotz, and A. W. Kemp. 2005. *Univariate Discrete Distributions*, 3rd ed. Hoboken, New Jersey: John Wiley & Sons.

Kotz, S., and J. Rene Van Dorp. 2004. *Beyond Beta, Other Continuous Families of Distributions with Bounded Support and Applications*. Singapore: World Scientific Press.

Kroese, D. P., T. Taimre, and Z. I. Botev. 2011. *Handbook of Monte Carlo Methods*. Hoboken, New Jersey: John Wiley & Sons.

Marsaglia, G., and J. Marsaglia. 2004. Evaluating the Anderson-Darling Distribution. *Journal of Statistical Software*, 9:2, .

Simulation algorithms: run simulation

Running a simulation involves generating data for each of the simulated inputs, evaluating the predictive model based on the simulated data (along with values for any fixed inputs), and calculating metrics based on the model results.

Generating correlated data

Simulated values of input variables are generated so as to account for any correlations between pairs of variables. This is accomplished using the NORTA (Normal-To-Anything) method described by Biller and Ghosh (2006). The central idea is to transform standard multivariate normal variables to variables with the desired marginal distributions and Pearson correlation matrix.

Suppose that the desired variables are X_j, $j = 1, \cdots, k$, with the desired Pearson correlation matrix Σ_X, where the elements of Σ_X are given by ρ_{ij}. Then the NORTA algorithm is as follows:

1. For each pair X_i and X_j, where $i < j$, use a stochastic root finding algorithm (described in the following section) and the correlation ρ_{ij} to search for an approximate correlation ρ^*_{ij} of standard bivariate normal variables.
2. Construct the symmetric matrix Σ_z whose elements are given by ρ^*_{ij}, where $\rho^*_{ii} = 1$ and $\rho^*_{ij} = \rho^*_{ji}$.
3. Generate the standard multivariate normal variables $Z_1, \cdots, Z_k$ with Pearson correlation matrix Σ_z.
4. Transform the variables $Z_1, \cdots, Z_k$ to $X_1, \cdots, X_k$ using

$$X_i = F_i^{-1}(\Phi(Z_i)), i = 1, \cdots, k$$

where F_i is the desired marginal cumulative distribution, and Φ () is the cumulative standard normal distribution function. Then the correlation matrix of $X_1, \cdots, X_k$ will be close to the desired Pearson correlation matrix Σ_z.

Stochastic root finding algorithm

Given a correlation ρ_{ij}, a stochastic root finding algorithm is used to find an approximate correlation ρ^*_{ij} such that if standard bivariate normal variables Z_i and Z_j have the Pearson correlation ρ^*_{ij}, then after transforming Z_i and Z_j to X_i and X_j (using the transformation described in Step 4 of the previous section) the Pearson correlation between X_i and X_j is close to ρ_{ij}. The stochastic root finding algorithm is as follows:

1. Let $LowCorr = -1$ and $HighCorr = 1$
2. Simulate N samples of standard normal variables $Z_i^{(L)}$ and $Z_j^{(L)}$, $Z_i^{(H)}$ and $Z_j^{(H)}$, such that the Pearson correlation between $Z_i^{(L)}$ and $Z_j^{(L)}$ is *LowCorr* and the Pearson correlation between $Z_i^{(H)}$ and $Z_j^{(H)}$ is *HighCorr*. The sample size N is set to 1000.
3. Transform the variables $Z_i^{(L)}$, $Z_j^{(L)}$, $Z_i^{(H)}$ and $Z_j^{(H)}$ to the variables $X_i^{(L)}$, $X_j^{(L)}$, $X_i^{(H)}$ and $X_j^{(H)}$ using the transformation described in Step 4 of the previous section.

4. Compute the Pearson correlation between $X_i^{(L)}$ and $X_j^{(L)}$ and denote it as ρ_{ij}^L. Similarly, compute the Pearson correlation between $X_i^{(H)}$ and $X_j^{(H)}$ and denote it as ρ_{ij}^H.
5. If the desired correlation $\rho_{ij} \leq \rho_{ij}^L$ or $\rho_{ij} \geq \rho_{ij}^H$ then stop and set $\rho_{ij}^* = LowCorr$ if $\rho_{ij} \leq \rho_{ij}^L$ or set $\rho_{ij}^* = HighCorr$ if $\rho_{ij} \geq \rho_{ij}^H$. Otherwise go to Step 6.
6. Simulate N samples of standard bivariate normal variables $Z_i^{(M)}$ and $Z_j^{(M)}$ with a Pearson correlation of $MidCorr = \frac{1}{2}(LowCorr + HighCorr)$. As in Steps 3 and 4, transform $Z_i^{(M)}$ and $Z_j^{(M)}$ to $X_i^{(M)}$ and $X_j^{(M)}$ and compute the Pearson correlation between $X_i^{(M)}$ and $X_j^{(M)}$, which will be denoted ρ_{ij}^M.
7. If $\left|\rho_{ij} - \rho_{ij}^M\right| \leq \epsilon$ or $|HighCorr - LowCorr| \leq \epsilon$ where ε is the tolerance level (set to 0.01), then stop and set $\rho_{ij}^* = MidCorr$. Otherwise go to Step 8.
8. If $\rho_{ij} > \rho_{ij}^M$, set $LowCorr = MidCorr$, else set $HighCorr = MidCorr$ and return to Step 6.

Inverse CDF for binomial, Poisson and negative binomial distributions

Use of the NORTA method for generating correlated data requires the inverse cumulative distribution function for each desired marginal distribution. This section describes the method for computing the inverse CDF for the binomial, Poisson and negative binomial distributions. Two parameterizations of the negative binomial distribution are supported. The first parameterization describes the distribution of the number of trials before the r^{th} success, whereas the second parameterization describes the distribution of the number of failures before the r^{th} success.

The choice of method for determining the CDF depends on the mean μ of the distribution. If $\mu \geq Threshold$, where *Threshold* is set to 20, the following approximate normal method will be used to compute the inverse CDF for the binomial distribution, the Poisson distribution and the second parameterization of the negative binomial distribution.

$$X = \left[F^{-1}(\Phi(Z))\right] = \left[\sigma\left(\Phi^{-1}(\Phi(Z))\right) + \mu\right] = [\sigma Z + \mu]$$

For the first parameterization of the negative binomial distribution, the formula is as follows:

$$X = [\sigma Z + \mu] + r$$

The parameters μ and σ are given by:

- **Binomial distribution.** $\mu = NP$ and σ = $\sqrt{NP\,(1-P)}$, where N is the number of trials and P is the probability of success.
- **Poisson distribution.** μ = λ and σ = $\sqrt{\lambda}$, where λ is the rate parameter.
- **Negative binomial distribution (both parameterizations).** μ = $r\frac{1-\theta}{\theta}$ and σ = $\sqrt{r\frac{1-\theta}{\theta^2}}$, where r is the specified number of successes and θ is the probability of success.

The notation $[x]$ used above denotes the integer part of x.

If $\mu \leq Threshold$ then the bisection method will be used.

Suppose that x and z are the values of X and Z respectively, where X is a random variable with a binomial, Poisson or negative binomial distribution, and Z is a random variable with the standard normal distribution. The objective function $f(x)$ to be used in the bisection search method is as follows:

- **Binomial distribution.** $f(x) = 1 - \Pr(B(x+1, N-x) \leq P) - \Phi(z)$
- **Poisson distribution.** $f(x) = 1 - \Pr(G(x+1, 1) \leq \lambda) - \Phi(\mathrm{z})$
- **Negative binomial distribution (second parameterization).** $f(x) = \Pr(B(r, x+1) \leq \theta) - \Phi(\mathrm{z})$

where $B(\alpha, \beta)$ and $G(\alpha, \beta)$ are random variables with the beta distribution and gamma distribution, respectively, with parameters α and β.

The bisection method is as follows:

1. If $f(\mu) = 0$ then stop and set $x = [\mu + 0.5]$. Otherwise go to step 2 to determine two values x_1 and x_2 such that $f(x_1) \times f(x_2) \leq 0$.
2. If $f(\mu) > 0$ then let $x_1 = 0$ and $x_2 = \mu$. If $f(\mu) < 0$ then let $x_1 = 2^{J-1} \times \mu$ and $x_2 = 2^J \times \mu$, where J is the minimum integer such that $f(x_1) \times f(x_2) \leq 0$.
3. Let $m = \frac{1}{2}(x_1 + x_2)$. If $|f(m)| < \epsilon$ or $|x_1 - x_2| < 1$ where ϵ is a tolerance level, which is set to 10^{-6}, then stop and set $x = [m + 0.5]$. Otherwise go to Step 4.
4. If $f(m) > 0$, let $x_2 = m$, else let $x_1 = m$ and return to Step 3.

Note: The inverse CDF for the first parameterization of the negative binomial distribution is determined by taking the inverse CDF for the second parameterization and adding the distribution parameter r, where r is the specified number of successes.

Sensitivity measures

Sensitivity measures provide information on the relationship between the values of a target and the values of the simulated inputs that give rise to the target. The following sensitivity measures are supported (and rendered as Tornado charts in the output of the simulation):

- **Correlation.** Measures the Pearson correlation between a target and a simulated input.
- **One-at-a-time measure.** Measures the effect on the target of modulating a simulated input by plus or minus a specified number of standard deviations of the input.
- **Contribution to variance.** Measures the contribution to the variance of the target from a simulated input.

Notation

The following notation is used throughout this section unless otherwise stated:

Table 33-9
Notation

Notation	Description
n	Number of records of simulated data

X	An $n \times p$ matrix of values of the inputs to the predictive model. The rows $x_i = (x_{i1}, \ldots, x_{ip})$; $i = 1, \ldots, n$ contain the values of the inputs for each simulated record, excluding the target value. The columns $x_j^T = (x_{1j}, \ldots, x_{nj})$; $j = 1, \ldots, p$ represent the set of inputs.
y	An $n \times 1$ vector of values of the target variable, consisting of $y_i, i = 1, \ldots, n$
$F(X)$	A known model which can generate y from X
sa_j	The value of a sensitivity measure for the input x_j

Correlation measure

The correlation measure is the Pearson correlation coefficient between the values of a target and one of its simulated predictors. The correlation measure is not supported for targets with a nominal measurement level or for simulated inputs with a categorical distribution.

One-at-a-time measure

The one-at-a-time measure is the change in the target due to modulating a simulated input by plus or minus a specified number of standard deviations of the distribution associated with the input. The one-at-a-time measure is not supported for targets with an ordinal or nominal measurement level, or for simulated inputs with any of the following distributions: categorical, Bernoulli, binomial, Poisson, or negative binomial.

The procedure is to modulate the values of a simulated input by the specified number of standard deviations and recompute the target with the modulated values, without changing the values of the other inputs. The mean change in the target is then taken to be the value of the one-at-a-time sensitivity measure for that input.

For each simulated input x_j for which the one-at-a-time measure is supported:

1. Define the temporary data matrix $X^{'} = X$
2. Add the specified number of standard deviations of the input's distribution to each value of x_j in $X^{'}$.
3. Calculate $y^{'} = \mathrm{F}(X')$
4. Calculate $sa_j = \frac{1}{n}\sum_{i=1}^{n}\left(y^{'}{}_{i} - y_i\right)$
5. Repeat Step 2, but now subtracting the specified number of standard deviations from each value of x_j. Continue with Steps 3 and 4 to obtain the value of sa_j in this case.

Contribution to variance measure

The contribution to variance measure uses the method of Sobol (2001) to calculate the total contribution to the variance of a target due to a simulated input. The total contribution to variance, as defined by Sobol, automatically includes interaction effects between the input of interest and the other inputs in the predictive model.

The contribution to variance measure is not supported for targets with an ordinal or nominal measurement level, or for simulated inputs with any of the following distributions: categorical, Bernoulli, binomial, Poisson, or negative binomial.

Let X' be an additional set of simulated data, in the same form as X and with the same number of simulated records.

Define the following:

$$f_0 = \frac{1}{n} \sum_{i=1}^{n} y_i$$

$$D = \frac{1}{n} \sum_{i=1}^{n} y_i^2 - (f_0)^2$$

For each simulated input x_j for which the contribution to variance measure is supported, calculate

$$D_{x \setminus x_j} = \frac{1}{n} \sum_{i=1}^{n} y_i \big[F\left(X'_{x_j} + X_{x \setminus x_j} \right) \big]_i - (f_0)^2$$

where:

- $x \setminus x_j$ denotes the set of all inputs excluding x_j
- $X'_{x_j} + X_{x \setminus x_j}$ is a derived data matrix where the column associated with x_j is taken from X' and the remaining columns (for all inputs excluding x_j) are taken from X

The total contribution to variance from x_j is then given by

$$sa_j = \frac{D - D_{x \setminus x_j}}{D}$$

Note: When interaction terms are present, the sum of the sa_j over all simulated inputs for which the contribution of variance is supported, may be greater than 1.

References

Biller, B., and S. Ghosh. 2006. Multivariate input processes. In: *Handbooks in Operations Research and Management Science: Simulation,* B. L. Nelson, and S. G. Henderson, eds. Amsterdam: Elsevier Science, 123–153.

Sobol, I. M. 2001. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55, 271–280.

Support Vector Machine (SVM) Algorithms

Introduction to Support Vector Machine Algorithms

The Support Vector Machine (SVM) is a supervised learning method that generates input-output mapping functions from a set of labeled training data. The mapping function can be either a classification function or a regression function. For classification, nonlinear kernel functions are often used to transformed input data to a high-dimensional feature space in which the input data become more separable compared to the original input space. Maximum-margin hyperplanes are then created. The produced model depends on only a subset of the training data near the class boundaries.

Similarly, the model produced by Support Vector Regression ignores any training data that is sufficiently close to the model prediction. (Support Vectors can appear only on the error tube boundary or outside the tube.)

SVM Algorithm Notation

x_i	The *i*th training sample
y_i	The class label for the *i*th training sample
l	The number of training samples
$K\left(x_i \cdot x_j\right)$	The kernel function value for the pair of samples i, j
$Q\left(x_i \cdot x_j\right) = y_i y_j K\left(x_i \cdot x_j\right)$	The kernel matrix element at row i and column j
α_i	Coefficients for training samples (zero for non-support vectors)
α_i^*	Coefficients for training samples for support vector regression models
$f\left(x\right)$	Decision function
m	The number of classes of the training samples
C	The upper bound of all variables
e	The vector with all elements equal to 1
$sgn\left(x\right)$	The sign function: $\begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$

SVM Types

This section describes the types of SVM available, based on the descriptions in the LIBSVM technical report(Chang and Lin, 2003). $K\left(x_i \cdot x_j\right)$ is the kernel function selected by the user. For more information, see the topic “SMO Algorithm” on p. 371.

C-Support Vector Classification (C-SVC)

Given training vectors $x_i \in R^l$, i = 1, ..., l, in two classes, and a vector $y \in R^l$ such that $y_i \in \{-1, 1\}$, C-SVC solves the following dual problem:

$$\min f(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

such that $0 \leq \alpha_i \leq C, i = 1, ..., l$ and $y^T\alpha = 0$, where

$$\alpha = (\alpha_1, \alpha_2, ..., \alpha_l)^T$$

and Q is an $l \times l$ matrix, $Q(x_i \cdot x_j) = y_i y_j K(x_i \cdot x_j)$

The decision function is

$$sgn\left(\sum_{i=1}^{l} y_i\alpha_i K(x_i, x) + b\right)$$

where b is a constant term.

ε-Support Vector Regression (ε-SVR)

In regression models, we estimate the functional dependence of the dependent (target) variable $y \in R$ on an n-dimensional input vector **x**. Thus, unlike classification problems, we deal with real-valued functions and model an $R^n \to R^1$ mapping. Given a set of data $\{(x_1, z_1), ..., (x_l, z_l)\}$, such that $x_I \in R^n$ is an input and $z_i \in R^1$ is a target output, the dual form of ε-Support Vector Regression is

$$\min f(\alpha, \alpha^*) = \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + z_i\sum_{i=1}^{l}(\alpha_i - \alpha_i^*)$$

such that $0 \leq \alpha_i$ and $\alpha_i^* \leq C$ for $i = 1, ..., l$, and

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0$$

where $\alpha = (\alpha_1, \alpha_2, ..., \alpha_l)^T$, $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_l^*)^T$, and Q is an $l \times l$ matrix, $Q_{ij} = K(x_i \cdot x_j)$

The approximate function is

$$\sum_{i=1}^{l}(-\alpha_i + \alpha_i^*) K(x_i, x) + b$$

where b is a constant term.

Primary Calculations

The primary calculations for building SVM models are described below.

Solving Quadratic Problems

In order to find the decision function or the approximate function, the quadratic problem must be solved. After the solution is obtained, we can get different coefficients α_i:

- if $0 < \alpha_i < C$, the corresponding training sample is a **free support vector**.
- if $\alpha_i = C$, the corresponding training sample is a **boundary support vector**.
- if $\alpha_i = 0$, the corresponding training sample is a **non-support vector**, which doesn't affect the classification or regression result.

Free support vectors and boundary support vectors are called **support vectors**.

This document adapts the decomposition method to solve the quadratic problem using second order information (Fan, Chen, and Lin, 2005). In order to solve all the SVM's in a unified framework, we'll introduce a general form for C-SVC and ε-SVR.

For ε-SVR, we can rewrite the dual form as

$$\min \quad f(\alpha, \alpha^*) = \frac{1}{2}\left(\alpha^T (\alpha^*)^T\right)\begin{pmatrix} Q & -Q \\ -Q & Q \end{pmatrix}\begin{pmatrix} \alpha \\ \alpha^* \end{pmatrix} + \begin{pmatrix} \varepsilon e^T + z^T \\ \varepsilon e^T - z^T \end{pmatrix}^T \begin{pmatrix} \alpha \\ \alpha^* \end{pmatrix}$$

such that $y^T \begin{pmatrix} \alpha \\ \alpha^* \end{pmatrix} = 0$ and $0 \le \alpha_i, \alpha_i^* \le C$ for *i* = 1, ... , *l*, where y is a 2*l* × 1 vector with $y_i = 1$ for *i* = 1, ..., *l* and $y_i = -1$ for *i* = *l* + 1, ... , 2*l*.

Given this, the general form is

$$\min f(\alpha) = \frac{1}{2}\alpha^T Q \alpha + p^T \alpha$$

such that $0 \le \alpha_i \le C$ for *i* = 1, ... , *l*, and $y^T \alpha = constant$

	α in $W(\alpha)$	**p^T**	**$y^T\alpha = constant$**
C-SVC	$(\alpha_1, \alpha_2, ..., \alpha_l)^T$	$-e^T$	$y = (y_1, ..., y_l)^T$ $y^T\alpha = 0$
ε-SVR	$(\alpha_1, \alpha_2, ..., \alpha_l, \alpha_1^*, \alpha_2^*, ..., \alpha_l^*)^T$	$\begin{pmatrix} \varepsilon e^T + z^T \\ \varepsilon e^T - z^T \end{pmatrix}^T$	$y = (1_1, ..., 1_l, -1_{l+1}, ..., -1_{2l})^T$ $y^T\alpha = 0$

The Constant in the Decision Function

After the quadratic programming problem is solved, we get the support vector coefficients in the decision function. We need to compute the constant term in the decision function as well. We introduce two accessory variables r_1 and r_2:

- For y_i = 1:

  If $0 < \alpha_i < C$,

  $$r_1 = \frac{\Sigma_{0<\alpha_i<C, y_i=1} \nabla f\left(\alpha_i\right)}{\Sigma_{0<\alpha_i<C, y_i=1} 1}$$

  Otherwise,

  $$r_1 = \frac{\max_{\alpha_i=C, y_i=1} \nabla f\left(\alpha_i\right) + \min_{\alpha_i=0, y_i=1} \nabla f\left(\alpha_i\right)}{2}$$

- For y_i = −1:

  If $0 < \alpha_i < C$,

  $$r_2 = \frac{\Sigma_{0<\alpha_i<C, y_i=-1} \nabla f\left(\alpha_i\right)}{\Sigma_{0<\alpha_i<C, y_i=-1} 1}$$

  Otherwise,

  $$r_2 = \frac{\max_{\alpha_i=C, y_i=-1} \nabla f\left(\alpha_i\right) + \min_{\alpha_i=0, y_i=-1} \nabla f\left(\alpha_i\right)}{2}$$

  After r_1 and r_2 are obtained, calculate $b = \frac{r_1 + r_2}{2}$

Variable Scale

For continuous input variables, linearly scale each attribute to [-1, 1] or [0, 1]:

$$V^{new} = \frac{V - V_{\min}}{V_{\max} - V_{\min}} (newmax - newmin) + newmin$$

For categorical input fields, if there are *m* categories, then use (0, 1, 2, ..., *m*) to represent the categories and scale the values as for continuous input variables.

Model Building Algorithm

In this section, we provide a fast algorithm to train the SVM. A modified sequential minimal optimization (SMO) algorithm is provided for C-SVC binary and ε-SVR models. A fast SVM training algorithm based on divide-and-conquer is used for all SVMs.

SMO Algorithm

Due to the density of the kernel matrix, traditional optimization methods cannot be directly applied to solve for the vector α. Unlike most optimization methods which update the whole vector α in each step of an iterative process, the decomposition method modifies a subset of α per iteration. This subset, denoted as the working set *B*, leads to a small sub-problem to be minimized in each iteration. Sequential minimal optimization (SMO) is an extreme example of this approach which restricts *B* to have only two elements. In each iteration no optimization algorithm is needed to solve a simple two-variable problem. The key step of SML is the working set selection method, which determines the speed of convergence for the algorithm.

Kernel functions

The algorithm supports four kernel functions:

Linear function	$K\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) = \mathbf{x}_i^T \cdot \mathbf{x}_j$
Polynomial function	$K\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) = \left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)^d$
RBF function	$K\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) = \exp\left(-\frac{\lVert\mathbf{x}_i - \mathbf{x}_j\rVert^2}{2\sigma^2}\right)$ $= \exp\left(-\gamma\lVert\mathbf{x}_i - \mathbf{x}_j\rVert^2\right), \gamma = \frac{1}{2\sigma^2}$
Hyperbolic tangent function	$K\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) = \tanh\left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)$ $\tanh\left(x\right) = \frac{e^x - e^{(-x)}}{e^x + e^{(-x)}}$

Base Working Set Selection Algorithm

The base selection algorithm derives the selection set $B = \{i, j\}$ based on τ, *C*, the target vector **y**, and the selected kernel function $K(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j})$.

Let

$$a_{ij} = K_{ii} + K_{jj} - 2K_{ij}, b_{ij} = -y_i \nabla f\left(\alpha^k\right)_i + y_j \nabla f\left(\alpha^k\right)_j$$

and

$$\overline{a}_{ts} = \begin{cases} a_{ts} & \text{if } a_{ts} > 0 \\ \tau & \text{otherwise} \end{cases}$$

where τ is a small positive number.

Select

$$\begin{aligned} & i \in \arg\max\nolimits_t \left\{-y_t \nabla f\left(\alpha^k\right)_t | t \in I_{up}\left(\alpha^k\right)\right\}, \\ & j \in \arg\min\nolimits_t \left\{-\frac{b_{it}^2}{\overline{a}_{it}} | t \in I_{low}\left(\alpha^k\right), -y_t \nabla f\left(\alpha^k\right)_t < -y_i \nabla f\left(\alpha^k\right)_i\right\} \end{aligned}$$

where

$$I_{up}(\alpha) = \{t|\alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}$$

$$I_{low}(\alpha) = \{t|\alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}$$

Return $B = \{i, j\}$, where $\nabla f(\alpha) = Q\alpha + \mathbf{p}$.

Shrink Algorithm

In order to speed up the convergence of the algorithm near the end of the iterative process, the decomposition method identifies a possible set A containing all final free support vectors. Hence, instead of solving the whole problem, the decomposition method works on a smaller problem:

$$\min_{\alpha_A} \frac{1}{2}\alpha_A^T Q_{AA}\alpha_A - (\mathbf{p}_A - Q_{AN}\alpha_N)^T \alpha_A$$

$$\text{s. t. } 0 \le (\alpha_A)_t \le C, t = 1, \cdots, q$$

$$\mathbf{y}_A^T \alpha_A = const - \mathbf{y}_N^T \alpha_N$$

where $N = \{1, 2, \cdots l\} \setminus A$ is the set of shrunken variables.

Afer every min(l, 1000) iterations, we try to shrink some variables. During the iterative process $m(\alpha^k) > M(\alpha^k)$. Until $m(\alpha^k) - M(\alpha^k) \le \epsilon$ is satisfied, we can shrink variables in the following set:

$$\left\{t| -y_t \nabla f(\alpha_t) > m\left(\alpha^k\right), \alpha_t = C, y_t = 1 \text{ or } \alpha_t = 0, y_t = -1\right\} \cup$$
$$\left\{t| -y_t \nabla f(\alpha_t) < M\left(\alpha^k\right), \alpha_t = 0, y_t = 1 \text{ or } \alpha_t = C, y_t = -1\right\}$$

Thus the set A of activated variables is dynamically reduced every min(l, 1000) iterations.

▶ To account for the tendency of the shrinking method to be too aggressive, we reconstruct the gradient when the tolerance reaches

$$m\left(\alpha^k\right) \le M\left(\alpha^k\right) + 10\epsilon$$

After reconstructing the gradient, we restore some of the previously shrunk variables based on the relationship

$$\left\{t| -y_t \nabla f(\alpha_t) \le m\left(\alpha^k\right), \alpha_t = C, y_t = 1 \text{ or } \alpha_t = 0, y_t = -1\right\} \cup$$
$$\left\{t| -y_t \nabla f(\alpha_t) \ge M\left(\alpha^k\right), \alpha_t = 0, y_t = 1 \text{ or } \alpha_t = C, y_t = -1\right\}$$

Gradient Reconstruction

To decrease the cost of reconstruction of the gradient $\nabla f(\alpha)$, during the iterations we always keep

$$\overline{G}_i = C \sum_{\alpha_j = C} Q_{ij} i = 1, \cdots l$$

Then for the gradient $\nabla f\left(\alpha_i\right),\ i \notin A$, we have

$$\nabla f\left(\alpha_i\right) = \sum_{j=1}^{l} Q_{ij}\alpha_j + p_i = \overline{G}_i + \sum_{0<\alpha_j<C} Q_{ij}\alpha_j + p_i$$

and for the gradient $\nabla f\left(\alpha_i\right),\ i \in A$ we have

$$\nabla f\left(\alpha_i^{k+1}\right) = \nabla f\left(\alpha_i^{k}\right) + Q_{it}\Delta\alpha_t + Q_{is}\Delta\alpha_s$$

where *t* and *s* are the working set indices.

Unbalanced Data Strategy

For some classification problems, the algorithm uses different parameters in the SVM formulation. The differences only affect the procedure for updating α^k. Different conditions are handled as follows:

For :

Conditions		Update parameters
$y_i \neq y_j$	$\alpha_i - \alpha_j > C_i - C_j$ and $\alpha_i > C_i$	$\alpha_i^{new} = C_i$ $\alpha_j^{new} = C_i - (\alpha_i - \alpha_j)$
	$\alpha_i - \alpha_j \leq C_i - C_j$ and $\alpha_j > C_j$	$\alpha_j^{new} = C_j$ $\alpha_i^{new} = C_j + (\alpha_i - \alpha_j)$
	$\alpha_i - \alpha_j > 0$ and $\alpha_j < 0$	$\alpha_j^{new} = 0$ $\alpha_i^{new} = (\alpha_i - \alpha_j)$
	$\alpha_i - \alpha_j \leq 0$ and $\alpha_i < 0$	$\alpha_i^{new} = 0$ $\alpha_j^{new} = -(\alpha_i - \alpha_j)$
$y_i = y_j$	$\alpha_i + \alpha_j > C_i$ and $\alpha_i > C_i$	$\alpha_i^{new} = C_i$ $\alpha_j^{new} = (\alpha_i + \alpha_j) - C_i$
	$\alpha_i + \alpha_j \leq C_i$ and $\alpha_j < 0$	$\alpha_j^{new} = 0$ $\alpha_i^{new} = (\alpha_i + \alpha_j)$
	$\alpha_i + \alpha_j > C_j$ and $\alpha_j > C_j$	$\alpha_i^{new} = (\alpha_i + \alpha_j) - C_j$ $\alpha_j^{new} = C_j$
	$\alpha_i + \alpha_j \leq C_j$ and $\alpha_i < 0$	$\alpha_i^{new} = 0$ $\alpha_j^{new} = (\alpha_i + \alpha_j)$

SMO Decomposition

The following steps are used in the SMO decomposition:

1. Find α^1 as the initial feasible solution, and set *k* = 1.
2. If α^k is a stationary solution, stop.

   A feasible solution is stationary if $m(\alpha) - M(\alpha) \leq \epsilon$, where

   $$m(\alpha) = \max_{i \in I_{up}} \{-y_i \nabla f(\alpha_i)\}$$

   $$M(\alpha) = \min_{i \in I_{low}} \{-y_i \nabla f(\alpha_i)\}$$

   $$I_{up}(\alpha) = \{t | \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}$$

   $$I_{low}(\alpha) = \{t | \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}$$

   Find a two-element working set $B = \{i, j\}$ using the working set selection algorithm. (For more information, see the topic “Base Working Set Selection Algorithm” on p. 371.)
3. If the shrink algorithm is being used to speed up convergence, apply the algorithm here. (For more information, see the topic “Shrink Algorithm” on p. 372.)
4. Derive α^{k+1} as follows:

- If $C_i \neq C_j$, or if solving a classification problem, use the unbalanced data strategy. (For more information, see the topic “Unbalanced Data Strategy” on p. 373.)
- If $a_{ij} > 0$, solve the subproblem

  $$\min_{\alpha_B} \frac{1}{2} \begin{bmatrix} \alpha_i & \alpha_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ji} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \left(-\mathbf{p}_B + Q_{BN}\alpha_N^k\right)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \text{cont}$$

  Subject to the constraints

  $$0 \leq \alpha_i, \alpha_j \leq C,$$
  $$y_i\alpha_i + y_j\alpha_j = -\mathbf{y}_N^T \alpha_N^k$$

  and let

  $$\alpha_i^{new} = \alpha_i^{old} + \tfrac{y_i b_{ij}}{a_{ij}}$$
  $$\alpha_j^{new} = \alpha_j^{old} - \tfrac{y_j b_{ij}}{a_{ij}}$$

- Otherwise, solve the subproblem

$$\min_{\alpha_B} \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij}\\ Q_{ji} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} + \left(-\mathbf{p}_B + Q_{BN}\alpha_N^k\right)^T\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} +$$
$$\frac{\tau - a_{ij}}{4}\left(\left(\alpha_i - \alpha_i^k\right)^2 + \left(\alpha_j - \alpha_j^k\right)^2\right)$$

subject to the same constraints described above, where τ is a small positive number and $N = \{1, 2, \cdots l\} \setminus B$, and let

$$\alpha_i^{new} = \alpha_i^{old} + \frac{y_i b_{ij}}{\tau}$$
$$\alpha_j^{new} = \alpha_j^{old} - \frac{y_j b_{ij}}{\tau}$$

Finally, set α_B^{k+1} to be the optimal point of the subproblem.

Set $\alpha_N^{k+1} = \alpha_N^k$, set $k = k + 1$, and go to step 2.

Fast SVM Training

For binary SVM models, the dense kernel matrix cannot be stored in memory when the number of training samples *l* is large. Rather than using the standard decomposition algorithm which depends on a cache strategy to compute the kernel matrix, a divide-and-conquer approach is used, dividing the original problem into a set of small subproblems that can be solved by the SMO algorithm (Dong, Suen, and Krzyzak, 2005). For each subproblem, the kernel matrix can be stored in a kernel cache defined as part of contiguous memory. The size of the kernel matrix should be large enough to hold all the support vectors in the whole training set and small enough to satisfy the memory constraint. Since the kernel matrix for the subproblem is completely cached, each element of the kernel matrix needs to be evaluated only once and must be calculated using a fast method.

There are two steps in the fast SVM training algorithm:

- Parallel optimization
- Fast sequential optimization

These steps are described in more detail below.

Parallel Optimization

Since the kernel matrix **Q** is symmetric and semi-positive definite, its block diagonal matrices are semi-positive definite, and can be written as

$$Q_{diag} = \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_k \end{bmatrix}$$

where $l_i \times l_i$ matrices $Q_i, i = 1, \cdots, k, \sum_{i}^{k} l_i = l$ are block diagonal. Then we obtain k optimization subproblems as described in "Base Working Set Selection Algorithm" on p. 371. All the subproblems are optimized using the SMO decomposition algorithm in parallel. After this parallel optimization, most non-support vectors will be removed from the training set. Then a new training set can be obtained by collecting support vectors from the sub-problems. Although the size of the new training set is much smaller than that of the original one, the memory may not be large enough to store the kernel matrix, especially when dealing with a large dataset. Therefore a fast sequential optimization technique is used.

Fast Sequential Optimization

The technique for fast sequential optimization works by iteratively optimizing subsets of the problem. Initially, the training set is shuffled, all $\alpha_i, i = 1, \cdots, l$ are set to zero, and a subset Sub $\subseteq S$ is selected from the training set *S*. The size of the subset *d* is set ($d \leq l$).

Optimization proceeds as follows:

- ▶ Apply the SMO algorithm to optimize a subproblem in Sub with kernel caching, and update α_i and the kernel matrix. For more information, see the topic "SMO Algorithm" on p. 371.
- ▶ Select a new subset using the queue subset method. The size of the subset is chosen to be large enough to contain all support vectors in the training set but small enough to satisfy the memory constraint. For more information, see the topic "Queue Method for Subset Selection" on p. 376.
- ▶ Return to step 1 unless any of the following stopping conditions is true:
  - ■ $|\Delta SV| < 20$ and (Number of learned samples) > *l*
  - ■ $SV \geq (d - 1)$
  - ■ (Number of learned samples) $> T \cdot l$

where $|\Delta SV|$ is the change in number of support vectors between two successive subsets, *l* is the size of the new training set, and *T* (> 1.0) is a user-defined maximum number of loops through the data allowed.

Queue Method for Subset Selection

The **queue method** selects subsets of the training set that can be trained by fast sequential optimization. For more information, see the topic "Fast Sequential Optimization" on p. 376..

The method is initialized by setting the subset to contain the first *d* records in the training data and the queue $\mathbf{Q}_\mathrm{S}$ to contain all the remaining records, and computing the kernel matrix for the subset.

Once initialized, subset selection proceeds as follows: each non-support vector in the subset is added to the end of the queue, and replaced in the subset with the record at the front of the queue (which is consequently removed from the queue). When all non-support vectors have been replaced, the subset is returned for optimization. On the next iteration, the same process is applied, starting with the subset and the queue in the same state they were in at the end of the last iteration.

Blank Handling

All records with missing values for any input or output field are excluded from the estimation of the model.

Model Nugget/Scoring

The SVM Model Nugget generates predictions and predicted probabilities for output classes. Predictions are based on the category with the highest predicted probability for each record.

To choose a predicted value, posterior probabilities are approximated using a sigmoid function(Platt, 2000). The approximation used is

$$P\left(y=1|x\right) \approx P_{A,B}\left(x\right) = \frac{1}{1+\exp\left(Af\left(x\right)+B\right)}.$$

The optimal parameters A and B are the estimated by solving the following regularized maximum likelihood problem with a set of labeled examples $\left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \cdots, \left(\mathbf{x}_l, y_l\right), \mathbf{x} \in \mathbf{R}^n, y \in \{+1, -1\}$, and N_+ is the number of positive examples and N_- is the number of negative examples:

$$\min_{z=(A,B)} F\left(z\right) = -\sum_{i=1}^{l}\left(t_i \log\left(p_i\right) + \left(1-t_i\right)\log\left(1-p_i\right)\right)$$

$$p_i = P_{A,B}\left(x_i\right) \text{ and } t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } y_i = +1 \\ \frac{1}{N_- + 2} & \text{if } y_i = -1 \end{cases}, i = 1, \cdots, l$$

$$\nabla F\left(z\right) = \begin{bmatrix} \sum_{i=1}^{l} f_i\left(t_i - p_i\right) \\ \sum_{i=1}^{l}\left(t_i - p_i\right) \end{bmatrix}$$

$$H\left(z\right) = \nabla^2 F\left(z\right) = \begin{bmatrix} \sum_{i=1}^{l} f_i^2 p_i\left(1-p_i\right) & \sum_{i=1}^{l} f_i p_i\left(1-p_i\right) \\ \sum_{i=1}^{l} f_i p_i\left(1-p_i\right) & \sum_{i=1}^{l} p_i\left(1-p_i\right) \end{bmatrix}$$

Blank Handling

Records with missing values for any input field cannot be scored and are assigned a predicted value and probability value(s) of $null$.

Time Series Algorithms

The Time Series node builds univariate exponential smoothing, ARIMA (Autoregressive Integrated Moving Average), and transfer function (TF) models for time series, and produces forecasts. The procedure includes an Expert Modeler that identifies and estimates an appropriate model for each dependent variable series. Alternatively, you can specify a custom model.

This algorithm is designed with help from professor Ruey Tsay at The University of Chicago.

Notation

The following notation is used throughout this chapter unless otherwise stated:

Y_t (t=1, 2, ..., n)	Univariate time series under investigation.
n	Total number of observations.
$\hat{Y}_t\,(k)$	Model-estimated k-step ahead forecast at time t for series Y.
S	The seasonal length.

Models

The Time Series node estimates exponential smoothing models and ARIMA/TF models.

Exponential Smoothing Models

The following notation is specific to exponential smoothing models:

α	Level smoothing weight
γ	Trend smoothing weight
ϕ	Damped trend smoothing weight
δ	Season smoothing weight

Simple Exponential Smoothing

Simple exponential smoothing has a single level parameter and can be described by the following equations:

$$L(t) = \begin{cases} \alpha Y\,(t) + (1-\alpha)L(t-1), & if \;\; Y\,(t) \;\; is \;\; not \;\; missing \\ L(t-1), & else \end{cases}$$

$$\hat{Y}_t\,(k) = L\,(t)$$

It is functionally equivalent to an ARIMA(0,1,1) process.

Brown's Exponential Smoothing

Brown's exponential smoothing has level and trend parameters and can be described by the following equations:

$$L(t) = \begin{cases} \alpha Y(t) + (1-\alpha)L(t-1), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ L(t-1) + T(t-1), & else \end{cases}$$

$$T(t) = \begin{cases} \alpha(L(t) - L(t-1)) + (1-\alpha)T(t-1), & if \;\; Y(t) \, is \;\; not \;\; missing \\ T(t-1), & else \end{cases}$$

$$\hat{Y}_t(k) = L(t) + \left((k-1) + \alpha^{-1}\right)T(t)$$

It is functionally equivalent to an ARIMA(0,2,2) with restriction among MA parameters.

Holt's Exponential Smoothing

Holt's exponential smoothing has level and trend parameters and can be described by the following equations:

$$L(t) = \begin{cases} \alpha Y(t) + (1-\alpha)(L(t-1) + T(t-1)), & if \;\; Y(t) \, is \;\; not \;\; missing \\ L(t-1) + T(t-1), & else \end{cases}$$

$$T(t) = \begin{cases} \gamma(L(t) - L(t-1)) + (1-\gamma)T(t-1), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ T(t-1), & else \end{cases}$$

$$\hat{Y}_t(k) = L(t) + kT(t)$$

It is functionally equivalent to an ARIMA(0,2,2).

Damped-Trend Exponential Smoothing

Damped-Trend exponential smoothing has level and damped trend parameters and can be described by the following equations:

$$L(t) = \begin{cases} \alpha Y(t) + (1-\alpha)(L(t-1) + \varphi T(t-1)), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ L(t-1) + \varphi T(t-1), & else \end{cases}$$

$$T(t) = \begin{cases} \gamma(L(t) - L(t-1)) + (1-\gamma)\varphi T(t-1), & if \;\; Y(t) \, is \;\; not \;\; missing \\ \varphi T(t-1), & else \end{cases}$$

$$\hat{Y}_t(k) = L(t) + \sum_{i=1}^{k} \phi^i T(t)$$

It is functionally equivalent to an ARIMA(1,1,2).

Simple Seasonal Exponential Smoothing

Simple seasonal exponential smoothing has level and season parameters and can be described by the following equations:

$$L(t) = \begin{cases} \alpha(Y(t) - S(t-s)) + (1-\alpha)L(t-1), & if \;\; Y(t)\, is \;\; not \;\; missing \\ L(t-1), & else \end{cases}$$

$$S(t) = \begin{cases} \delta(Y(t) - L(t)) + (1-\delta)S(t-s), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ S(t-s) & else \end{cases}$$

$$\hat{Y}_t(k) = L(t) + S(t+k-s)$$

It is functionally equivalent to an ARIMA(0,1,(1,s,s+1))(0,1,0) with restrictions among MA parameters.

Winters' Additive Exponential Smoothing

Winters' additive exponential smoothing has level, trend, and season parameters and can be described by the following equations:

$$L(t) = \begin{cases} \alpha(Y(t) - S(t-s)) + (1-\alpha)(L(t-1) + T(t-1)), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ L(t-1) + T(t-1), & else \end{cases}$$

$$T(t) = \begin{cases} \gamma(L(t) - L(t-1)) + (1-\gamma)T(t-1), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ T(t-1), & else \end{cases}$$

$$S(t) = \begin{cases} \delta(Y(t) - L(t)) + (1-\delta)S(t-s), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ S(t-s) & else \end{cases}$$

$$\hat{Y}_t(k) = L(t) + kT(t) + S(t+k-s)$$

It is functionally equivalent to an ARIMA(0,1,s+1)(0,1,0) with restrictions among MA parameters.

Winters' Multiplicative Exponential Smoothing

Winters' multiplicative exponential smoothing has level, trend and season parameters and can be described by the following equations:

$$L(t) = \begin{cases} \alpha(Y(t)/S(t-s)) + (1-\alpha)(L(t-1) + T(t-1)), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ L(t-1) + T(t-1), & else \end{cases}$$

$$T(t) = \begin{cases} \gamma(L(t) - L(t-1)) + (1-\gamma)T(t-1), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ T(t-1), & else \end{cases}$$

$$S(t) = \begin{cases} \delta(Y(t)/L(t)) + (1-\delta)S(t-s), & if \;\; Y(t) \;\; is \;\; not \;\; missing \\ S(t-s) & else \end{cases}$$

$$\hat{Y}_t(k) = (L(t) + kT(t))S(t+k-s)$$

There is no equivalent ARIMA model.

Estimation and Forecasting of Exponential Smoothing

The sum of squares of the one-step ahead prediction error, $\Sigma\left(Y_t - \hat{Y}_{t-1}(1)\right)^2$, is minimized to optimize the smoothing weights.

Initialization of Exponential Smoothing

Let *L* denote the level, *T* the trend and, *S*, a vector of length s, denote the seasonal states. The initial smoothing states are made by back-casting from t=n to t=0. Initialization for back-casting is described here.

For all the models $L = y_n$.

For all non-seasonal models with trend, *T* is the negative of the slope of the line (with intercept) fitted to the data with time as a regressor.

For the simple seasonal model, the elements of *S* are seasonal averages minus the sample mean; for example, for monthly data the element corresponding to January will be average of all January values in the sample minus the sample mean.

For the additive Winters' model, fit $y = \alpha t + \sum_{i=1}^{s} \beta_i I_i(t)$ to the data where *t* is time and $I_i(t)$ are seasonal dummies. Note that the model does not have an intercept. Then $T = -\alpha$, and $S = \beta - mean(\beta)$.

For the multiplicative Winters' model, fit a separate line (with intercept) for each season with time as a regressor. Suppose μ is the vector of intercepts and β is the vector of slopes (these vectors will be of length s). Then $T = -mean(\beta)$ and $S = (\mu + \beta) / (mean(\mu) + mean(\beta))$.

The initial smoothing states are:

$L' = L(0)$

$T' = -T(0)$

$S^{'} = (S(1-s), S(2-s), \ldots S(-1), S(0))= (S(1), S(2), \ldots, S(-1+s), S(0))$

ARIMA and Transfer Function Models

The following notation is specific to ARIMA/TF models:

a_{t} $(t = 1, 2, \ldots, n)$	White noise series normally distributed with mean zero and variance σ^2
p	Order of the non-seasonal autoregressive part of the model
q	Order of the non-seasonal moving average part of the model
d	Order of the non-seasonal differencing
P	Order of the seasonal autoregressive part of the model
Q	Order of the seasonal moving-average part of the model
D	Order of the seasonal differencing

s	Seasonality or period of the model
$\phi_p\left(B\right)$	AR polynomial of B of order p, $\phi_p\left(B\right) = 1 - \varphi_1 B - \varphi_2 B^2 - ... - \varphi_p B^p$
$\theta_q\left(B\right)$	MA polynomial of B of order q, $\theta_q\left(B\right) = 1 - \vartheta_1 B - \vartheta_2 B^2 - ... - \vartheta_q B^q$
$\Phi_P\left(B^s\right)$	Seasonal AR polynomial of BS of order P, $\Phi_P\left(B^s\right) = 1 - \Phi_1 B^s - \Phi_2 B^{s2} - ... - \Phi_P B^{sP}$
$\Theta_Q\left(B^s\right)$	Seasonal MA polynomial of BS of order Q, $\Theta_Q\left(B^s\right) = 1 - \Theta_1 B^s - \Theta_2 B^{s2} - ... - \Theta_Q B^{sQ}$
Δ	Differencing operator $\Delta = (1-B)^d(1-B^s)^D$
B	Backward shift operator with $BY_t = Y_{t-1}$ and $Ba_t = a_{t-1}$
$Z\sigma_t^2$	Prediction variance of Z_t
$N\sigma_t^2$	Prediction variance of the noise forecasts

Transfer function (TF) models form a very large class of models, which include univariate ARIMA models as a special case. Suppose Y_t is the dependent series and, optionally, $X_{1t}, X_{2t}, ..., X_{kt}$ are to be used as predictor series in this model. A TF model describing the relationship between the dependent and predictor series has the following form:

$$Z_t = f\left(Y_t\right),$$

$$\Delta Z_t = \mu + \sum_{i=1}^{k} \frac{Num_i}{Den_i} \Delta_i B^{b_i} f_i(X_{it}) + \frac{MA}{AR} a_t.$$

The univariate ARIMA model simply drops the predictors from the TF model; thus, it has the following form:

$$\Delta Z_t = \mu + \tfrac{MA}{AR} a_t$$

The main features of this model are:

- An initial transformation of the dependent and predictor series, *f* and f_{i}. This transformation is optional and is applicable only when the dependent series values are positive. Allowed transformations are log and square root. These transformations are sometimes called variance-stabilizing transformations.
- A constant term μ.
- The unobserved i.i.d., zero mean, Gaussian error process a_t with variance σ^2.
- The moving average lag polynomial *MA*=$\theta_q\left(B\right)\Theta_Q\left(B^s\right)$ and the auto-regressive lag polynomial *AR*=$\phi_p\left(B\right)\Phi_P\left(B^s\right)$.
- The difference/lag operators Δ and Δ_i.
- A delay term, B^{b_i}, where b_i is the order of the delay
- Predictors are assumed given. Their numerator and denominator lag polynomials are of the form: Num_i=$(\omega_{i0} - \omega_{i1}B - \cdots - \omega_{iu}B^u)(1 - \Omega_{i1}B^s - \cdots - \Omega_{iv}B^{vs})B^b$ and Den_i=$(1 - \delta_{i1}B - \cdots - \delta_{ir}B^r)(1 - \Delta_{i1}B^s - \cdots)$.
- The "noise" series

$$N_t = \Delta Z_t - \mu - \sum_{i=1}^{k} \frac{Num_i}{Den_i} \Delta_i B^{b_i} X_{it}$$

is assumed to be a mean zero, stationary ARMA process.

Interventions and Events

Interventions and events are handled like any other predictor; typically they are coded as 0/1 variables, but note that a given intervention variable's exact effect upon the model is determined by the transfer function in front of it.

Estimation and Forecasting of ARIMA/TF

There are two forecasting algorithms available: Conditional Least Squares (CLS) and Exact Least Squares (ELS) or Unconditional Least Squares forecasting (ULS). These two algorithms differ in only one aspect: they forecast the noise process differently. The general steps in the forecasting computations are as follows:

1. Computation of noise process N_t through the historical period.
2. Forecasting the noise process N_t up to the forecast horizon. This is one step ahead forecasting during the historical period and multi-step ahead forecasting after that. The differences in CLS and ELS forecasting methodologies surface in this step. The prediction variances of noise forecasts are also computed in this step.
3. Final forecasts are obtained by first adding back to the noise forecasts the contributions of the constant term and the transfer function inputs and then integrating and back-transforming the result. The prediction variances of noise forecasts also may have to be processed to obtain the final prediction variances.

Let $\hat{N}_t\left(k\right)$ and $\sigma_t^2\left(k\right)$ be the k-step forecast and forecast variance, respectively.

Conditional Least Squares (CLS) Method

$\hat{N}_t\left(k\right) = E\left(N_{t+k}|N_t, N_{t-1}, \cdots\right)$ assuming $N_t = 0$ for t<0.

$$\sigma_t^2\left(k\right) = \sigma^2 \sum_{j=0}^{k-1} \psi_j^2$$

where ψ_j are coefficients of the power series expansion of $MA/\left(\Delta \times AR\right)$.

Minimize $S = \Sigma\left(N_t - \hat{N}_t\left(1\right)\right)^2$.

Missing values are imputed with forecast values of N_t.

Maximum Likelihood (ML) Method (Brockwell and Davis, 1991)

$$\hat{N}_t\left(k\right) = E\left(N_{t+k}|N_t, N_{t-1}, \cdots, N_1\right)$$

Maximize likelihood of $\left\{N_t - \hat{N}_t\left(1\right)\right\}_{t=1}^{n}$; that is,

$$L = -\ln\left(S/n\right) - \left(1/n\right)\sum_{j=1}^{n}\ln\left(\eta_j\right)$$

where $S = \Sigma\left(N_t - \hat{N}_t\left(1\right)\right)^2/\eta_t$, and $\sigma_t^2 = \sigma^2\eta_t$ is the one-step ahead forecast variance.

When missing values are present, a Kalman filter is used to calculate $\hat{N}_t\left(k\right)$.

Error Variance

$$\hat{\sigma}^2 = S/\left(n-k\right)$$

in both methods. Here *n* is the number of non-zero residuals and *k* is the number of parameters (excluding error variance).

Initialization of ARIMA/TF

A slightly modified Levenberg-Marquardt algorithm is used to optimize the objective function. The modification takes into account the “admissibility” constraints on the parameters. The admissibility constraint requires that the roots of AR and MA polynomials be outside the unit circle and the sum of denominator polynomial parameters be non-zero for each predictor variable. The minimization algorithm requires a starting value to begin its iterative search. All the numerator and denominator polynomial parameters are initialized to zero except the coefficient of the 0th power in the numerator polynomial, which is initialized to the corresponding regression coefficient.

The ARMA parameters are initialized as follows:

Assume that the series Y_t follows an ARMA(p,q)(P,Q) model with mean 0; that is:

$$Y_t - \varphi_1 Y_{t-1} - \cdots - \varphi_p Y_{t-p} = a_t - \theta_1 a_{t-1} - \cdots - \theta_q a_{t-q}$$

In the following c_l and ρ_l represent the *l*th lag autocovariance and autocorrelation of Y_t respectively, and $\hat{c}_l$ and $\hat{\rho}_l$ represent their estimates.

Non-Seasonal AR Parameters

For AR parameter initial values, the estimated method is the same as that in appendix A6.2 of (Box, Jenkins, and Reinsel, 1994). Denote the estimates as $\hat{\varphi}_1^{'}, \cdots, \hat{\varphi}_{p+q}^{'}$.

Non-Seasonal MA Parameters

Let

$$w_t = Y_t - \varphi_1 Y_{t-1} - \cdots - \varphi_p Y_{t-p} = a_t - \theta_1 a_{t-1} - \cdots - \theta_q a_{t-q}$$

The cross covariance

$$\lambda_l = E(w_{t+l}a_t) = E((a_{t+l} - \theta_1 a_{t+l-1} - \cdots - \theta_q a_{t+l-q})a_t) = \begin{cases} \sigma_a^2 & l = 0 \\ -\theta_1 \sigma_a^2 & l = 1 \\ \dots & \dots \\ -\theta_q \sigma_a^2 & l = q \\ 0 & l > q \end{cases}$$

Assuming that an AR(p+q) can approximate Y_t, it follows that:

$$Y_t - {\varphi'}_1 Y_{t-1} - \cdots - {\varphi'}_p Y_{t-p} - {\varphi'}_{p+1} Y_{t-p-1} - \cdots - {\varphi'}_{p+q} Y_{t-p-q} = a_t$$

The AR parameters of this model are estimated as above and are denoted as $\hat{\varphi}'_1, \cdots, \hat{\varphi}'_{p+q}$.

Thus λ_l can be estimated by

$$\begin{aligned} \lambda_l &\approx E\Big((Y_{t+l} - \varphi_1 Y_{t+l-1} - \cdots - \varphi_p Y_{t+l-p}) \left(Y_t - {\varphi'}_1 Y_{t-1} - \cdots - {\varphi'}_{p+q} Y_{t-p-q}\right)\Big) \\ &= \left(\rho_l - \sum_{j=1}^{p+q} \varphi_j \rho_{l+j} - \sum_{i=1}^{p} \varphi_i \rho_{l-i} + \sum_{i=1}^{p} \sum_{j=1}^{p+q} \varphi_i \varphi_j \rho_{l+j-i}\right) c_0 \end{aligned}$$

And the error variance σ_a^2 is approximated by

$$\hat{\sigma}_a^2 = Var\left(-\sum_{j=0}^{p+q} {\varphi'}_j Y_{t-j}\right) = \sum_{i=0}^{p+q} \sum_{j=0}^{p+q} {\varphi'}_i {\varphi'}_j c_{i-j} = c_0 \sum_{i=0}^{p+q} \sum_{j=0}^{p+q} {\varphi'}_i {\varphi'}_j \rho_{i-j}$$

with $\hat{\varphi}'_0 = -1$.

Then the initial MA parameters are approximated by $\theta_l = -\lambda_l / \sigma_a^2$ and estimated by

$$\hat{\theta}_l = -\hat{\lambda}_l / \hat{\sigma}_a^2 = \frac{\rho_l - \sum_{j=1}^{p+q} \hat{\varphi}_j \rho_{l+j} - \sum_{i=1}^{p} \hat{\varphi}_i \rho_{l-i} + \sum_{i=1}^{p} \sum_{j=1}^{p+q} \hat{\varphi}_i \hat{\varphi}_j \rho_{l+j-i}}{\sum_{i=0}^{p+q} \sum_{j=0}^{p+q} \hat{\varphi}'_i \hat{\varphi}'_j \rho_{i-j}}$$

So $\hat{\theta}_l$ can be calculated by $\hat{\varphi}'_j$, $\hat{\varphi}_i$, and $\{\hat{\rho}_l\}_{l=1}^{p+2q}$. In this procedure, only $\{\hat{\rho}_l\}_{l=1}^{p+q}$ are used and all other parameters are set to 0.

Seasonal parameters

For seasonal AR and MA components, the autocorrelations at the seasonal lags in the above equations are used.

Calculation of the Transfer Function

The transfer function needs to be calculated for each predictor series. For the predictor series X_{it}, let the transfer function be:

$$V_{it} = \frac{Num_i}{Den_i} \Delta_i B^{b_i} f_i(X_{it})$$

It can be calculated as follows:

1. Calculate $U_{it} = \Delta_i B^{b_i} f_i(X_{it})$

2. Recursively calculate

$$V_{it} = -\sum_{j=1}^{D_i} Cden_i\,(j) * V_{it-j} + \sum_{j=0}^{N_i} Cnum_i\,(j) * U_{it-j}$$

where $Cden_i\,(j)$ and $Cnum_i\,(j)$ are the coefficients of B^j in the polynomials Den_i and Num_i respectively. Likewise, the summation limits D_i and N_i are the maximum degree of B^j in the polynomials Den_i and Num_i respectively.

All missing V_{it-j} in the first term of V_{it} are taken to be $V_{i,-\infty}$ and missing U_{it-j} in the second term are taken to be $U_{i,-\infty}$, where $U_{i,-\infty}$ is the first non-missing measurement of U_{it}. $V_{i,-\infty}$ is given by

$$V_{i,-\infty} = \frac{Num_i\,(1)}{Den_i\,(1)} {*} U_{i,-\infty}$$

where $Num_i\,(1)$ and $Den_i\,(1)$ are the Num_i and Den_i polynomials evaluated at $B = 1$.

Diagnostic Statistics

ARIMA/TF diagnostic statistics are based on residuals of the noise process, $R\,(t) = N\,(t) - \hat{N}\,(t)$.

Ljung-Box Statistic

$$Q\,(K) = n\,(n+2) \sum_{k=1}^{K} r_k^2/(n-k)$$

where r_k is the kth lag ACF of residual.

Q(K) is approximately distributed as $\chi^2\,(K - m)$, where m is the number of parameters other than the constant term and predictor related-parameters.

Outlier Detection in Time Series Analysis

The observed series may be contaminated by so-called outliers. These outliers may change the mean level of the uncontaminated series. The purpose of outlier detection is to find if there are outliers and what are their locations, types, and magnitudes.

The Time Series node considers seven types of outliers. They are additive outliers (AO), innovational outliers (IO), level shift (LS), temporary (or transient) change (TC), seasonal additive (SA), local trend (LT), and AO patch (AOP).

Notation

The following notation is specific to outlier detection:

U(t) or U_t	The uncontaminated series, outlier free. It is assumed to be a univariate ARIMA or transfer function model.

Definitions of Outliers

Types of outliers are defined separately here. In practice any combination of these types can occur in the series under study.

AO (Additive Outliers)

Assuming that an AO outlier occurs at time *t=T*, the observed series can be represented as

$Y\left(t\right)=U\left(t\right)+wI_{T}\left(t\right)$

where $I_{T}\left(t\right)=\begin{cases}0 & t\neq T\\ 1 & t=T\end{cases}$ is a pulse function and w is the deviation from the true *U(T)* caused by the outlier.

IO (Innovational Outliers)

Assuming that an IO outlier occurs at time *t=T*, then

$Y\left(t\right)=\mu\left(t\right)+\frac{\theta(B)}{\Delta\varphi(B)}\left(a\left(t\right)+wI_{T}\left(t\right)\right)$

LS (Level Shift)

Assuming that a LS outlier occurs at time *t=T*, then

$Y\left(t\right)=U\left(t\right)+wS_{T}\left(t\right)$

where $S_{T}\left(t\right)=\frac{1}{1-B}I_{T}\left(t\right)=\begin{cases}0 & t<T\\ 1 & t\geq T\end{cases}$ is a step function.

TC (Temporary/Transient Change)

Assuming that a TC outlier occurs at time *t=T*, then

$Y\left(t\right)=U\left(t\right)+wD_{T}\left(t\right)$

where $D_{T}\left(t\right)=\frac{1}{1-\delta B}I_{T}\left(t\right)$, $0<\delta<1$ is a damping function.

SA (Seasonal Additive)

Assuming that a SA outlier occurs at time *t=T*, then

$$Y(t) = U(t) + wSS_T(t)$$

where $SS_T(t) = \frac{1}{1-B^s} I_T(t) = \begin{cases} 1 & t = T + ks, k \geq 0 \\ 0 & o.w. \end{cases}$ is a step seasonal pulse function.

LT (Local Trend)

Assuming that a LT outlier occurs at time *t*=*T*, then

$$Y(t) = U(t) + wT_T(t)$$

where $T_T(t) = \frac{1}{(1-B)^2} I_T(t) = \begin{cases} t + 1 - T & t \geq T \\ 0 & o.w. \end{cases}$ is a local trend function.

AOP (AO patch)

An AO patch is a group of two or more consecutive AO outliers. An AO patch can be described by its starting time and length. Assuming that there is a patch of AO outliers of length *k* at time *t*=*T*, the observed series can be represented as

$$Y(t) = U(t) + \sum_{i=1}^{k} w_i I_{T-1+i}(t)$$

Due to a masking effect, a patch of AO outliers is very difficult to detect when searching for outliers one by one. This is why the AO patch is considered as a separate type from individual AO. For type AO patch, the procedure searches for the whole patch together.

Summary

For an outlier of type O at time *t*=*T* (except AO patch):

$$Y(t) = \mu(t) + wL_O(B) I_T(t) + \frac{\theta(B)}{\Delta\varphi(B)} a(t)$$

where

$$L_O(B) = \begin{cases} 1 & O = AO \\ 1/(\Delta\pi(B)) & O = IO \\ 1/(1-B) & O = LS \\ 1/(1-\delta B) & O = TC \\ 1/(1-B^s) & O = SA \\ 1/(1-B)^2 & O = LT \end{cases}$$

with $\pi(B) = \varphi(B)/\theta(B)$. A general model for incorporating outliers can thus be written as follows:

$$Y(t) = \mu(t) + \sum_{k=1}^{M} w_k L_{O_k}(B) I_{T_k}(t) + \frac{\theta(B)}{\Delta\varphi(B)} a(t)$$

where *M* is the number of outliers.

Estimating the Effects of an Outlier

Suppose that the model and the model parameters are known. Also suppose that the type and location of an outlier are known. Estimation of the magnitude of the outlier and test statistics are as follows.

The results in this section are only used in the intermediate steps of outlier detection procedure. The final estimates of outliers are from the model incorporating all the outliers in which all parameters are jointly estimated.

Non-AO Patch Deterministic Outliers

For a deterministic outlier of any type at time T (except AO patch), let $e\left(t\right)$ be the residual and $x\left(t\right)=\pi\left(B\right)L\left(B\right)\Delta I_T\left(t\right)$, so:

$$e\left(t\right)=wx\left(t\right)+a\left(t\right)$$

From residuals *e(t)*, the parameters for outliers at time T are estimated by simple linear regression of *e(t)* on *x(t)*.

For j = 1 (AO), 2 (IO), 3 (LS), 4 (TC), 5 (SA), 6 (LT), define test statistics:

$$\lambda_j(\mathrm{T})=\frac{w_j(T)}{\sqrt{\mathrm{Var}(w_j(T))}}$$

Under the null hypothesis of no outlier, λ_j(T) is distributed as N(0,1) assuming the model and model parameters are known.

AO Patch Outliers

For an AO patch of length k starting at time T, let $x_i\left(t;T\right)=\pi\left(B\right)\Delta I_{T+i-1}\left(t\right)$ for i = 1 to k, then

$$e\left(t\right)=\sum_{i=1}^{k}w_i\left(T\right)x_i\left(t;T\right)+a\left(t\right)$$

Multiple linear regression is used to fit this model. Test statistics are defined as:

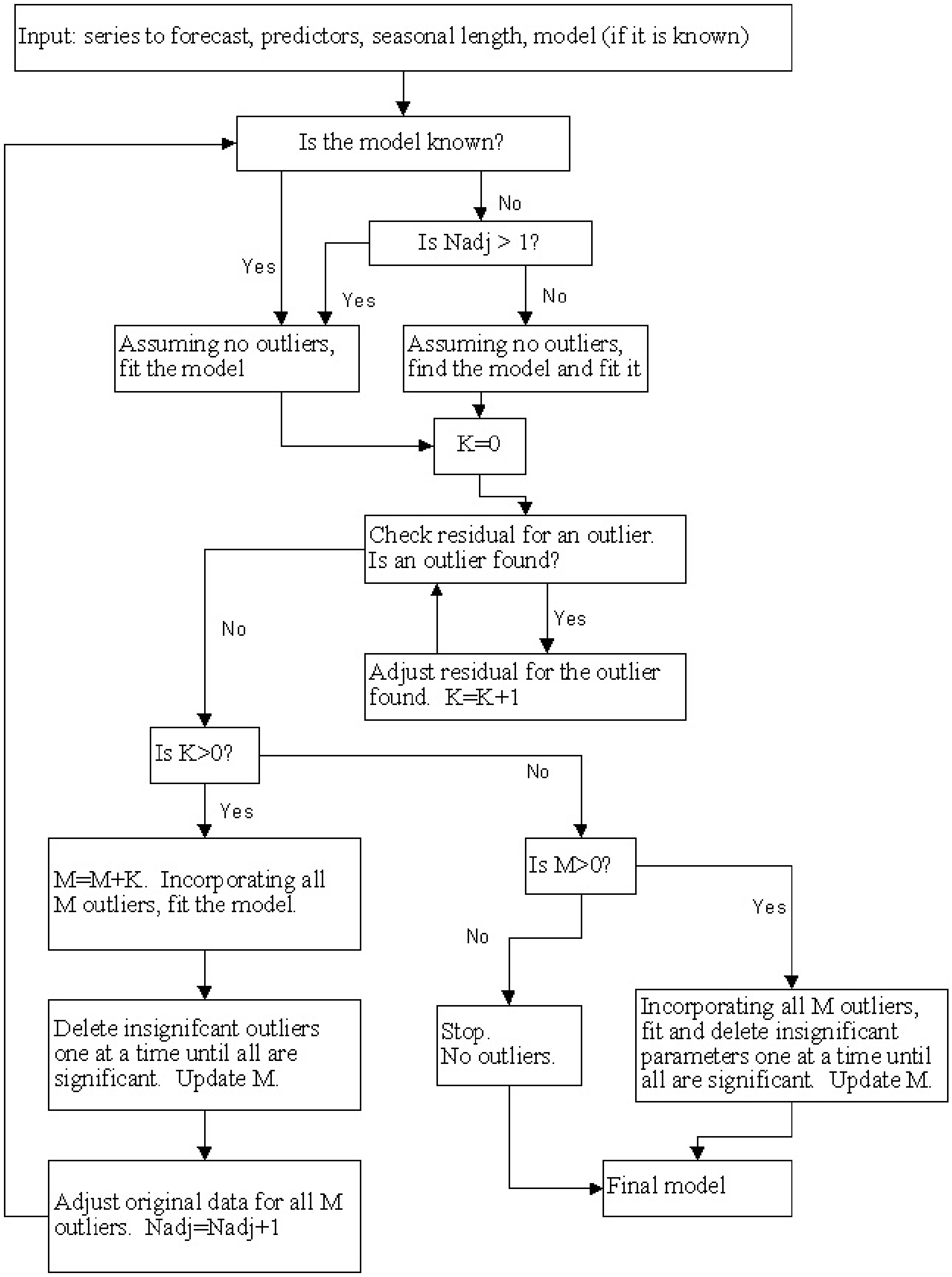
$$\chi^2\left(T\right)=\frac{\mathbf{w}^{'}(T)(X_TX_T)\mathbf{w}(T)}{\sigma^2}$$

Assuming the model and model parameters are known, $\chi^2\left(T\right)$ has a Chi-square distribution with k degrees of freedom under the null hypothesis $w_1\left(T\right)=\cdots=w_k\left(T\right)=0$.

Detection of Outliers

The following flow chart demonstrates how automatic outlier detection works. Let M be the total number of outliers and Nadj be the number of times the series is adjusted for outliers. At the beginning of the procedure, M = 0 and Nadj = 0.

Figure 35-1


Input: series to forecast, predictors, seasonal length, model (if it is known)
Is the model known?
No
Is Nadj > 1?
Yes
Yes
No
Assuming no outliers, fit the model
Assuming no outliers, find the model and fit it
K=0
Check residual for an outlier. Is an outlier found?
No
Yes
Adjust residual for the outlier found. K=K+1
Is K>0?
No
Yes
M=M+K. Incorporating all M outliers, fit the model.
Is M>0?
Yes
No
Delete insignifcant outliers one at a time until all are significant. Update M.
Stop. No outliers.
Incorporating all M outliers, fit and delete insignificant parameters one at a time until all are significant. Update M.
Final model
Adjust original data for all M outliers. Nadj=Nadj+1

Goodness-of-Fit Statistics

Goodness-of-fit statistics are based on the original series Y(t). Let k= number of parameters in the model, n = number of non-missing residuals.

Mean Squared Error

$$MSE = \frac{\Sigma\left(Y\left(t\right) - \hat{Y}\left(t\right)\right)^2}{n-k}$$

Mean Absolute Percent Error

$$MAPE = \tfrac{100}{n}\Sigma\left|\left(Y\left(t\right) - \hat{Y}\left(t\right)\right)/Y\left(t\right)\right|$$

Maximum Absolute Percent Error

$$MaxAPE = 100\max\left(\left|\left(Y\left(t\right) - \hat{Y}\left(t\right)\right)/Y\left(t\right)\right|\right)$$

Mean Absolute Error

$$MAE = \tfrac{1}{n}\Sigma\left|Y\left(t\right) - \hat{Y}\left(t\right)\right|$$

Maximum Absolute Error

$$MaxAE = \max\left(\left|Y\left(t\right) - \hat{Y}\left(t\right)\right|\right)$$

Normalized Bayesian Information Criterion

$$\text{Normalized } BIC = \ln\left(MSE\right) + k\tfrac{\ln(n)}{n}$$

R-Squared

$$R^2 = 1 - \frac{\Sigma\left(Y\left(t\right) - \hat{Y}\left(t\right)\right)^2}{\Sigma\left(Y\left(t\right) - \overline{Y}\right)^2}$$

Stationary R-Squared

A similar statistic was used by Harvey (Harvey, 1989).

$$R_S^2 = 1 - \frac{\sum\limits_t\left(Z\left(t\right) - \hat{Z}\left(t\right)\right)^2}{\sum\limits_t\left(\Delta Z\left(t\right) - \overline{\Delta Z}\right)^2}$$

where

The sum is over the terms in which both $Z\left(t\right) - \hat{Z}\left(t\right)$ and $\Delta Z\left(t\right) - \overline{\Delta Z}$ are not missing.

$\overline{\Delta Z}$ is the simple mean model for the differenced transformed series, which is equivalent to the univariate baseline model ARIMA(0,d,0)(0,D,0).

For the exponential smoothing models currently under consideration, use the differencing orders (corresponding to their equivalent ARIMA models if there is one).

$$d = \begin{cases} 2 & \text{Brown, Holt} \\ 1 & \text{other} \end{cases}, D = \begin{cases} 0 & s = 1 \\ 1 & s > 1 \end{cases}$$

Note: Both the stationary and usual R-squared can be negative with range $(-\infty, 1]$. A negative R-squared value means that the model under consideration is worse than the baseline model. Zero R-squared means that the model under consideration is as good or bad as the baseline model. Positive R-squared means that the model under consideration is better than the baseline model.

Expert Modeling

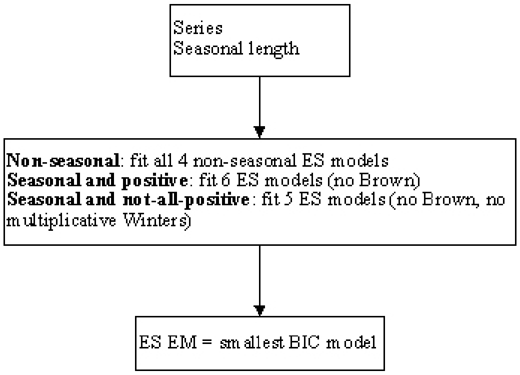
Univariate Series

Users can let the Expert Modeler select a model for them from:

- All models (default).
- Exponential smoothing models only.
- ARIMA models only.

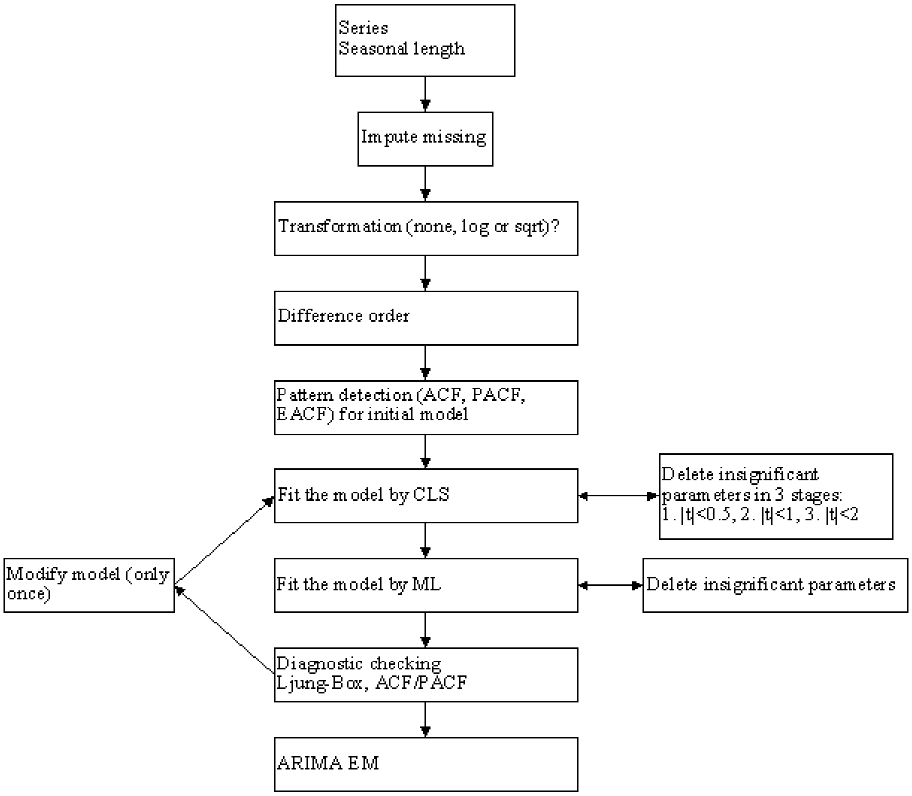
Exponential Smoothing Expert Model

Figure 35-2


Series
Seasonal length
Non-seasonal: fit all 4 non-seasonal ES models
Seasonal and positive: fit 6 ES models (no Brown)
Seasonal and not-all-positive: fit 5 ES models (no Brown, no multiplicative Winters)
ES EM = smallest BIC model

ARIMA Expert Model

Figure 35-3


Series
Seasonal length
Impute missing
Transformation (none, log or sqrt)?
Difference order
Pattern detection (ACF, PACF, EACF) for initial model
Fit the model by CLS
Delete insignificant parameters in 3 stages: 1. |t|<0.5, 2. |t|<1, 3. |t|<2
Modify model (only once)
Fit the model by ML
Delete insignificant parameters
Diagnostic checking Ljung-Box, ACF/PACF
ARIMA EM

Note: If 10<n<3s, set s=1 to build a non-seasonal model.

All Models Expert Model

In this case, the Exponential Smoothing and ARIMA expert models are computed, and the model with the smaller normalized BIC is chosen.

Note: For short series, n<max(20,3s), use Exponential Smoothing Expert Model on p. 393.

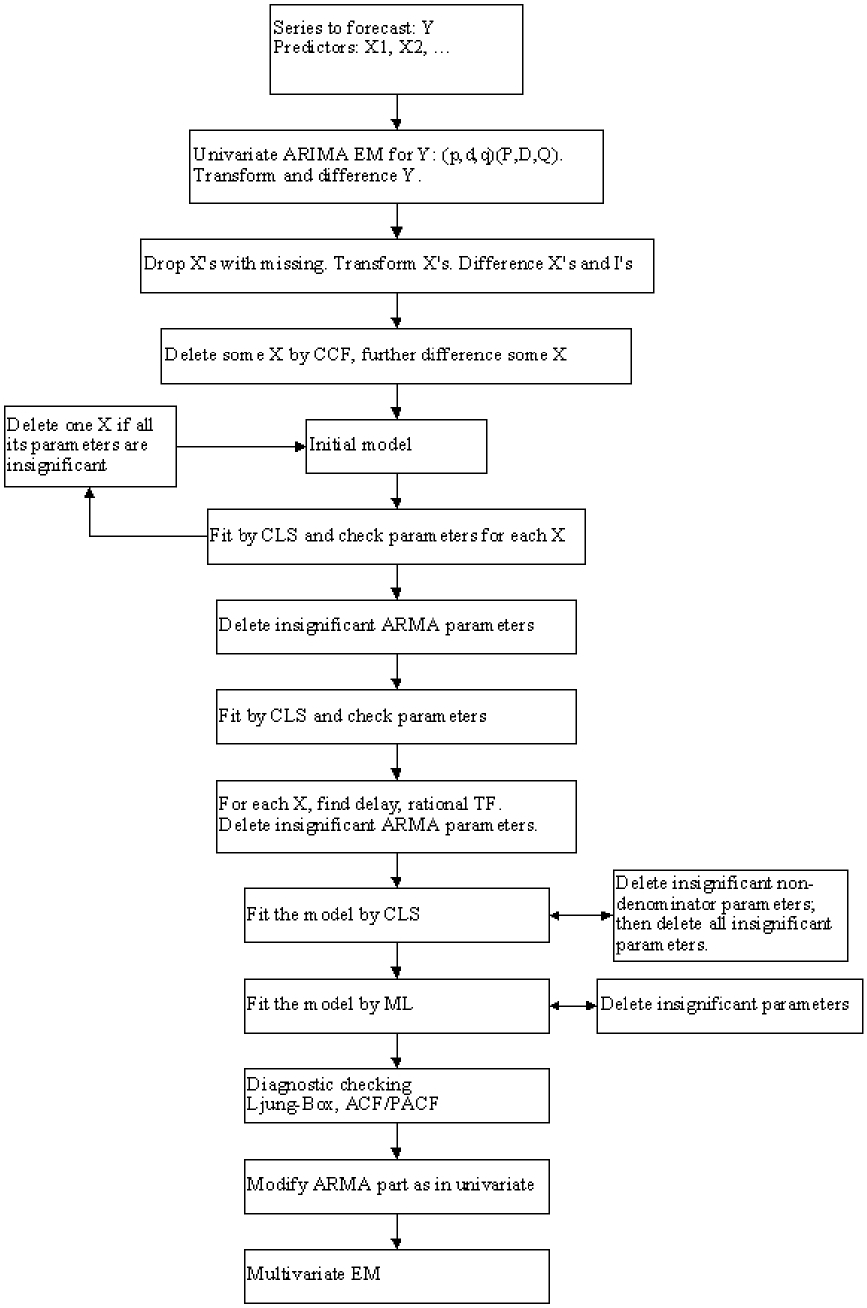
Multivariate Series

In the multivariate situation, users can let the Expert Modeler select a model for them from:

- All models (default). Note that if the multivariate expert ARIMA model drops all the predictors and ends up with a univariate expert ARIMA model, this univariate expert ARIMA model will be compared with expert exponential smoothing models as before and the Expert Modeler will decide which is the best overall model.
- ARIMA models only.

Transfer Function Expert Model

Figure 35-4


Series to forecast: Y
Predictors: X1, X2, ...
Univariate ARIMA EM for Y: (p,d,q)(P,D,Q).
Transform and difference Y.
Drop X's with missing. Transform X's. Difference X's and I's
Delete some X by CCF, further difference some X
Delete one X if all its parameters are insignificant
Initial model
Fit by CLS and check parameters for each X
Delete insignificant ARMA parameters
Fit by CLS and check parameters
For each X, find delay, rational TF.
Delete insignificant ARMA parameters.
Fit the model by CLS
Delete insignificant non-denominator parameters; then delete all insignificant parameters.
Fit the model by ML
Delete insignificant parameters
Diagnostic checking
Ljung-Box, ACF/PACF
Modify ARMA part as in univariate
Multivariate EM

Note: For short series, n<max(20,3s), fit a univariate expert model.

Blank Handling

Generally, any missing values in the series data will be imputed in the Time Intervals node used to prepare the data for time series modeling. If blanks remain in the series data submitted to the modeling node, ARIMA models will attempt to impute values, as described in “Estimation and Forecasting of ARIMA/TF” on p. 384.

Missing values for predictors will result in the field containing the missing values to be excluded from the time series model.

Generated Model/Scoring

Predictions or forecasts for Time Series models are intricately related to the modeling process itself. Forecasting computations are described with the algorithm for the corresponding model type. For information on forecasting in exponential smoothing models, see “Exponential Smoothing Models” on p. 379. For information on forecasting in ARIMA models, see “Estimation and Forecasting of ARIMA/TF” on p. 384.

Blank Handling

Blank handling for the generated model is very similar to that for the modeling node.

If any predictor has missing values within the forecast period, the procedure issues a warning and forecasts as far as it can.

References

Box, G. E. P., G. M. Jenkins, and G. C. Reinsel. 1994. *Time series analysis: Forecasting and control*, 3rd ed. Englewood Cliffs, N.J.: Prentice Hall.

Brockwell, P. J., and R. A. Davis. 1991. *Time Series: Theory and Methods*, 2 ed. : Springer-Verlag.

Gardner, E. S. 1985. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4, 1–28.

Harvey, A. C. 1989. *Forecasting, structural time series models and the Kalman filter*. Cambridge: Cambridge University Press.

Makridakis, S. G., S. C. Wheelwright, and R. J. Hyndman. 1997. *Forecasting: Methods and applications*, 3rd ed. ed. New York: John Wiley and Sons.

Melard, G. 1984. A fast algorithm for the exact likelihood of autoregressive-moving average models. *Applied Statistics*, 33:1, 104–119.

Pena, D., G. C. Tiao, and R. S. Tsay, eds. 2001. *A course in time series analysis*. New York: John Wiley and Sons.

TwoStep Cluster Algorithms

Overview

The TwoStep cluster method is a scalable cluster analysis algorithm designed to handle very large data sets. It can handle both continuous and categorical variables or attributes. It requires only one data pass. It has two steps 1) pre-cluster the cases (or records) into many small sub-clusters; 2) cluster the sub-clusters resulting from pre-cluster step into the desired number of clusters. It can also automatically select the number of clusters.

Model Parameters

As the name implies, the TwoStep clustering algorithm involves two steps: Pre-clustering and Clustering.

Pre-cluster

The pre-cluster step uses a sequential clustering approach. It scans the data records one by one and decides if the current record should be merged with the previously formed clusters or starts a new cluster based on the distance criterion (described below).

The procedure is implemented by constructing a modified cluster feature (CF) tree. The CF tree consists of levels of nodes, and each node contains a number of entries. A leaf entry (an entry in the leaf node) represents a final sub-cluster. The non-leaf nodes and their entries are used to guide a new record quickly into a correct leaf node. Each entry is characterized by its CF that consists of the entry's number of records, mean and variance of each range field, and counts for each category of each symbolic field. For each successive record, starting from the root node, it is recursively guided by the closest entry in the node to find the closest child node, and descends along the CF tree. Upon reaching a leaf node, it finds the closest leaf entry in the leaf node. If the record is within a threshold distance of the closest leaf entry, it is absorbed into the leaf entry and the CF of that leaf entry is updated. Otherwise it starts its own leaf entry in the leaf node. If there is no space in the leaf node to create a new leaf entry, the leaf node is split into two. The entries in the original leaf node are divided into two groups using the farthest pair as seeds, and redistributing the remaining entries based on the closeness criterion.

If the CF tree grows beyond allowed maximum size, the CF tree is rebuilt based on the existing CF tree by increasing the threshold distance criterion. The rebuilt CF tree is smaller and hence has space for new input records. This process continues until a complete data pass is finished. For details of CF tree construction, see the BIRCH algorithm (Zhang, Ramakrishnon, and Livny, 1996).

All records falling in the same entry can be collectively represented by the entry's CF. When a new record is added to an entry, the new CF can be computed from this new record and the old CF without knowing the individual records in the entry. These properties of CF make it possible to maintain only the entry CFs, rather than the sets of individual records. Hence the CF-tree is much smaller than the original data and can be stored in memory more efficiently.

Note that the structure of the constructed CF tree may depend on the input order of the cases or records. To minimize the order effect, randomly order the records before building the model.

Cluster

The cluster step takes sub-clusters (non-outlier sub-clusters if outlier handling is used) resulting from the pre-cluster step as input and then groups them into the desired number of clusters. Since the number of sub-clusters is much less than the number of original records, traditional clustering methods can be used effectively. TwoStep uses an agglomerative hierarchical clustering method, because it works well with the auto-cluster method (see the section on auto-clustering below).

Hierarchical clustering refers to a process by which clusters are recursively merged, until at the end of the process only one cluster remains containing all records. The process starts by defining a starting cluster for each of the sub-clusters produced in the pre-cluster step. (For more information, see the topic “Pre-cluster” on p. 397.) All clusters are then compared, and the pair of clusters with the smallest distance between them is selected and merged into a single cluster. After merging, the new set of clusters is compared, the closest pair is merged, and the process repeats until all clusters have been merged. (If you are familiar with the way a decision tree is built, this is a similar process, except in reverse.) Because the clusters are merged recursively in this way, it is easy to compare solutions with different numbers of clusters. To get a five-cluster solution, simply stop merging when there are five clusters left; to get a four-cluster solution, take the five-cluster solution and perform one more merge operation, and so on.

Distance Measure

The TwoStep clustering method uses a log-likelihood distance measure, to accommodate both symbolic and range fields. It is a probability-based distance. The distance between two clusters is related to the decrease in log-likelihood as they are combined into one cluster. In calculating log-likelihood, normal distributions for range fields and multinomial distributions for symbolic fields are assumed. It is also assumes that the fields are independent of each other, and so are the records. The distance between clusters i and j is defined as

$$d\left(i,j\right) = \xi_i + \xi_j - \xi_{\langle i,j \rangle}$$

where

$$\xi_v = -N_v \left(\sum_{k=1}^{K^A} \frac{1}{2} \log \left(\hat{\sigma}_k^2 + \hat{\sigma}_{vk}^2 \right) + \sum_{k=1}^{K^B} \hat{E}_{vk} \right)$$

and

$$\hat{E}_{vk} = -\sum_{l=1}^{L_k} \frac{N_{vkl}}{N_v} \log \frac{N_{vkl}}{N_v}$$

In these expressions,

K^{A} is the number of range type input fields,

K^{B} is the number of symbolic type input fields,

L_k is the number of categories for the *k*th symbolic field,

N_v is the number of records in cluster *v*,

N_vkl is the number of records in cluster *v* which belongs to the *l*th category of the *k*th symbolic field,

$\hat{\sigma}_k^2$ is the estimated variance of the *k*th continuous variable for all records,

$\hat{\sigma}_{vk}^2$ is the estimated variance of the *k*th continuous variable for records in the *v*th cluster, and

< *i*, *j* > is an index representing the cluster formed by combining clusters *i* and *j*.

If $\hat{\sigma}_k^2$ is ignored in the expression for ξ_v, the distance between clusters *i* and *j* would be exactly the decrease in log-likelihood when the two clusters are combined. The $\hat{\sigma}_k^2$ term is added to solve the problem caused by $\hat{\sigma}_{vk}^2 = 0$, which would result in the natural logarithm being undefined. (This would occur, for example, when a cluster has only one case.)

Number of Clusters (auto-clustering)

TwoStep can use the hierarchical clustering method in the second step to assess multiple cluster solutions and automatically determine the optimal number of clusters for the input data. A characteristic of hierarchical clustering is that it produces a sequence of partitions in one run: 1, 2, 3, … clusters. In contrast, a *k*-means algorithm would need to run multiple times (one for each specified number of clusters) in order to generate the sequence. To determine the number of clusters automatically, TwoStep uses a two-stage procedure that works well with the hierarchical clustering method. In the first stage, the BIC for each number of clusters within a specified range is calculated and used to find the initial estimate for the number of clusters. The BIC is computed as

$$BIC(J) = -2\sum_{j=1}^{J} \xi_j + m_J \log(N)$$

where

$$m_J = J\left\{2K^A + \sum_{k=1}^{K^B} (L_K - 1)\right\}$$

and other terms defined as in “Distance Measure”. The ratio of change in BIC at each successive merging relative to the first merging determines the initial estimate. Let $dBIC(J)$ be the difference in BIC between the model with J clusters and that with (J + 1) clusters, $dBIC(J) = BIC(J) - BIC(J+1)$. Then the change ratio for model J is

$$R_1(J) = \frac{dBIC(J)}{dBIC(1)}$$

If $dBIC(1) < 0$, then the number of clusters is set to 1 (and the second stage is omitted). Otherwise, the initial estimate for number of clusters$k$ is the smallest number for which $R_1(J) < 0.04$

In the second stage, the initial estimate is refined by finding the largest relative increase in distance between the two closest clusters in each hierarchical clustering stage. This is done as follows:

- Starting with the model C_{k} indicated by the BIC criterion, take the ratio of minimum inter-cluster distance for that model and the next larger model $C_{\mathrm{k+1}}$, that is, the previous model in the hierarchical clustering procedure,

  $$R_2(k) = \frac{d_{\min}(C_k)}{d_{\min}(C_{k+1})}$$

  where C_{k} is the cluster model containing k clusters and $d_{\mathrm{min}}(C)$ is the minimum inter-cluster distance for cluster model C.

- Now from model $C_{\mathrm{k\text{-}1}}$, compute the same ratio with the following model C_{k}, as above. Repeat for each subsequent model until you have the ratio $R_2(2)$.

- Compare the two largest R_2 ratios; if the largest is more that 1.15 times the second largest, then select the model with the largest R_2 ratio as the optimal number of clusters; otherwise, from those two models with the largest R_2 values, select the one with the larger number of clusters as the optimal model.

Blank Handling

The TwoStep cluster node does not support blanks. Records containing blanks, nulls, or missing values of any kind are excluded from model building.

Effect of Options

Outlier Handling

An optional outlier-handling step is implemented in the algorithm in the process of building the CF tree. Outliers are considered as data records that do not fit well into any cluster. We consider data records in a leaf entry as outliers if the number of records in the entry is less than a certain fraction (25% by default) of the size of the largest leaf entry in the CF tree. Before rebuilding the CF tree, the procedure checks for potential outliers and sets them aside. After rebuilding the CF tree, the procedure checks to see if these outliers can fit in without increasing the tree size. At the end of CF tree building, small entries that cannot fit in are outliers.

Generated Model/Scoring

Predicted Values

When scoring a record with a TwoStep Cluster generated model, the record is assigned to the cluster to which it is closest. The distance between the record and each cluster is calculated, and the cluster with the smallest distance is selected as the closest, and that cluster is assigned as the predicted value for the record. Distance is calculated in a similar manner to that used during model building, considering the record to be scored as a "cluster" with only one record. For more information, see the topic "Distance Measure" on p. 398.

If outlier handling was enabled during model building, the distance between the record and the closest cluster is compared to a threshold C = log(V), where

$$V = \prod_k R_k \prod_m L_m.$$

where R_k is the range of the kth numeric field and L_m is number of categories for the mth symbolic field.

If the distance from the nearest cluster is smaller than C, assign that cluster as the predicted value for the record. If the distance is greater than C, assign the record as an outlier.

Blank Handling

As with model building, records containing blanks are not handled by the model, and are assigned a predicted value of $null$.

Appendix

A

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing, Legal and Intellectual Property Law, IBM Japan Ltd., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, and SPSS are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at *http://www.ibm.com/legal/copytrade.shtml*.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Java

COMPATIBLE

Bibliography

Aggarwal, C. C., and P. S. Yu. 1998. Online generation of association rules. In: *Proceedings of the 14th International Conference on Data Engineering,* Los Alamitos, Calif: IEEE ComputerSociety Press, 402–411.

Agrawal, R., and R. Srikant. 1994. Fast Algorithms for Mining Association Rules. In: *Proceedings of the 20th International Conference on Very Large Databases,* J. B. Bocca, M. Jarke, and C. Zaniolo, eds. San Francisco: Morgan Kaufmann, 487–499.

Agrawal, R., and R. Srikant. 1995. Mining Sequential Patterns. In: *Proceedings of the Eleventh International Conference on Data Engineering,* Los Alamitos, Calif.: IEEE Computer Society Press, 3–14.

Agresti, A., J. G. Booth, and B. Caffo. 2000. Random-effects Modeling of Categorical Response Data. *Sociological Methodology*, 30, 27–80.

Aitkin, M., D. Anderson, B. Francis, and J. Hinde. 1989. *Statistical Modelling in GLIM*. Oxford: Oxford Science Publications.

Albert, A., and J. A. Anderson. 1984. On the Existence of Maximum Likelihood Estimates in Logistic Regression Models. *Biometrika*, 71, 1–10.

Anderson, T. W. 1958. *Introduction to multivariate statistical analysis*. New York: John Wiley & Sons, Inc..

Arya, S., and D. M. Mount. 1993. Algorithms for fast vector quantization. In: *Proceedings of the Data Compression Conference 1993,* , 381–390.

Belsley, D. A., E. Kuh, and R. E. Welsch. 1980. *Regression diagnostics: Identifying influential data and sources of collinearity*. New York: John Wiley and Sons.

Biggs, D., B. de Ville, and E. Suen. 1991. A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18, 49–62.

Biller, B., and S. Ghosh. 2006. Multivariate input processes. In: *Handbooks in Operations Research and Management Science: Simulation,* B. L. Nelson, and S. G. Henderson, eds. Amsterdam: Elsevier Science, 123–153.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*, 3rd ed. Oxford: Oxford University Press.

Box, G. E. P., and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society, Series B*, 26, 211–246.

Box, G. E. P., G. M. Jenkins, and G. C. Reinsel. 1994. *Time series analysis: Forecasting and control*, 3rd ed. Englewood Cliffs, N.J.: Prentice Hall.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. New York: Chapman & Hall/CRC.

Breslow, N. E. 1974. Covariance analysis of censored survival data. *Biometrics*, 30, 89–99.

Brockwell, P. J., and R. A. Davis. 1991. *Time Series: Theory and Methods*, 2 ed. : Springer-Verlag.

Cain, K. C., and N. T. Lange. 1984. Approximate case influence for the proportional hazards regression model with censored data. *Biometrics*, 40, 493–499.

Cameron, A. C., and P. K. Trivedi. 1998. *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Chang, C. C., and C. J. Lin. 2003. *LIBSVM: A library for support vector machines. Technical Report*. Taipei, Taiwan: Department of Computer Science, National Taiwan University.

Chow, C. K., and C. N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14, 462–467.

Cooley, W. W., and P. R. Lohnes. 1971. *Multivariate data analysis*. New York: John Wiley & Sons, Inc..

Cox, D. R. 1972. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society, Series B*, 34, 187–220.

Cunningham, P., and S. J. Delaney. 2007. k-Nearest Neighbor Classifiers. *Technical Report UCD-CSI-2007-4, School of Computer Science and Informatics, University College Dublin, Ireland*, , – .

Dempster, A. P. 1969. *Elements of Continuous Multivariate Analysis*. Reading, MA: Addison-Wesley.

Diggle, P. J., P. Heagerty, K. Y. Liang, and S. L. Zeger. 2002. *The analysis of Longitudinal Data*, 2 ed. Oxford: Oxford University Press.

Dixon, W. J. 1973. *BMD Biomedical computer programs*. Los Angeles: University of California Press.

Dobson, A. J. 2002. *An Introduction to Generalized Linear Models*, 2 ed. Boca Raton, FL: Chapman & Hall/CRC.

Dong, J., C. Y. Suen, and A. Krzyzak. 2005. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 603–618.

Dougherty, J., R. Kohavi, and M. Sahami. 1995. Supervised and unsupervised discretization of continuous features. In: *Proceedings of the Twelfth International Conference on Machine Learning,* Los Altos, CA: Morgan Kaufmann, 194–202.

Drucker, H. 1997. Improving regressor using boosting techniques. In: *Proceedings of the 14th International Conferences on Machine Learning ,* D. H. Fisher,Jr., ed. San Mateo, CA: Morgan Kaufmann, 107–115.

Dunn, P. K., and G. K. Smyth. 2005. Series Evaluation of Tweedie Exponential Dispersion Model Densities. *Statistics and Computing*, 15, 267–280.

Dunn, P. K., and G. K. Smyth. 2001. Tweedie Family Densities: Methods of Evaluation. In: *Proceedings of the 16th International Workshop on Statistical Modelling,* Odense, Denmark: .

D'Agostino, R., and M. Stephens. 1986. *Goodness-of-Fit Techniques*. New York: Marcel Dekker.

Fahrmeir, L., and G. Tutz. 2001. *Multivariate Statistical Modelling Based on Generalized Linear Models*, 2nd ed. New York: Springer-Verlag.

Fan, R. E., P. H. Chen, and C. J. Lin. 2005. *Working set selection using the second order information for training SVM. Technical Report*. Taipei, Taiwan: Department of Computer Science, National Taiwan University.

Fayyad, U., and K. Irani. 1993. Multi-interval discretization of continuous-value attributes for classification learning. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* San Mateo, CA: Morgan Kaufmann, 1022–1027.

Fine, T. L. 1999. *Feedforward Neural Network Methodology*, 3rd ed. New York: Springer-Verlag.

Fox, J., and G. Monette. 1992. Generalized collinearity diagnostics. *Journal of the American Statistical Association*, 87, 178–183.

Fox, J. 1997. *Applied Regression Analysis, Linear Models, and Related Methods*. Thousand Oaks, CA: SAGE Publications, Inc..

Freund, Y., and R. E. Schapire. 1995. A decision theoretic generalization of on-line learning and an application to boosting. In: *Computational Learning Theory: 7 Second European Conference, EuroCOLT '95,* , 23–37.

Friedman, J. H., J. L. Bentley, and R. A. Finkel. 1977. An algorithm for finding best matches in logarithm expected time. *ACM Transactions on Mathematical Software*, 3, 209–226.

Friedman, N., D. Geiger, and M. Goldszmidt. 1997. Bayesian network classifiers. *Machine Learning*, 29, 131–163.

Gardner, E. S. 1985. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4, 1–28.

Gill, J. 2000. *Generalized Linear Models: A Unified Approach*. Thousand Oaks, CA: Sage Publications.

Goodman, L. A. 1979. Simple models for the analysis of association in cross-classifications having ordered categories. *Journal of the American Statistical Association*, 74, 537–552.

Hardin, J. W., and J. M. Hilbe. 2003. *Generalized Linear Models and Extension*. Station, TX: Stata Press.

Hardin, J. W., and J. M. Hilbe. 2001. *Generalized Estimating Equations*. Boca Raton, FL: Chapman & Hall/CRC.

Harman, H. H. 1976. *Modern Factor Analysis*, 3rd ed. Chicago: University of Chicago Press.

Hartzel, J., A. Agresti, and B. Caffo. 2001. Multinomial Logit Random Effects Models. *Statistical Modelling*, 1, 81–102.

Harvey, A. C. 1989. *Forecasting, structural time series models and the Kalman filter*. Cambridge: Cambridge University Press.

Haykin, S. 1998. *Neural Networks: A Comprehensive Foundation*, 2nd ed. New York: Macmillan College Publishing.

Heckerman, D. 1999. A Tutorial on Learning with Bayesian Networks. In: *Learning in Graphical Models,* M. I. Jordan, ed. Cambridge, MA: MIT Press, 301–354.

Hedeker, D. 1999. Generalized Linear Mixed Models. In: *Encyclopedia of Statistics in Behavioral Science,* B. Everitt, and D. Howell, eds. London: Wiley, 729–738.

Hendrickson, A. E., and P. O. White. 1964. Promax: a quick method for rotation to oblique simple structure. *British Journal of Statistical Psychology*, 17, 65–70.

Hidber, C. 1999. Online Association Rule Mining. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data,* New York: ACM Press, 145–156.

Horton, N. J., and S. R. Lipsitz. 1999. Review of Software to Fit Generalized Estimating Equation Regression Models. *The American Statistician*, 53, 160–169.

Hosmer, D. W., and S. Lemeshow. 2000. *Applied Logistic Regression*, 2nd ed. New York: John Wiley and Sons.

Huber, P. J. 1967. The Behavior of Maximum Likelihood Estimates under Nonstandard Conditions. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* Berkeley, CA: University of California Press, 221–233.

Jennrich, R. I., and P. F. Sampson. 1966. Rotation for simple loadings. *Psychometrika*, 31, 313–323.

Johnson, N. L., S. Kotz, and A. W. Kemp. 2005. *Univariate Discrete Distributions*, 3rd ed. Hoboken, New Jersey: John Wiley & Sons.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The statistical analysis of failure time data*, 2 ed. New York: John Wiley & Sons, Inc.

Kass, G. 1980. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29:2, 119–127.

Kaufman, L., and P. J. Rousseeuw. 1990. *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley and Sons.

Kohavi, R., B. Becker, and D. Sommerfield. 1997. Improving Simple Bayes. In: *Proceedings of the European Conference on Machine Learning,* , 78–87.

Kohonen, T. 2001. *Self-Organizing Maps*, 3rd ed. New York: Springer-Verlag.

Kotz, S., and J. Rene Van Dorp. 2004. *Beyond Beta, Other Continuous Families of Distributions with Bounded Support and Applications*. Singapore: World Scientific Press.

Kroese, D. P., T. Taimre, and Z. I. Botev. 2011. *Handbook of Monte Carlo Methods*. Hoboken, New Jersey: John Wiley & Sons.

Lane, P. W., and J. A. Nelder. 1982. Analysis of Covariance and Standardization as Instances of Prediction. *Biometrics*, 38, 613–621.

Lawless, R. F. 1982. *Statistical models and methods for lifetime data*. New York: John Wiley & Sons, Inc..

Lawless, J. E. 1984. Negative Binomial and Mixed Poisson Regression. *The Canadian Journal of Statistics*, 15, 209–225.

Liang, K. Y., and S. L. Zeger. 1986. Longitudinal Data Analysis Using Generalized Linear Models. *Biometrika*, 73, 13–22.

Lipsitz, S. H., K. Kim, and L. Zhao. 1994. Analysis of Repeated Categorical Data Using Generalized Estimating Equations. *Statistics in Medicine*, 13, 1149–1163.

Liu, H., F. Hussain, C. L. Tan, and M. Dash. 2002. Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, 6, 393–423.

Loh, W. Y., and Y. S. Shih. 1997. Split selection methods for classification trees. *Statistica Sinica*, 7, 815–840.

Makridakis, S. G., S. C. Wheelwright, and R. J. Hyndman. 1997. *Forecasting: Methods and applications*, 3rd ed. ed. New York: John Wiley and Sons.

Marsaglia, G., and J. Marsaglia. 2004. Evaluating the Anderson-Darling Distribution. *Journal of Statistical Software*, 9:2, .

McCullagh, P. 1983. Quasi-Likelihood Functions. *Annals of Statistics*, 11, 59–67.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

McCulloch, C. E., and S. R. Searle. 2001. *Generalized, Linear, and Mixed Models*. New York: John Wiley and Sons.

Melard, G. 1984. A fast algorithm for the exact likelihood of autoregressive-moving average models. *Applied Statistics*, 33:1, 104–119.

Miller, M. E., C. S. Davis, and J. R. Landis. 1993. The Analysis of Longitudinal Polytomous Data: Generalized Estimating Equations and Connections with Weighted Least Squares. *Biometrics*, 49, 1033–1044.

Nelder, J. A., and R. W. M. Wedderburn. 1972. Generalized Linear Models. *Journal of the Royal Statistical Society Series A*, 135, 370–384.

Neter, J., W. Wasserman, and M. H. Kutner. 1990. *Applied Linear Statistical Models*, 3rd ed. Homewood, Ill.: Irwin.

Pan, W. 2001. Akaike's Information Criterion in Generalized Estimating Equations. *Biometrics*, 57, 120–125.

Pena, D., G. C. Tiao, and R. S. Tsay, eds. 2001. *A course in time series analysis*. New York: John Wiley and Sons.

Platt, J. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: *Advances in Large Margin Classifiers,* A. J. Smola, P. Bartlett, B. Scholkopf, and D. Schuumans, eds. Cambridge, MA: MITPress, 61–74.

Pregibon, D. 1981. Logistic Regression Diagnostics. *Annals of Statistics*, 9, 705–724.

Prim, R. C. 1957. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36, 1389–1401.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press.

Saltelli, A., S. Tarantola, F. , F. Campolongo, and M. Ratto. 2004. *Sensitivity Analysis in Practice – A Guide to Assessing Scientific Models*. : John Wiley.

Saltelli, A. 2002. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145:2, 280–297.

Schatzoff, M., R. Tsao, and S. Fienberg. 1968. Efficient computing of all possible regressions. *Technometrics*, 10, 769–779.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

Smyth, G. K., and B. Jorgensen. 2002. Fitting Tweedie's Compound Poisson Model to Insurance Claims Data: Dispersion Modelling. *ASTIN Bulletin*, 32, 143–157.

Sobol, I. M. 2001. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55, 271–280.

Storer, B. E., and J. Crowley. 1985. A diagnostic for Cox regression and general conditional likelihoods. *Journal of the American Statistical Association*, 80, 139–147.

Tan, P., M. Steinbach, and V. Kumar. 2006. *Introduction to Data Mining*. : Addison-Wesley.

Tao, K. K. 1993. A closer look at the radial basis function (RBF) networks. In: *Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers,* A. Singh, ed. Los Alamitos, Calif.: IEEE Comput. Soc. Press, 401–405.

Tatsuoka, M. M. 1971. *Multivariate analysis*. New York: John Wiley & Sons, Inc. .

Tuerlinckx, F., F. Rijmen, G. Molenberghs, G. Verbeke, D. Briggs, W. Van den Noortgate, M. Meulders, and P. De Boeck. 2004. Estimation and Software. In: *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach,* P. De Boeck, and M. Wilson, eds. New York: Springer-Verlag, 343–373.

Uykan, Z., C. Guzelis, M. E. Celebi, and H. N. Koivo. 2000. Analysis of input-output clustering for determining centers of RBFN. *IEEE Transactions on Neural Networks*, 11, 851–858.

Velleman, P. F., and R. E. Welsch. 1981. Efficient computing of regression diagnostics. *American Statistician*, 35, 234–242.

White, H. 1980. A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica*, 48, 817–836.

Williams, D. A. 1987. Generalized Linear Models Diagnostics Using the Deviance and Single Case Deletions. *Applied Statistics*, 36, 181–191.

Wolfinger, R., R. Tobias, and J. Sall. 1994. Computing Gaussian likelihoods and their derivatives for general linear mixed models. *SIAM Journal on Scientific Computing*, 15:6, 1294–1310.

Wolfinger, R., and M. O'Connell. 1993. Generalized Linear Mixed Models: A Pseudo-Likelihood Approach. *Journal of Statistical Computation and Simulation*, 4, 233–243.

Zeger, S. L., and K. Y. Liang. 1986. Longitudinal Data Analysis for Discrete and Continuous Outcomes. *Biometrics*, 42, 121–130.

Zhang, T., R. Ramakrishnon, and M. Livny. 1996. BIRCH: An efficient data clustering method for very large databases. In: *Proceedings of the ACM SIGMOD Conference on Management of Data,* Montreal, Canada: ACM, 103–114.

Index

absolute confidence difference to prior
  Apriori evaluation measure, 10
accuracy
  Binary Classifier node, 49
  neural networks algorithms, 301
  Pass, Stream, Merge algorithms, 134
activation functions
  multilayer perceptron algorithms, 292
AdaBoost
  boosting algorithms, 128
adaptive boosting
  boosting algorithms, 128
adjacency lattice
  in sequence rules, 336
adjusted propensities algorithms, 1
adjusted R-square
  in regression, 265
advanced output
  in factor analysis/PCA, 152
  in logistic regression, 273
  in regression, 265
AICC
  linear modeling algorithms, 256
Akaike information criterion
  generalized linear models algorithms, 181, 199
allow splitting of merged categories (CHAID), 76
alpha factoring
  in factor analysis/PCA, 143
anomaly detection
  blank handling, 7
  generated model, 7
  overview, 3
  predicted values, 7
  scoring, 7
anomaly index, 6
Apriori
  blank handling, 11–12
  confidence (predicted values), 12
  deriving rules, 9
  evaluation measures, 10
  frequent itemsets, 9
  generated model, 12
  generating rules, 10
  items and itemsets, 9
  maximum number of antecedents, 11
  maximum number of rules, 11
  minimum rules support/confidence, 11
  only true values for flags, 11
  options, 11
  overview, 9
  predicted values, 12
  ruleset evaluation options, 12
area under curve
  Binary Classifier node, 49
association rules, 341
  Apriori, 9
  Carma, 53
  sequence rules, 333
auto-clustering
  in TwoStep clustering, 399
automated data preparation algorithms, 13
  bivariate statistics collection, 22
  categorical variable handling, 25
  checkpoint, 17
  continuous variable handling, 31
  date/time handling, 14
  discretization of continuous predictors, 35
  feature construction, 32
  feature selection, 32
  measurement level recasting, 17
  missing values, 19
  notation, 13
  outliers, 18
  predictive power, 35
  principal component analysis, 33
  references, 36
  supervised binning, 32
  supervised merge, 26
  target handling, 21
  transformations, 20
  univariate statistics collection, 15
  unsupervised merge, 30
  variable screening, 17
automatic field selection
  regression, 263

backward elimination
  multinomial logistic regression algorithms, 275
backward field selection
  in regression, 265
backward stepwise
  multinomial logistic regression algorithms, 274
bagging algorithms, 125–126
  accuracy, 127
  diversity, 127
  notation, 125
  references, 130
Bayes Information Criterion (BIC)
  in TwoStep clustering, 399
Bayesian information criterion
  generalized linear models algorithms, 181, 199
Bayesian network algorithms, 37
  binning, 38
  blank handling, 47
  feature selection, 38
  Markov blanket algorithms, 43, 45–47
  notation, 37
  scoring, 47
  tree augmented naïve Bayes (TAN) models, 40–43
  variable types, 38
best subsets selection
  linear modeling algorithms, 253

binary classifier comparison metrics, 49
binning
  automatic binning in BN models, 38
  CHAID predictors, 74
binomial logistic regression
  algorithms, 279
BIRCH algorithm
  in TwoStep clustering, 397
blank handling
  Apriori, 11–12
  Carma, 55, 57
  Cox regression algorithms, 102
  in anomaly detection, 7
  in Bayesian network algorithms, 47
  in C&RT, 61, 71
  in CHAID, 81, 85
  in Decision List algorithm, 111
  in discriminant analysis, 121, 123
  in factor analysis/PCA, 151–152
  in k-means, 230
  in k-means clustering, 232
  in Kohonen models, 244–245
  in logistic regression, 271, 279
  in nearest neighbor algorithms, 236
  in optimal binning algorithms, 309
  in QUEST, 319, 326
  in regression, 265–266
  in scoring Decision List models, 111
  in support vector machines (SVM), 377
  in TwoStep clustering, 400–401
  nearest neighbor algorithms, 238
blanks
  imputing missing values, 223
Bonferroni adjustment
  in CHAID tests, 80
boosting algorithms, 125
  accuracy, 130
  adaptive boosting (AdaBoost), 128
  notation, 125
  stagewise additive modeling (SAMME), 129
Borgelt, Christian, 9
Box-Cox transformation
  automated data preparation algorithms, 21

C&RT
  blank handling, 61, 71
  confidence values, 71
  finding splits, 60
  gain summary, 69
  Gini index, 62
  impurity measures, 62, 64
  least squared deviation index, 64
  misclassification costs, 66
  overview, 59
  predicted values, 70
  prior probabilities, 65
  profits, 65
  pruning, 66
  risk estimates, 68
  stopping rules, 65
  surrogate splitting, 61
  twoing index, 64
  weight fields, 59
C5.0, 51
  scoring, 51
Carma
  blank handling, 55, 57
  confidence (predicted values), 56
  deriving rules, 53
  exclude rules with multiple consequents, 55
  frequent itemsets, 53
  generated model, 56
  generating rules, 54
  maximum rule size, 55
  minimum rules support/confidence, 55
  options, 55
  overview, 53
  predicted values, 56
  pruning value, 55
  ruleset evaluation options, 56
Carma (sequence rules algorithm), 337
case weights, 60, 74
CF (cluster feature) tree
  TwoStep clustering, 397
CHAID
  binning of continuous predictors, 74
  blank handling, 81, 85
  Bonferroni adjustment, 80
  chi-squared tests, 78
  compared to other methods, 73
  confidence values, 85
  costs, 82
  Exhaustive CHAID, 73
  expected frequencies, 78
  gain summary, 83
  merging categories, 76
  predicted values, 84
  profits, 82
  risk estimates, 82
  row effects chi-squared test, 79
  score values, 82
  splitting nodes, 77
  statistical tests, 77–80
  stopping rules, 81
  weight fields, 73
Chebychev distance
  in Kohonen models, 243
chi-square
  generalized linear models algorithms, 177
chi-square test
  in QUEST, 317
class entropy
  optimal binning algorithms, 305

class information entropy
  optimal binning algorithms, 306
cluster assignment
  in k-means, 229
cluster evaluation algorithms, 87
  goodness measures, 87
  notation, 87
  predictor importance, 89
  references, 91
  Silhouette coefficient, 89
  sum of squares between, 89
  sum of squares error, 89
cluster feature tree
  TwoStep clustering, 397
cluster membership
  in k-means, 231
  in Kohonen models, 245
  in TwoStep clustering, 401
cluster proximities
  in k-means, 231
clustering
  k-means, 227
  TwoStep algorithm, 397
coefficients
  in factor analysis/PCA, 151
  in regression, 261
comparison metrics
  Binary Classifier node, 49
complete separation
  in logistic regression, 271
component extraction
  in factor analysis/PCA, 139
conditional statistic
  Cox regression algorithms, 98
confidence
  in Apriori, 10
  in C&RT models, 71
  in CHAID models, 85
  in QUEST models, 326
  in sequence rules, 340, 342
  neural networks algorithms, 302
confidence difference
  Apriori evaluation measure, 10
confidence ratio
  Apriori evaluation measure, 11
confidence values
  rulesets, 12, 56
consistent AIC
  generalized linear models algorithms, 181
convergence criteria
  logistic regression, 271
Cook's distance
  linear modeling algorithms, 258
  logistic regression algorithms, 288
corrected Akaike information criterion (AICC)
  linear modeling algorithms, 256
costs
  in C&RT, 66
  in CHAID, 82
  in QUEST, 322
Cox and Snell R-square
  in logistic regression, 273
Cox regression
  blank handling, 102
Cox regression algorithms, 93
  baseline function estimation, 96
  blank handling, 102
  output statistics, 99
  plots, 101
  regression coefficient estimation, 94
  stepwise selection, 97
cross-entropy error
  multilayer perceptron algorithms, 292

data aggregation
  in logistic regression, 268
Decision List algorithm, 105
  blank handling, 111
  blank handling in scoring, 111
  confidence intervals, 110
  coverage, 111
  decision rule algorithm, 107–108
  decision rule split algorithm, 108–109
  frequency, 111
  primary algorithm, 106
  probability, 111
  scoring, 111
  secondary measures, 111
  terminology, 105
deviance
  generalized linear models algorithms, 178
  logistic regression algorithms, 287
deviance goodness-of-fit measure
  in logistic regression, 273
DfBeta
  logistic regression algorithms, 288
difference of confidence quotient to 1
  Apriori evaluation measure, 11
direct oblimin rotation
  factor analysis/PCA, 147
discretization
  see binning, 74
discriminant analysis
  blank handling, 121
discriminant analysis algorithms, 113
  basic statistics, 113
  blank handling, 123
  canonical discriminant functions, 118
  classification, 121
  classification functions, 117
  cross-validation, 122
  notation, 113
  references, 123

variable selection, 114
distances
in k-means, 229, 231
in Kohonen models, 243
in TwoStep clustering, 398
diversity
Pass, Stream, Merge algorithms, 134
dummy coding
in logistic regression, 267

encoding value for sets
in k-means, 228, 231
ensembles algorithms, 125
ensembling model scores, 136
equamax rotation
in factor analysis/PCA, 145
error backpropagation
multilayer perceptron algorithms, 295
estimated marginal means
generalized linear mixed models algorithms, 200
eta decay
in Kohonen models, 244
evaluation measures
in Apriori, 10
Exhaustive CHAID
merging categories, 76
see CHAID, 73
expected frequencies
CHAID tests, 78
in CHAID tests, 80

F-test
in CHAID, 77
factor analysis/PCA
advanced output, 152
alpha factoring, 143
blank handling, 151–152
chi-square statistics, 142
direct oblimin rotation, 147
equamax rotation, 145
factor score coefficients, 151
factor scores, 152
factor/component extraction, 139
generalized least squares extraction, 142
image factoring, 144
maximum likelihood extraction, 140
overview, 139
principal axis factoring, 140
principal components analysis (PCA), 139
promax rotation, 150
quartimax rotation, 145
rotations, 145
unweighted least squares extraction, 142
varimax rotation, 145
factor equations
in factor analysis/PCA, 151
factor extraction
in factor analysis/PCA, 139
factor score coefficients
in factor analysis/PCA, 151
factor scores
in factor analysis/PCA, 152
feature selection
in Bayesian network algorithms, 38
field encoding
encoding of flag fields, 229, 242
encoding of symbolic fields, 227, 241
Kohonen models, 241
scaling of range fields, 227, 241
finite sample corrected AIC
generalized linear models algorithms, 181, 199
flag fields
encoding, 229, 242
forward entry
multinomial logistic regression algorithms, 274
forward field selection
in regression, 264
forward stepwise
multinomial logistic regression algorithms, 274
forward stepwise selection
linear modeling algorithms, 250
frequency weights, 59, 74, 315
frequent itemsets
in Apriori, 9
in Carma, 53
frequent sequences, 335

gain summary
in C&RT, 69
in CHAID, 83
in QUEST, 325
GDI
see Group Deviation Index, 5
generalized least squares
in factor analysis/PCA, 142
generalized linear mixed models algorithms, 187, 204
estimated marginal means, 200
fixed effects transformation, 191
goodness of fit statistics, 199
link function, 189
model, 188
model for nominal multinomial, 207
model for ordinal multinomial, 214
multiple comparisons, 202
notation, 187, 206
references, 220
scale parameter, 190
tests of fixed effects, 199
generalized linear models algorithms, 163
chi-square statistic, 177
default tests of model effects, 182
estimation, 169
goodness of fit, 178

link function, 168
model, 163
model fit test, 182
model testing, 177
notation, 163
probability distribution, 164
references, 184
scoring, 183
generalized logit model
in logistic regression, 269
Gini index
in C&RT, 62
goodness of fit
generalized linear models algorithms, 178
goodness-of-fit measures
in logistic regression, 273
gradient descent
multilayer perceptron algorithms, 295
Group Deviation Index, 5

hazard plots
Cox regression algorithms, 102
hierarchical clustering
in TwoStep clustering, 398
Hosmer-Lemeshow goodness-of-fit statistic
logistic regression algorithms, 286
hyperbolic tangent activation function
multilayer perceptron algorithms, 292
hyperbolic tangent kernel function (SVM), 371

identity activation function
multilayer perceptron algorithms, 292
image factoring
in factor analysis/PCA, 144
impurity measures (C&RT), 62, 64
imputing missing values, 223
indicator coding, 227, 241
information criteria
generalized linear models algorithms, 181, 199
information difference
Apriori evaluation measure, 11
information gain
optimal binning algorithms, 306
initial cluster centers
in k-means, 229
items
in Apriori, 9
itemsets
in Apriori, 9
in sequence rules, 333

k-means
assigning records to clusters, 229
blank handling, 230
cluster centers, 229–230, 232
cluster proximities, 231
distance field (predicted values), 231
distance measure, 229
encoding value for sets, 228, 231
error tolerance, 231
field encoding, 227
initial cluster centers, 229
iterating, 229
maximum iterations, 230
overview, 227
predicted cluster membership, 231
Kohonen models
blank handling, 244–245
cluster centers, 242
cluster membership, 245
distances, 243
learning rate (eta), 242, 244
model parameters, 242
neighborhoods, 243
overview, 241
random seed, 244
scoring, 245
stopping criteria, 244
weights, 242–243

Lagrange multiplier test
generalized linear models algorithms, 177
learning rate (eta)
in Kohonen models, 242, 244
least significant difference
generalized linear mixed models algorithms, 203
least squared deviation index
in C&RT, 64
leave-one-out classification
discriminant analysis algorithms, 122
legal notices, 403
length
of sequences, 334
Levene’s test
in QUEST, 318
leverage
linear modeling algorithms, 258
logistic regression algorithms, 288
lift
Binary Classifier node, 49
likelihood ratio chi-squared test
in CHAID, 78
likelihood ratio statistic
Cox regression algorithms, 98
likelihood-based distance measure
in TwoStep clustering, 398
linear kernel function (SVM), 371
linear modeling algorithms, 247
coefficients, 256
diagnostics, 258
least squares estimation, 248
model, 248
model evaluation, 255

model selection, 249–250, 253
notation, 247
predictor importance, 259
references, 259
scoring, 258
linear regression, 261
link function
generalized linear mixed models algorithms, 189
generalized linear models algorithms, 168
log-likelihood
in logistic regression, 269, 272
log-minus-log plots
Cox regression algorithms, 102
logistic regression
advanced output, 273
binomial logistic regression algorithms, 279
blank handling, 271, 279
checking for separation, 271
convergence criteria, 271
Cox and Snell R-square, 273
data aggregation, 268
field encoding, 267
generalized logit model, 269
goodness-of-fit measures, 273
log-likelihood, 269, 272
maximum likelihood estimation, 270
McFadden R-square, 273
model chi-square, 272
Nagelkerke R-square, 273
notation, 279
overview, 267
parameter start values, 270
predicted probability, 279
predicted values, 279
pseudo R-square measures, 273
reference category, 267
stepping, 270
logistic regression algorithms
maximum likelihood estimates, 280
model, 280
notation, 279
output statistics, 284
stepwise variable selection, 281
logit residuals
logistic regression algorithms, 288
logits
in logistic regression, 269

Markov blanket Bayesian network models
adjustment for small cell counts, 47
algorithms, 43, 45
chi-square independence test, 44
conditional independence tests, 43
deriving the Markov blanket, 46
G^2 test, 44
likelihood ratio test, 44
parameter learning, 46
posterior estimation, 46
structure learning algorithm, 45
maximal sequences, 334
maximum likelihood
in factor analysis/PCA, 140
in logistic regression, 270
maximum profit
Binary Classifier node, 49
maximum profit occurs in %
Binary Classifier node, 49
McFadden R-square
in logistic regression, 273
MDLP
optimal binning algorithms, 305
merging categories
CHAID, 76
Exhaustive CHAID, 76
min-max transformation
automated data preparation algorithms, 20
misclassification costs
in C&RT, 66
in QUEST, 322
missing values
imputing, 223
model chi-square
in logistic regression, 272
model information
Cox regression algorithms, 99
model updates
multilayer perceptron algorithms, 298
multilayer perceptron algorithms, 291
activation functions, 292
architecture, 291
error functions, 292
expert architecture selection, 293
model updates, 298
notation, 291
training, 294
multinomial logistic regression, 267
multinomial logistic regression algorithms
stepwise variable selection, 274

Nagelkerke R-square
in logistic regression, 273
naive bayes
see self-learning response models, 327
Naive Bayes algorithms, 327
model, 327
notation, 327
nearest neighbor algorithms, 233
blank handling, 236, 238
distance metric, 234
feature selection, 235
feature weights, 234
k selection, 235
notation, 233
output statistics, 237

preprocessing, 233
references, 239
scoring, 238
training, 234
neighborhoods
in Kohonen models, 242–243
network architecture
multilayer perceptron algorithms, 291
radial basis function algorithms, 299
neural networks algorithms, 291
confidence, 302
missing values, 301
multilayer perceptron (MLP), 291
output statistics, 301
radial basis function (RBF), 298
references, 302
simplemax, 302
nominal regression, 267
normalized chi-square
Apriori evaluation measure, 11
number of clusters
auto-selecting in TwoStep clustering, 399

optimal binning algorithms, 305
blank handling, 309
class entropy, 305
class information entropy, 306
hybrid MDLP, 308
information gain, 306
MDLP, 305
merging bins, 309
notation, 305
references, 310
ordinal fields
in CHAID, 79
ordinary least squares regression, 261
outlier handling
in TwoStep clustering, 400
overall accuracy
Binary Classifier node, 49
overdispersion
generalized linear models algorithms, 180

Pass, Stream, Merge algorithms, 130
accuracy, 134
adaptive feature selection, 132
category balancing, 133
diversity, 134
Merge, 132
Pass, 131
scoring, 135
Stream, 132
Pearson chi-square
generalized linear models algorithms, 179
Pearson chi-squared test
in CHAID, 78
Pearson goodness-of-fit measure
in logistic regression, 273
polynomial kernel function (SVM), 371
pre-clustering
in TwoStep clustering, 397
predicted group
logistic regression algorithms, 288
predicted values
anomaly detection, 7
generalized linear models algorithms, 183
rulesets, 12, 56
predictive power
automated data preparation algorithms, 35
predictor importance
cluster evaluation algorithms, 89
linear modeling algorithms, 259
predictor importance algorithms, 311
notation, 311
references, 314
variance based method, 311
principal axis factoring
in factor analysis/PCA, 140
principal component analysis
automated data preparation algorithms, 33
principal components analysis (PCA), 139
priors
in C&RT, 65
in QUEST, 322
profits
in C&RT, 65
in CHAID, 82
in QUEST, 321
promax rotation
in factor analysis/PCA, 150
pruning
in C&RT, 66
in QUEST, 323

quartimax rotation
in factor analysis/PCA, 145
quasi-complete separation
in logistic regression, 271
QUEST
blank handling, 319, 326
chi-square test, 317
confidence values, 326
F-test, 317
finding splits, 316
gain summary, 325
Levene's test, 318
misclassification costs, 322
overview, 315
predicted values, 325
prior probabilities, 322
profits, 321
pruning, 323
risk estimates, 324

stopping rules, 321
surrogate splitting, 319
weight fields, 315

R-square
in regression, 265
radial basis function algorithms, 298
architecture, 299
automatic selection of number of basis functions, 300
center and width for basis functions, 300
model updates, 301
notation, 298
training, 299
random seed
in Kohonen models, 244
range fields
rescaling, 227, 241
RBF kernel function (SVM), 371
regression
adjusted R-square, 265
advanced output, 265
automatic field selection, 263
backward field selection, 265
blank handling, 265–266
forward field selection, 264
model parameters, 261
notation, 261
overview, 261
predicted values, 265
R-square, 265
stepwise field selection, 264
replacing missing values, 223
risk estimates
in C&RT, 68
in CHAID, 82
in QUEST, 324
row effects model
in CHAID tests, 79
RuleQuest Research, 51
rulesets
confidence (predicted values), 12, 56
predicted values, 12, 56

SAMME
boosting algorithms, 129
Satterthwaite approximation
generalized linear mixed models algorithms, 203
scale parameter
generalized linear mixed models algorithms, 190
scaled conjugate gradient
multilayer perceptron algorithms, 296
scaled deviance
generalized linear models algorithms, 179
scaled Pearson chi-square
generalized linear models algorithms, 179
score coefficients
in factor analysis/PCA, 151
score statistic
Cox regression algorithms, 98
score test
multinomial logistic regression algorithms, 276
score values (CHAID), 82
scoring
Decision List algorithm, 111
in anomaly detection, 7
self-learning response model algorithms, 327
information measure, 331
model assessment, 328
predictor importance, 330–331
scoring, 329
updating the model, 329
separation
checking for in logistic regression, 271
sequence rules, 341
adjacency lattice, 336
antecedents, 335
blank handling, 340, 342
Carma algorithm, 337
confidence, 340, 342
consequents, 335
frequent sequences, 335
gap, 335
itemsets, 333
length of sequences, 334
maximal sequences, 334
overview, 333
predictions, 341
sequential patterns, 339
size of sequences, 334
subsequences, 334
support, 334
timestamp tolerance, 335
transactions, 333
sequences
in sequence rules, 333
sequential Bonferroni
generalized linear mixed models algorithms, 203
sequential minimal optimization (SMO) algorithm
support vector machines (SVM), 371
sequential sidak
generalized linear mixed models algorithms, 203
set encoding value
in k-means, 228
Silhouette coefficient
cluster evaluation algorithms, 89
simplemax
neural network confidence, 302
simulation
algorithms, 343
simulation algorithms, 343, 361
beta distribution fitting, 349
binomial distribution fitting, 344
categorical distribution fitting, 345
contribution to variance measure of sensitivity, 364

correlation measure of sensitivity, 364
distribution fitting, 343
exponential distribution fitting, 349
gamma distribution fitting, 350
generating correlated data, 361
goodness of fit measures, 352
goodness of fit measures: Anderson-Darling test , 355
goodness of fit measures: continuous distributions, 353
goodness of fit measures: discrete distributions, 352
goodness of fit measures: Kolmogorov-Smirnov test , 357
lognormal distribution fitting, 348
negative binomial distribution fitting, 345
normal distribution fitting, 348
one-at-a-time measure of sensitivity, 364
Poisson distribution fitting, 345
references, 360, 365
sensitivity measures, 363
tornado charts, 363
triangular distribution fitting, 347
uniform distribution fitting, 348
Weibull distribution fitting, 351
size
of sequences, 334
softmax activation function
multilayer perceptron algorithms, 292
splitting
of merged categories (CHAID), 76
splitting nodes
CHAID, 77
stagewise additive modeling
boosting algorithms, 129
standardized residuals
logistic regression algorithms, 288
stepping
in logistic regression, 270
stepwise field selection
in regression, 264
stepwise selection
Cox regression algorithms, 97
stopping rules
in C&RT, 65
in CHAID, 81
in QUEST, 321
multilayer perceptron algorithms, 297
studentized residuals
linear modeling algorithms, 258
logistic regression algorithms, 288
subpopulations, 268
subsequences, 334
sum of squares between
cluster evaluation algorithms, 89
sum of squares error
cluster evaluation algorithms, 89
multilayer perceptron algorithms, 292
support
sequence rules, 334
support vector machines (SVM), 367
ε-Support Vector Regression (ε-SVR), 368
algorithm notation, 367
blank handling, 377
C-support vector classification, 368
decision function constant, 370
fast training algorithm, 375
gradient reconstruction, 372
kernel functions, 371
model building algorithm, 370
parallel optimization, 375
predicted probabilities, 377
predictions, 377
queue method, 376
scoring, 377
sequential minimal optimization (SMO) algorithm, 371
sequential optimization, 376
shrinking, 372
SMO decomposition, 374
solving quadratic problems, 369
subset selection, 376
types of SVM models, 367
unbalanced data, 373
variable scaling, 370
working set selection, 371
surrogate splitting
in C&RT, 61
in QUEST, 319
survival plots
Cox regression algorithms, 102
symbolic fields
recoding, 227, 241

Time Series algorithms, 379
additive outliers, 388
all models expert model, 394
AO (additive outliers), 388
AO patch (AOP), 389
AO patch outliers, 390
AOP, 389
ARIMA and transfer function models, 382
ARIMA expert model, 394
Brown’s exponential smoothing, 380
CLS, 384
conditional least squares (CLS) method, 384
damped-trend exponential smoothing, 380
definitions of outliers, 388
detection of outliers, 390
diagnostic statistics, 387
error variance, 385
estimating the effects of an outlier, 390
estimation and forecasting of ARIMA/TF, 384
estimation and forecasting of exponential smoothing, 382
expert modeling, 393
exponential smoothing expert model, 393
exponential smoothing models, 379

goodness-of-fit statistics, 391
Holt’s exponential smoothing, 380
initialization of ARIMA/TF, 385
initialization of exponential smoothing, 382
innovational outliers, 388
IO (innovational outliers), 388
level shift, 388
Ljung-Box statistic, 387
local trend, 389
LS (level shift), 388
LT (local trend), 389
maximum absolute error, 392
maximum absolute percent error, 392
maximum likelihood (ML) method, 384
mean absolute error, 392
mean absolute percent error, 392
mean squared error, 392
ML, 384
models, 379
multivariate series, 394
non-AO patch deterministic outliers, 390
normalized bayesian information criterion, 392
notation, 379, 388
outlier detection in time series analysis, 387
outliers summary, 389
R-squared, 392
references, 396
SA (seasonal additive), 388
seasonal additive, 388
simple exponential smoothing, 379
simple seasonal exponential smoothing, 381
stationary R-squared, 392
TC (temporary/transient change), 388
temporary/transient change, 388
transfer function calculation, 386
transfer function expert model, 395
univariate series, 393
Winters’ additive exponential smoothing, 381
Winters’ exponential smoothing, 381
timestamp tolerance
in sequence rules, 335
trademarks, 404
transactions
in sequence rules, 333
tree augmented naïve Bayes (TAN) models
adjustment for small cell counts, 43
algorithms, 40
learning algorithm, 41
parameter learning, 42
posterior estimation, 43
structure learning, 42
twoing index
in C&RT, 64
TwoStep clustering
auto-clustering, 399
blank handling, 400–401
cluster feature tree, 397
clustering step, 398
distance measure, 398
model parameters, 397
outlier handling, 400
overview, 397
pre-clustering step, 397
predicted values, 401

unweighted least squares
in factor analysis/PCA, 142
updating
self-learning response models, 329

variable contribution measure
in anomaly detection, 6
Variable Deviation Index, 5
varimax rotation
in factor analysis/PCA, 145
VDI
see Variable Deviation Index, 5

Wald statistic
Cox regression algorithms, 98
weight fields
CHAID, 73
weights
in Kohonen models, 242–243

z-score transformation
automated data preparation algorithms, 20

Generalized Linear Engine (GLE) Algorithm

1. Introduction – Phase I

Generalized linear models (GZLMs) have been commonly used analytical tools for different types of data for quite some time because they cover not only widely used statistical models, such as linear regression for normally distributed targets, logistic models for binary data, and log linear model for count data, but also many useful statistical models via its very general model formulation. Since those models are under the independence assumption, we have a new "Generalized Linear Engine" (GLE) to build them for large and distributed data and run within Analytic Engine (AE).

GLE Phase I is mainly to replace GENLIN functionality in a Big Data situation in addition to adding the nominal multinomial model. Section 2 describes the model. Section 3 describes parameter estimation. Inference and model summary is given in Section 4. Scoring is presented in the last section.

2. Model

There are two subsections under the model section: (1) notations and (2) model formation. Then for the model formation subsection, four sub-subsections are furthered derived: (1) probability distribution; (2) link function; (3) combination of probability distribution and link function; (4) data transformation.

2.1. Notations

n	Number of distinct records in the dataset. It is an integer and $n \geq 1$.
p	Number of parameters (including the constant, if exists) in the model. It is an integer and $p \geq 1$.
p_x	Number of non-redundant columns in the design matrix. It is an integer and $1 \leq p_x \leq p$.
$\boldsymbol{y}$	$n \times 1$ vector of target variable consists of $y_i, i = 1, \ldots, n$.
$\boldsymbol{r}$	$n \times 1$ vector of event variable for binomial distribution. It is usually the number of successes or the number of 1's. All elements are non-negative integers.
$\boldsymbol{m}$	$n \times 1$ vector of trial variable for binomial distribution. All elements are positive integers and m_i

	$\geq r_i,\ i=1,\ldots,n$.
$\boldsymbol{\mu}$	$n \times 1$ vector of expectation of target variable.
$\boldsymbol{\eta}$	$n \times 1$ vector of linear predictor.
$\boldsymbol{X}$	$n \times p$ design matrix. The rows represent the records and the columns represent the parameters. The i^{th} row is $\boldsymbol{x}_i^{\mathrm{T}} = \left(x_{i1},\ldots,x_{ip}\right)$, where superscript T means transpose of a matrix or vector, $i=1,\ldots,n$ with $x_{i1}=1$ if model has an intercept.
$\boldsymbol{O}$	$n \times 1$ vector of offset variable. This variable can't be the dependent variable (***y***) or one of the predictor variables (***X***). Also this variable can't be a categorical variable (factor).
$\boldsymbol{\beta}$	$p \times 1$ vector of unknown parameters. The first element in $\boldsymbol{\beta}$ is the intercept, if there is one.
$\boldsymbol{\omega}$	$n \times 1$ vector of scale weight variable. The elements don't have to be integers. If an element is less than or equal to 0 or missing, the corresponding record is not used.
$\boldsymbol{f}$	$n \times 1$ vector of frequency count variable. Non-integer elements are treated by rounding the value to the nearest integer. For values less than 0.5 or missing, the corresponding records are not used.
N	Effective sample size. $N=\sum_{i=1}^{n} f_i$. If frequency count variable ***f*** is not used, $N = n$.

2.2. Model formation

A GZLM of the target ***y*** with predictor variables ***X*** and offset variable ***O*** has the form

$$\boldsymbol{\eta} = g(\mathrm{E}(\boldsymbol{y})) = \boldsymbol{X\beta} + \boldsymbol{O}, \quad \boldsymbol{y} \sim F,$$

where $\boldsymbol{\eta}$ is the linear predictor; ***O*** is an offset variable with a constant coefficient of 1 for each observation; $g(.)$ is the monotonic differentiable link function which states how the mean of ***y***, $\mathrm{E}(\boldsymbol{y}) = \boldsymbol{\mu}$, is related to the linear predictor $\boldsymbol{\eta}$; F is the target probability distribution. Choosing different combinations of a proper probability distribution and a link function can result in different models. Some combinations are well known models and have been provided in different SPSS procedures. The following table lists these combinations and corresponding SPSS procedures.

Table 1: Distribution, Link Function and Corresponding SPSS Procedure

Distribution	Link function	Model	SPSS procedure
Normal	Identity	Linear regression model	GLM, REGRESSION
Binomial	Logit	Logistic regression model	LOGISTIC REGRESSION

Poisson	Log	Log- linear model	GENLOG
Nominal multinomial	Generalized logit	Generalized logistic regression model	NUMREG
Ordinal multinomial	Cumulative logit	Ordinal proportional-odds model	PLUM

In addition, GZLM also assumes y_i are independent for record $i = 1, \ldots , n$, then the model becomes

$$\eta_i = g(\mu_i) = \boldsymbol{x}_{\boldsymbol{i}}^{\mathrm{T}}\boldsymbol{\beta} + o_i, \quad y_i \sim F.$$

<u>Notes</u>:

- To improve numerical stability, the $\boldsymbol{X}$ matrix will be transformed, see Section 2.2.4 for details. Note that the computation of transformation can be implemented in map/reduce environment.
- The $\boldsymbol{X}$ matrix can be any combination of continuous variables (covariates), categorical variables (factors) and interactions. The parameterization of design matrix $\boldsymbol{X}$ is the same as in GLM procedure. See Lam (1995a) for further details on the model parameterization.

  Due to use of over-parameterized model where there is a separate parameter for every factor effect level occurring in the data, the columns of the design matrix $\boldsymbol{X}$ are often dependent. Collinearities among continuous variables in the data can also occur. To establish the dependencies in the design matrix, columns of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\Psi}\boldsymbol{X},$ where $\boldsymbol{\Psi} = \mathrm{diag}\left(f_1\omega_1, \ldots f_n\omega_n \right)$, are examined by using the sweep operator. When a column is found to be dependent on previous columns, the corresponding parameter is treated as redundant. The solution for redundant parameters is fixed at zero. Details of the sweep operator employed can be found in Lam (1995b).
- When the target variable is in a binary format which can be character or numeric, such as the form of male/female, 1/2, a/b, its values will be transformed to 0 and 1 with 1 as typically representing a success or some other positive result. In this document, we assume that $\boldsymbol{y}$ has been transformed to 0/1 values and we always model the probability of success, i.e., Prob($\boldsymbol{y}$ = 1). Which original value should be transformed to 0 or 1 depends on what the reference category is. If the reference category is the last value, then the first category represents a success and we are modeling the probability of it. For example, if the reference category is the last value, "male", "2" and "b" in "male/female", "1/2" and "a/b" binary forms are the last values and would be transformed to 0, and "female", "1" and "a" would be transformed to 1 as we model the probability of them,

respectively. However, one way to change to model the probability of “male”, “2” and “b” instead is to specify the reference category to be the first value. Note if original binary format is 0/1 and the reference category is the last value, then 0 would be transformed to 1 and 1 to 0.

- For the binomial distribution and the target is a number of events (***r***) occurring in a set of trials (***m***), in this document, we assume that ***y*** is the binomial proportion, i.e., ***y*** = ***r/m***.

GLE would also include ordinal and nominal multinomial distributions. However, since the model form is not the same as that of the above traditional generalized linear models, we include them in Appendix A and Appendix B, respectively.

2.2.1. Probability distribution

GLE will include 9 distributions which include 3 continuous ones: normal, inverse Gaussian, gamma; 5 discrete ones: binomial, Poisson, negative binomial, ordinal multinomial, nominal multinomial; and 1 mixed distribution: Tweedie.

Table 2 lists distribution of y, corresponding range of y, the variance function ($V(\mu)$), the variance of y (Var(y)) and the 1st derivative of the variance function ($V'(\mu)$), which will be used later. Again ordinal multinomial and nominal multinomial would be handled in Appendices A and B, respectively.

Table 2: Distribution, Range and Variance of the Target, Variance Function and Its 1st Derivative

Distribution	Range of y	$V(\mu)$	Var(y)	$V'(\mu)$
Normal	$(-\propto, \propto)$	1	ϕ	0
Inverse Gaussian	$(0, \propto)$	μ^3	$\phi\mu^3$	$3\mu^2$
Gamma	$(0, \propto)$	μ^2	$\phi\mu^2$	2μ
Negative binomial	$0(1)\propto$	$\mu + k\mu^2$	$\mu + k\mu^2$	$1+2k\mu$
Poisson	$0(1)\propto$	μ	μ	1
Binomial(m)	$\frac{0(1)m}{m}$	$\mu(1-\mu)$	$\frac{\mu(1-\mu)}{m}$	$1-2\mu$
Tweedie	$[0, \propto)$	μ^q	$\phi\mu^q$	$q\mu^{q-1}$

Notes:

- 0(1)*z* means the range is from 0 to *z* with increment of 1 (i.e. 0, 1, 2, …, *z*).

- For the binomial distribution, the binomial trial variable *m* is considered as a part of the weight variable ω.

- If a weight variable ω is included, ϕ is replaced by ϕ/ω.

- For the negative binomial distribution, there is an ancillary parameter (*k*) and there are two ways to handle it:

  1. It can be estimated with $\boldsymbol{\beta}$ jointly by the maximum likelihood (ML) method.

  2. It can be set to a fixed positive value.

  In general, only when *k* is known, the target *y* with a negative binomial distribution is a generalized linear model. Furthermore, the default for *k* should be the fixed value provided by the user because, according to McCullagh and Nelder (1989), the interpretation of using negative binomial distribution and canonical link function might be problematical as it makes the linear predictor a function of a parameter of the variance function.

  Typical values of *k* range between 0.01 and 2, but we will also allow $k = 0$, which reduces the negative binomial distribution to the Poisson distribution. When $k = 0$, we simply apply the Poisson distribution to do the estimation. When $k = 1$, the negative binomial is the geometric distribution.

- The Tweedie's class of distributions includes discrete, continuous and mixed densities as long as $q \le 0$ or $q \ge 1$, where *q* is the exponent in the variance function, μ^q. Special cases include the normal ($q = 0$), Poisson ($q = 1$), gamma ($q = 2$) and inverse Gaussian ($q = 3$). Except those special cases, the Tweedie distributions with other values of *q* cannot be written in closed form, and hence evaluation of the density is difficult. Here, we only consider the Tweedie distributions for $1 < q < 2$ which can be represented as Poisson mixtures of gamma distributions and are mixed distributions with mass at zero and with support on the non-negative real values. These distributions have been called "compound Poisson", "compound gamma" and "Poisson-gamma" distributions, but we will still call "Tweedie". Here, the *q* value is set a fixed value. Thus, the user has to give a $q \in (1, 2)$.

- From the expressions for $V(\mu)$ and Var(*y*), continuous distributions (normal, inverse Gaussian and gamma) and Tweedie distributions for $1 < q < 2$ include the scale parameter ϕ which can be used to scale the relationship of the variance and mean (Var(y) and μ). Since it is usually unknown, there are three ways to fit the scale parameter ϕ:

1. It can be estimated with $\boldsymbol{\beta}$ jointly by ML method.
2. It can be set to a fixed positive value.
3. It can be specified by the deviance or Pearson chi-square (see Section 4.3.3).

On the other hand, discrete distributions (binomial, Poisson, negative binomial) do not have this extra parameter (it is theoretically equal to one). Because of it, the variance of y might not be equal to the nominal variance in practice (especially for Poisson and binomial because negative binomial has an ancillary parameter k). A simple way to adjust this situation is to allow the variance of y of discrete distributions to have the scale parameter ϕ as well. That's why ϕ/ω is included in the log likelihood function of each discrete distribution below, but, unlike ϕ for continuous distributions, it can't be estimated by ML method. So for discrete distributions, there are two ways to obtain the value of ϕ:

1. It can be set to a fixed positive value.
2. It can be specified by the deviance or Pearson chi-square.

To ensure the data fit the range of target $\boldsymbol{y}$ (or $\boldsymbol{r}$ and $\boldsymbol{m}$ for the binomial distribution) for the specified distribution, the following rules are enforced:

(a) For the gamma or inverse Gaussian distributions, values of $\boldsymbol{y}$ must be real and greater than zero. If a value of $\boldsymbol{y}$ is less than or equal to 0 or missing, the corresponding record is not used.

(b) For the negative binomial and Poisson distributions, values of $\boldsymbol{y}$ must be integer and non-negative. If a value of $\boldsymbol{y}$ is non-integer, less than 0 or missing, the corresponding record is not used.

(c) For the binomial distribution and if the target is in the form of a single variable, $\boldsymbol{y}$ must have only two distinct values. If $\boldsymbol{y}$ has more than two distinct values, then we stop the program and issue an error message, such as “The target variable has more than 2 levels. A binary target must have 2 levels.”

(d) For the binomial distribution and if the target is a number of events ($\boldsymbol{r}$) occurring in a set of trials ($\boldsymbol{m}$), values of $\boldsymbol{r}$ must be non-negative integers, values of $\boldsymbol{m}$ must be positive integers and $m_i \geq r_i$, $\forall$ i. If a value of $\boldsymbol{r}$ is not integer, less than 0, or missing, the corresponding record is not used. If a value of $\boldsymbol{m}$ is not integer, less than or equal to 0, less than the corresponding value of $\boldsymbol{r}$, or missing, the corresponding record is not used.

(e) For the Tweedie distributions, values of $\boldsymbol{y}$ must be zero or positive real. If a value of $\boldsymbol{y}$ is less than 0 or missing, the corresponding record is not used.

The ML method will be used to estimate $\boldsymbol{\beta}$ and possibly ϕ for continuous distributions and Tweedie distribution or k for negative binomial. The kernels of the log likelihood function (ℓ_k) and the full log likelihood function (ℓ), which will be used as the objective function for parameter estimation, are listed for each distribution in the following table. Using ℓ or ℓ_k won't affect the parameter estimation, but the selection will affect the calculation of information criteria in Section 4.3.4.

Table 3: The Log Likelihood Function for Probability Distribution

Distribution	ℓ_k and ℓ
Normal	$\ell_k = \sum_{i=1}^{n} -\frac{f_i}{2}\left\{\frac{\omega_i\left(y_i-\mu_i\right)^2}{\phi} + \ln\left(\frac{\phi}{\omega_i}\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
Inverse Gaussian	$\ell_k = \sum_{i=1}^{n} -\frac{f_i}{2}\left\{\frac{\omega_i\left(y_i-\mu_i\right)^2}{\phi y_i \mu_i^2} + \ln\left(\frac{\phi y_i^3}{\omega_i}\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
Gamma	$\ell_k = \sum_{i=1}^{n} f_i\left\{\frac{\omega_i}{\phi}\ln\left(\frac{\omega_i y_i}{\phi\mu_i}\right) - \frac{\omega_i y_i}{\phi\mu_i} - \ln\left(\Gamma\left(\frac{\omega_i}{\phi}\right)\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\left\{-\ln\left(y_i\right)\right\}$
Negative binomial	$\ell_k = \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\left\{y_i\ln\left(k\mu_i\right) - \left(y_i+1/k\right)\ln\left(1+k\mu_i\right) + \ln\left(\Gamma\left(y_i+1/k\right)\right) - \ln\left(\Gamma\left(1/k\right)\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\left\{-\ln\left(\Gamma\left(y_i+1\right)\right)\right\}$
Poisson	$\ell_k = \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\left\{y_i\ln\left(\mu_i\right) - \mu_i\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i\frac{\omega_i}{\phi}\left\{-\ln\left(y_i!\right)\right\}$

Binomial(*m*)	$\ell_k = \sum_{i=1}^{n} f_i \frac{\omega_i^*}{\phi}\left\{y_i \ln(\mu_i) + (1-y_i)\ln(1-\mu_i)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i \frac{\omega_i}{\phi}\left\{\ln\binom{m_i}{r_i}\right\}$, where $\binom{m_i}{r_i} = \frac{m_i!}{r_i!(m_i-r_i)!}$
Tweedie	$\ell_k = \sum_{i=1}^{n} f_i \left\{\ln(V_i) + \frac{\omega_i}{\phi}\left(\frac{y_i \mu_i^{1-q}}{(1-q)} - \frac{\mu_i^{2-q}}{(2-q)}\right)\right\}$ $\ell = \ell_k + \sum_{i=1}^{n} f_i \left\{-\ln(y_i)\right\}$ (note that the ∑ term won't include $y_i = 0$)

<u>Notes</u>:

- The computation of ℓ_k or ℓ can be implemented in map/reduce environment.
- When individual $y = 0$ for negative binomial, Poisson and Tweedie distributions and $y = 0$ or 1 for binomial distribution, separate value of the log likelihood is given. Let $\ell_{k,i}$ be the log likelihood value for individual record *i* when $y_i = 0$ for negative binomial, Poisson and Tweedie and 0/1 for binomial.

Distribution	$\ell_{k,i}$
Negative binomial	$-f_i \frac{\omega_i}{\phi} \frac{\ln(1+k\mu_i)}{k} \quad \text{if } y_i = 0$
Poisson	$-f_i \frac{\omega_i}{\phi} \mu_i \quad \text{if } y_i = 0$
Binomial(*m*)	$\begin{cases} f_i \frac{\omega_i}{\phi} \ln(1-\mu_i) & \text{if } y_i = 0 \\ f_i \frac{\omega_i}{\phi} \ln(\mu_i) & \text{if } y_i = 1 \end{cases}$
Tweedie	$-f_i \frac{\omega_i}{\phi} \frac{\mu_i^{2-q}}{(2-q)} \quad \text{if } y_i = 0$

Note that the full log likelihood for *i* is equal to the kernel of the log likelihood for *i*, i.e., $\ell_i = \ell_{k,i}$, for negative binomial, Poisson and Tweedie. However, for binomial with 0/1 binary target variable, they should be different (the full log likelihood has additional term. The full log likelihood, like deviance and Pearson chi-square, should be computed based on subpopulations. Please see Section 4.3.3.2 for details).

- $\Gamma(z)$ is a gamma function and $\ln(\Gamma(z))$ is a log-gamma function (the logarithm of the gamma function), evaluated at *z*. In general, $\ln(\Gamma(z))$ is calculated by using Sterling's formula, rather than first calculating the gamma function and then taking the natural logarithm because numerical calculation of $\Gamma(z)$ with large values of *z* may cause an overflow.
- For binomial distribution (***r***/***m***), the scale weight variable becomes $\omega_i^* = \omega_i m_i$ in ℓ_k, i.e., the binomial trials variable ***m*** is regarded as a part of weight. However, the scale weight in the extra term of ℓ is still ω_i.
- V_i in Tweedie distribution is an infinite series and the computational details are described in Appendix C.

2.2.2. Link function

Table 4 lists the link functions, inverse forms of them and ranges of μ for all distributions and Table 5 lists the 1st and 2nd derivatives for each link function in Table 4 which they will be used in Section 2.

Table 4: Link Function Name, Form, Inverse Form and Range of the Predicted Mean

Link function name	$\eta = g(\mu)$	Inverse $\mu = g^{-1}(\eta)$	Range of $\hat{\mu}$
Identity	μ	η	$\hat{\mu} \in \Re$
Log	$\ln(\mu)$	$\exp(\eta)$	$\hat{\mu} > 0$
Logit	$\ln\left(\frac{\mu}{1-\mu}\right)$	$\frac{\exp(\eta)}{1+\exp(\eta)}$	$\hat{\mu} \in (0, 1)$
Probit	$\Phi^{-1}(\mu)$, where $\Phi(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\xi} e^{-z^2/2} dz$	$\Phi(\eta)$	$\hat{\mu} \in (0, 1)$
Complementary log-log	$\ln(-\ln(1-\mu))$	$1-\exp(-\exp(\eta))$	$\hat{\mu} \in (0, 1)$

Power(α^*) $\begin{cases}\alpha \neq 0 \\ \alpha = 0\end{cases}$	$\begin{cases}\mu^{\alpha} \\ \ln(\mu)\end{cases}$	$\begin{cases}\eta^{1/\alpha} \\ \exp(\eta)\end{cases}$	$\begin{cases}\hat{\mu} \in \Re & \text{if } \alpha \text{ or } 1/\alpha \text{ is an odd integer} \\ \hat{\mu} > 0 & \text{otherwise (including } \alpha = 0)\end{cases}$
Log-complement	$\ln(1-\mu)$	$1-\exp(\eta)$	$\hat{\mu} < 1$
Negative log-log	$-\ln(-\ln(\mu))$	$\exp(-\exp(-\eta))$	$\hat{\mu} \in (0, 1)$
Negative binomial[†]	$\ln\left(\dfrac{\mu}{\mu + \dfrac{1}{k}}\right)$	$\dfrac{\exp(\eta)}{k(1-\exp(\eta))}$	$\hat{\mu} > 0$
Odds power(α^*) $\begin{cases}\alpha \neq 0 \\ \alpha = 0\end{cases}$	$\begin{cases}\dfrac{(\mu/(1-\mu))^{\alpha} - 1}{\alpha} \\ \ln\left(\dfrac{\mu}{1-\mu}\right)\end{cases}$	$\begin{cases}\dfrac{(1+\alpha\eta)^{1/\alpha}}{1+(1+\alpha\eta)^{1/\alpha}} \\ \dfrac{\exp(\eta)}{1+\exp(\eta)}\end{cases}$	$\hat{\mu} \in (0, 1)$

[*] α can be a real number. If $|\alpha| < 2.2\text{e-}16$, α is treated as 0.

[†] The negative binomial link function becomes unavailable for negative binomial distribution with $k = 0$.

Table 5: The First and Second Derivatives of Link Function

Link function name	First derivative $g'(\mu) = \dfrac{\partial \eta}{\partial \mu} = \Delta$	Second derivative $g''(\mu) = \dfrac{\partial^2 \eta}{\partial \mu^2}$
Identity	1	0
Log	$\dfrac{1}{\mu}$	$-\Delta^2$
Logit	$\dfrac{1}{\mu(1-\mu)}$	$\Delta^2(2\mu - 1)$
Probit	$\dfrac{1}{\phi(\Phi^{-1}(\mu))}$, where $\phi(z) = \dfrac{1}{\sqrt{2\pi}} e^{-z^2/2}$	$\Delta^2 \Phi^{-1}(\mu)$
Complementary log-log	$\dfrac{1}{(\mu - 1)\ln(1-\mu)}$	$-\Delta^2(1 + \ln(1-\mu))$

Power(α) $\begin{cases}\alpha \neq 0\\ \alpha = 0\end{cases}$	$\begin{cases}\alpha\mu^{\alpha-1}\\ \frac{1}{\mu}\end{cases}$	$\begin{cases}\Delta\frac{\alpha-1}{\mu}\\ -\Delta^2\end{cases}$
Log-complement	$\frac{-1}{1-\mu}$	$-\Delta^2$
Negative log-log	$\frac{-1}{\mu\ln(\mu)}$	$\Delta^2(1+\ln(\mu))$
Negative binomial	$\frac{1}{\mu+k\mu^2}$	$-\Delta^2(1+2k\mu)$
Odds power(α) $\begin{cases}\alpha \neq 0\\ \alpha = 0\end{cases}$	$\begin{cases}\frac{\mu^{\alpha-1}}{(1-\mu)^{\alpha+1}}\\ \frac{1}{\mu(1-\mu)}\end{cases}$	$\begin{cases}\Delta\left(\frac{\alpha-1}{\mu}+\frac{\alpha+1}{1-\mu}\right)\\ \Delta^2(2\mu-1)\end{cases}$

2.2.3. Combination of probability distribution and link function

Choosing different combinations of a proper probability distribution and a link function can result in different models. Table 6 gives a guideline for all distributions except ordinal and nominal multinomial distributions. Cumulative link functions in Table A.1 of Appendix A are only available for ordinal multinomial distribution and generalized logit link function specified in Appendix B is for nominal multinomial distribution. If improper combinations were specified, an error message will be issued.

Note that the available distributions depend on the measurement level of the target and there are 4 different levels in the applications:

a. If a target is continuous, all distributions except nominal and ordinal multinomial would be allowed. Note that binomial is allowed because target could be an “events” variable and user has to also specify “trials” variable). The default is normal distribution.

b. If a target is nominal, then nominal multinomial and binomial distributions are allowed. The default is nominal multinomial.

c. If a target is ordinal, then ordinal, nominal and binomial distributions are allowed. The default is ordinal multinomial.

d. If a target is flag, only binomial distribution is allowed.

Table 6: Proper Combinations of Probability Distribution and Link Function

Distribution Link	Normal	Inverse Gaussian	Gamma	Negative binomial	Poisson	Binomial	Tweedie
Identity	x	x	x	x	x	x	x
Log	x	x	x	x	x	x	x
Logit						x	
Probit						x	
Complementary log-log						x	
Power(α)	x	x	x	x	x	x	x
Log-complement						x	
Negative log-log						x	
Negative binomial				x			
Odds power(α)						x	

2.2.4. Data transformation

To improve numerical stability, the $\boldsymbol{X}$ matrix will be transformed by default (the GLE component has the option to turn it off) according to the following rules:

According to the definition of $\boldsymbol{X}$, the ith row is $\boldsymbol{x}_i = \left(x_{i1}, \cdots, x_{ip}\right)^T, i = 1, \cdots, n$, with $x_{i1} = 1$ if the model has an intercept. Suppose $\boldsymbol{x}_i^*$ is the transformation of $\boldsymbol{x}_i$ then the jth entry of $\boldsymbol{x}_i^*$ is defined as

$$x_{ij}^* = \frac{x_{ij} - c_j}{s_j}$$

where c_j and s_j are centering and scaling values for x_{ij}, respectively, for $j = 1, \cdots, p$ and choices of c_j and s_j, are listed as follows:

- For a non-constant continuous predictor or a derived predictor which includes continuous predictor,
  - if the model has an intercept, $c_1 = 0$ and $c_j = \bar{x}_j, j \neq 1$, where $\bar{x}_j$ is the sample mean of the jth predictor, $\bar{x}_j = \frac{1}{N}\sum_{i=1}^{n} f_i x_{ij}$ and $s_j = 1$ and $s_j = \sqrt{s_{x_j}^2}, j \neq 1$, where $\sqrt{s_{x_j}^2}$ is the

sample standard deviation of the j^{th} predictor and $s_{x_j}^2 = \frac{1}{N-1}\sum_{i=1}^{n} f_i\left(x_{ij} - \bar{x}_j\right)^2$**.** Note that the intercept column is not transformed.

  - o if the model has no intercept, $c_j = 0$ and $s_j = \sqrt{s_{x_j}^2 + \bar{x}_j^2}$**.**

- For a constant predictor, say $x_{ij} = a \neq 0$**,** $c_j = 0$ and $s_j = a$, i.e., only scaled it to be 1 but not centered.
- For a dummy predictor that is derived from a factor or a factor interaction, leave it unchanged, i.e., $c_j = 0$ and $s_j = 1$**.**

In terms of matrix format, if the model (including nominal multinomial distribution) has no intercept,

$$\boldsymbol{X}^* = \boldsymbol{X}\boldsymbol{S}^{-1}, \text{ where } = \mathrm{diag}\left(s_1, \cdots, s_p\right)\ \mathbf{.}$$

If the model (including nominal multinomial distribution) has an intercept,

$$\boldsymbol{X}^* = \boldsymbol{X}\begin{bmatrix} 1 & -\boldsymbol{c}_1^T\boldsymbol{S}_1^{-1} \\ \mathbf{0} & \boldsymbol{S}_1^{-1} \end{bmatrix} = \boldsymbol{X}\boldsymbol{A}, \text{ where } \boldsymbol{c}_1 = \left(c_2, \cdots, c_p\right)^T \text{ and } \boldsymbol{S}_1 = \mathrm{diag}\left(s_2, \cdots, s_p\right)\mathbf{.}$$

Then $\boldsymbol{X}$ is replaced by $\boldsymbol{X}^*$ during estimation.

For ordinal multinomial model, we have

$$\boldsymbol{X}_1^* = [\mathbf{1}_n, \quad -\boldsymbol{X}^*] = [\mathbf{1}_n, \quad -\boldsymbol{X}]\begin{bmatrix} 1 & \boldsymbol{c}_1^T\boldsymbol{S}_1^{-1} \\ \mathbf{0} & \boldsymbol{S}_1^{-1} \end{bmatrix} = [\mathbf{1}_n, \quad -\boldsymbol{X}]\boldsymbol{A} = \boldsymbol{X}_1\boldsymbol{A},$$

where $\mathbf{1}_q$ is a length q vector of 1.

<u>Implementation notes</u>:

- Some predictors may be derived from the original predictors, say interaction term $x_{4i} = x_{2i} \times x_{3i}$**.** For derived predictors (or composite effects), transformation is done only when all original predictors are covariates, i.e., no transformation is needed when there is a factor in derived predictors. And their means and standard deviations are calculated using the derived predictors.
- If the setting of a model includes intercept, normal distribution and identity link function, then the target is centered by its mean (but not scale it due to complication scale may result) for numerical stability, i.e., $y_i^* = y_i - \bar{y}, \forall i$, along with the $\boldsymbol{X}$ transformation. We will use $\boldsymbol{y}^*$ instead of $\boldsymbol{y}$ and treat $-\bar{y}$ as an offset value during estimation. Note this is done ***internally*** without the users knowing.
- The whole transformation process will affect the estimates of $\boldsymbol{\beta}$. After estimation, we need to transform the estimates of $\boldsymbol{\beta}$ and their covariance matrix back from transformed scale to original

scale. And all post-estimation statistics and scoring would also be displayed on original scale, no matter if they are calculated on original or transformed scale. The transform back formulae would be described below and we will simply use $\boldsymbol{X}^* = \boldsymbol{XA}$ and notice that $\boldsymbol{A}$ reduces to $\boldsymbol{S}^{-1}$ if the model has no intercept.

- The log likelihood value, ℓ, is the same on original or transformed scale.
- If the scale parameter, ϕ, for continuous distributions and Tweedie distribution is estimated with regression parameters, then its estimate will be the same based on original or transformed scale.
- If the ancillary parameter, k, in negative binomial distribution is estimated with regression parameters, then its estimate will be the same based on original or transformed scale.
- When iteration history tables are displayed, the parameter estimates in each iteration need to transform back. In addition, it will also display the final gradient vector and Hessian matrix. Suppose the gradient vectors based on original and transformed scale are $\boldsymbol{s}$ and $\boldsymbol{s}^*$, respectively; and the Hessian matrices based on original and transformed scale are $\boldsymbol{H}$ and $\boldsymbol{H}^*$, respectively. Then

$$\boldsymbol{s} = (\boldsymbol{A}^{\mathrm{T}})^{-\mathbf{1}}\boldsymbol{s}^* \text{ and } \boldsymbol{H} = (\boldsymbol{A}^{\mathrm{T}})^{-\mathbf{1}}\boldsymbol{H}^*\boldsymbol{A}^{-1}$$

- In the following sections, we will still use $\boldsymbol{X}$ and $\boldsymbol{y}$ no matter whether they are transformed or not, unless we need to distinguish them.

3. Estimation

Having selected a particular model, it is required to estimate the parameters ($\boldsymbol{\beta}$, ϕ) or ($\boldsymbol{\beta}$, k) and to assess the precision of the estimates. Here we only include parameter estimation first and will add other subsections later.

3.1. Parameter estimation

The parameters ($\boldsymbol{\beta}$, ϕ, k) is estimated by maximizing the log likelihood function ℓ (or the kernel of the log likelihood function ℓ_k) from the observed data. Let $\mathbf{s}$ be the first derivative (gradient) vector of the log likelihood with respect to $\boldsymbol{\beta}$ (and possible ϕ or k, see below), then we wish to solve

$$\mathbf{s} = \left[\frac{\partial \ell}{\partial \boldsymbol{\beta}}\right]_{p\times 1} = \mathbf{0}.$$

In general, there is no closed form solution except a normal distribution with identity link function, so estimates are obtained numerically via an iterative process. A Newton-Raphson and/or Fisher scoring algorithm is used and it is based on a linear Taylor series approximation of the first derivative of the log likelihood, so the first and second derivatives are needed and will be discussed in the first two subsections. Then the iterative process is discussed in the third subsection.

3.1.1. First derivatives

If the scale parameter ϕ for normal, inverse Gaussian, gamma and Tweedie is not estimated by ML method, $\mathbf{s}$ is a $p\times 1$ vector with the form:

$$\mathbf{s}=\sum_{i=1}^{n}\frac{f_i\omega_i\left(y_i-\mu_i\right)}{\phi V(\mu_i)g'(\mu_i)}\cdot\boldsymbol{x}_i=\frac{1}{\phi}\sum_{i=1}^{n}\frac{f_i\omega_i\left(y_i-\mu_i\right)}{V(\mu_i)g'(\mu_i)}\cdot\boldsymbol{x}_i\ ,$$

where μ_i, $V(\mu_i)$ and $g'(\mu_i)$ are defined in Table 4, Table 2 and Table 5, respectively.

Notes:

- The computation of $\mathbf{s}$ can be implemented in map/reduce environment. I.e., assume there are *J* mappers, in the *j*th mapper with n_j records, $\mathbf{s}_j=\frac{1}{\phi}\sum_{i=1}^{n_j}\frac{f_i\omega_i\left(y_i-\mu_i\right)}{V(\mu_i)g'(\mu_i)}\cdot\boldsymbol{x}_i$, then combine the results from all mappers in the reducer, $\mathbf{s}=\sum_{j=1}^{J}\mathbf{s}_j$.
- $\mu_i = g^{-1}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}+o_i)$ is an estimate of the mean of the *i*th observation, obtained from an estimate of the parameter vector $\boldsymbol{\beta}$.
- For binomial distribution (***r***/***m***), ω_i is replaced with ω_i^* .
- If the scale parameter is specified by the deviance or Pearson chi-square, then assume ϕ =1 to estimate $\boldsymbol{\beta}$.

If the scale parameter ϕ for normal, inverse Gaussian gamma and Tweedie is estimated by ML method, it is handled by searching for ln(ϕ) since ϕ is required to be greater than zero. Similarly, if the ancillary parameter *k* for negative binomial is estimated by ML method, it is still handled by searching for ln(*k*) (just replace ϕ with *k*) since *k* is also required to be greater than zero.

Let τ = ln(ϕ) so ϕ = exp(τ) (or τ = ln(*k*) and *k* = exp(τ) for negative binomial) , then $\mathbf{s}$ is a $(p+1)\times 1$ vector with the following form

$$\mathbf{s} = \begin{bmatrix} \dfrac{\partial \ell}{\partial \boldsymbol{\beta}} \\ \dfrac{\partial \ell}{\partial \tau} \end{bmatrix}_{(p+1)\times 1} = \begin{bmatrix} \dfrac{1}{\exp(\tau)} \sum_{i=1}^{n} \dfrac{f_i \omega_i (y_i - \mu_i)}{V(\mu_i) g'(\mu_i)} \cdot \boldsymbol{x}_i \\ \dfrac{\partial \ell}{\partial \tau} \end{bmatrix},$$

where $\partial \ell / \partial \boldsymbol{\beta}$ is the same as the above with ϕ is replaced with exp(τ) (for negative binomial, ϕ is not replaced), $\partial \ell / \partial \tau$ has a different form depending on the distribution as follows:

Table 7: The First Derivative Functions w.r.t. τ for Probability Distributions

Distribution	$\dfrac{\partial \ell}{\partial \tau}$
Normal	$\sum_{i=1}^{n} \dfrac{f_i}{2} \left\{ \dfrac{\omega_i (y_i - \mu_i)^2}{\exp(\tau)} - 1 \right\}$
Inverse Gaussian	$\sum_{i=1}^{n} \dfrac{f_i}{2} \left\{ \dfrac{\omega_i (y_i - \mu_i)^2}{\exp(\tau) y_i \mu_i^2} - 1 \right\}$
Gamma	$\sum_{i=1}^{n} - \dfrac{f_i \omega_i}{\exp(\tau)} \left\{ \ln\left(\dfrac{\omega_i y_i}{\exp(\tau) \mu_i} \right) + \left(1 - \dfrac{y_i}{\mu_i} \right) - \psi\left(\dfrac{\omega_i}{\exp(\tau)} \right) \right\}$
Negative binomial	For all appropriate link functions other than negative binomial link function, $\dfrac{\partial \ell}{\partial \tau} = \sum_{i=1}^{n} \dfrac{f_i \omega_i}{\phi \exp(\tau)} \left\{ \dfrac{\exp(\tau)(y_i - \mu_i)}{(1 + \exp(\tau)\mu_i)} + \ln(1 + \exp(\tau)\mu_i) - \psi\left(y_i + \dfrac{1}{\exp(\tau)} \right) + \psi\left(\dfrac{1}{\exp(\tau)} \right) \right\};$ for the negative binomial link function, $\dfrac{\partial \ell}{\partial \tau} = \sum_{i=1}^{n} \dfrac{f_i \omega_i}{\phi \exp(\tau)} \left\{ \ln(1 + \exp(\tau)\mu_i) - \psi\left(y_i + \dfrac{1}{\exp(\tau)} \right) + \psi\left(\dfrac{1}{\exp(\tau)} \right) \right\}.$
Tweedie	$\sum_{i=1}^{n} f_i \dfrac{\partial \ell_i}{\partial \tau}$, where $\dfrac{\partial \ell_i}{\partial \tau} = \begin{cases} \dfrac{\omega_i \mu_i^{2-q}}{\exp(\tau)(2-q)} & \text{for } y_i = 0 \\ \dfrac{\partial \mathrm{V}_i / \partial \tau}{V_i} - \dfrac{\omega_i y_i \mu_i^{1-q}}{\exp(\tau)(1-q)} + \dfrac{\omega_i \mu_i^{2-q}}{\exp(\tau)(2-q)} & \text{for } y_i > 0 \end{cases}$

Notes:

- $\psi(z)$ is a digamma function, which is the derivative of logarithm of a gamma function, evaluated at z, i.e. $\psi(z)=\dfrac{\partial \ln(\Gamma(z))}{\partial z}=\dfrac{\Gamma'(z)}{\Gamma(z)}$. The method to compute digamma and trigamma functions is described in Appendix D.
- $\dfrac{\partial V_i}{\partial \tau}=(\alpha-1)\sum_{j=1}^{\infty} jV_{ij}$. To avoid the possibility of floating point overflow for $\sum_{j=1}^{\infty} V_{ij}$ and $\sum_{j=1}^{\infty} jV_{ij}$, we will evaluate $\dfrac{\partial V_i / \partial \tau}{V_i}$ directly. See Appendix C for details.

As mentioned above, for normal distribution with identity link function which is a classical linear regression model, there is a closed form solution for both $\boldsymbol{\beta}$ and τ, so no iterative process is needed. The solution for $\boldsymbol{\beta}$, after applying the SWEEP operation, is

$$\hat{\boldsymbol{\beta}}=\left(\sum_{i=1}^{n} f_i\omega_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i\right)^{-}\left(\sum_{i=1}^{n} f_i\omega_i \boldsymbol{x}_i^{\mathrm{T}}\left(y_i-o_i\right)\right)=\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\Psi}\boldsymbol{X}\right)^{-}\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{\Psi}\left(\boldsymbol{y}-\boldsymbol{o}\right)\right),$$

where $\boldsymbol{\Psi}=\operatorname{diag}\left(f_1\omega_1,\dots f_n\omega_n\right)$ and $(\boldsymbol{Z})^{-}$ is the generalized inverse of a matrix $\boldsymbol{Z}$. If the scale parameter ϕ is also estimated by ML method, the estimate of τ (= ln(ϕ)) is

$$\hat{\tau}=\ln\left(\hat{\phi}\right)=\ln\left(\frac{1}{N}\sum_{i=1}^{n} f_i\omega_i\left(y_i-\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}-o_i\right)^2\right).$$

3.1.2. Second derivatives

Let $\mathbf{H}$ be the second derivative (Hessian) matrix. If the scale parameter ϕ for normal, inverse Gaussian, gamma and Tweedie is not estimated by ML method, $\mathbf{H}$ is a $p\times p$ matrix with the form:

$$\mathbf{H}=\left[\frac{\partial^2 \ell}{\partial\boldsymbol{\beta}\,\partial\boldsymbol{\beta}^{\mathrm{T}}}\right]_{p\times p}=-\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\,\boldsymbol{X}$$

where $\boldsymbol{W}$ is an $n\times n$ diagonal matrix. There are two definitions for $\boldsymbol{W}$ depending on which algorithm is used: $\boldsymbol{W_e}$ for Fisher scoring and $\boldsymbol{W_o}$ for Newton Raphson. The ith diagonal element for $\boldsymbol{W_e}$ is

$$w_{e,i} = \frac{f_i \omega_i}{\phi} \cdot \frac{1}{V(\mu_i)\left(g'(\mu_i)\right)^2},$$

and the i^{th} diagonal element for $\boldsymbol{W_o}$ is

$$w_{o,i} = w_{e,i} + \frac{f_i \omega_i}{\phi}\left(y_i - \mu_i\right) \cdot \frac{V(\mu_i)g''(\mu_i) + V'(\mu_i)g'(\mu_i)}{\left(V(\mu_i)\right)^2\left(g'(\mu_i)\right)^3},$$

where $V'(\mu_i)$ and $g''(\mu_i)$ are defined in Table 2 and Table 5, respectively. Then $\boldsymbol{W_e} = \operatorname{diag}\left(w_{e,1}, \ldots w_{e,n}\right)$ and $\boldsymbol{W_o} = \operatorname{diag}\left(w_{o,1}, \ldots, w_{o,n}\right)$. Note the expected value of $\boldsymbol{W_o}$ is $\boldsymbol{W_e}$ and when the canonical link is used for the specified distribution, then $\boldsymbol{W_o}$ = $\boldsymbol{W_e}$. Be aware that for binomial distribution (***r***/***m***), ω_i is replaced with ω_i^*.

<u>Notes</u>:

- The computation of $\mathbf{H}$ can be implemented in map/reduce environment. I.e., assume there are *J* mappers with $\boldsymbol{X}_j$ and $\boldsymbol{W}_j$ as the design matrix and an $n_j \times n_j$ diagonal matrix in the j^{th} mapper, respectively, so $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \vdots \\ \boldsymbol{X}_J \end{bmatrix}$, $\boldsymbol{W} = \operatorname{diag}\left(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_J\right)$ and $\mathbf{H}_j = -\boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{W}_j \boldsymbol{X}_j$ then combine the results from all mappers in the reducer, $\mathbf{H} = \left[\frac{\partial^2 \ell}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}^{\mathrm{T}}}\right]_{p \times p} = \sum_{j=1}^{J} \mathbf{H}_j$.

If the scale parameter ϕ for normal, inverse Gaussian, gamma and Tweedie is estimated by ML method, $\mathbf{H}$ becomes a $(p+1) \times (p+1)$ matrix with the form

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 \ell}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}^{\mathrm{T}}} & \dfrac{\partial^2 \ell}{\partial \boldsymbol{\beta}\, \partial \tau} \\ \dfrac{\partial^2 \ell}{\partial \tau\, \partial \boldsymbol{\beta}^{\mathrm{T}}} & \dfrac{\partial^2 \ell}{\partial \tau^2} \end{bmatrix}_{(p+1) \times (p+1)}$$

where $\partial^2 \ell / \partial \boldsymbol{\beta} \partial \tau$ is a $p \times 1$ vector and $\partial^2 \ell / \partial \tau\, \partial \boldsymbol{\beta}^{\mathrm{T}}$ is a $1 \times p$ vector and the transpose of $\partial^2 \ell / \partial \boldsymbol{\beta} \partial \tau$. The form of $\partial^2 \ell / \partial \boldsymbol{\beta} \partial \tau$ for all three continuous distributions is given below:

$$\frac{\partial^2 \ell}{\partial \boldsymbol{\beta}\, \partial \tau} = \sum_{i=1}^{n} -\frac{f_i \omega_i \left(y_i - \mu_i \right)}{\exp(\tau) V(\mu_i) g'(\mu_i)} \cdot \boldsymbol{x}_i = -\frac{\partial \ell}{\partial \boldsymbol{\beta}}.$$

Note that in theory $\partial \ell \big/ \partial \hat{\boldsymbol{\beta}} = 0$, so $\partial^2 \ell \big/ \partial \hat{\boldsymbol{\beta}} \partial \tau = 0$ when evaluated at the estimates of $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}$. In practice they might not be exact 0, but they should be very close to 0.

The forms of $\partial^2 \ell / \partial \boldsymbol{\beta} \partial \tau$ for negative binomial are as follows depending on the link functions:

For all appropriate link functions other than negative binomial link function,

$$\frac{\partial^2 \ell}{\partial \boldsymbol{\beta} \partial \tau} = \sum_{i=1}^{n} -\frac{f_i \omega_i \exp(\tau) \left(y_i - \mu_i \right)}{\phi \left(1 + \exp(\tau) \mu_i \right)^2 g'(\mu_i)} \cdot \boldsymbol{x}_i;$$

for the negative binomial link function,

$$\frac{\partial^2 \ell}{\partial \boldsymbol{\beta} \partial \tau} = \sum_{i=1}^{n} \frac{f_i \omega_i \mu_i}{\phi} \cdot \boldsymbol{x}_i.$$

The forms of $\partial^2 \ell / \partial \tau^2$ are listed in Table 8.

Table 8: The Second Derivative Functions w.r.t. τ for Probability Distributions

Distribution	$\frac{\partial^2 \ell}{\partial \tau^2}$
Normal	$\sum_{i=1}^{n} -\frac{f_i \omega_i}{2 \exp(\tau)} \left(y_i - \mu_i \right)^2$
Inverse Gaussian	$\sum_{i=1}^{n} -\frac{f_i \omega_i}{2 \exp(\tau) y_i \mu_i^2} \left(y_i - \mu_i \right)^2$
Gamma	$\sum_{i=1}^{n} \frac{f_i \omega_i}{\exp(\tau)} \left\{ \ln\left(\frac{\omega_i y_i}{\exp(\tau) \mu_i} \right) + \left(2 - \frac{y_i}{\mu_i} \right) - \psi\left(\frac{\omega_i}{\exp(\tau)} \right) - \frac{\omega_i}{\exp(\tau)} \psi' \left(\frac{\omega_i}{\exp(\tau)} \right) \right\}$

Negative binomial	For all appropriate link functions other than negative binomial link function, $\frac{\partial^2 \ell}{\partial \tau^2} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \left\{ \begin{array}{l} \frac{-y_i \exp(\tau)\mu_i + \mu_i + 2\exp(\tau)\mu_i^2}{\left(1+\exp(\tau)\mu_i\right)^2} - \frac{1}{\exp(\tau)} \ln\left(1+\exp(\tau)\mu_i\right) + \\ \frac{1}{\exp(\tau)}\left[\psi\left(y_i + \frac{1}{\exp(\tau)}\right) - \psi\left(\frac{1}{\exp(\tau)}\right)\right] + \frac{1}{\exp(2\tau)}\left[\psi'\left(y_i + \frac{1}{\exp(\tau)}\right) - \psi'\left(\frac{1}{\exp(\tau)}\right)\right] \end{array} \right\};$ for the negative binomial link function, $\frac{\partial^2 \ell}{\partial \tau^2} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \left\{ \begin{array}{l} -\frac{1}{\exp(\tau)} \ln\left(1+\exp(\tau)\mu_i\right) + \\ \frac{1}{\exp(\tau)}\left[\psi\left(y_i + \frac{1}{\exp(\tau)}\right) - \psi\left(\frac{1}{\exp(\tau)}\right)\right] + \frac{1}{\exp(2\tau)}\left[\psi'\left(y_i + \frac{1}{\exp(\tau)}\right) - \psi'\left(\frac{1}{\exp(\tau)}\right)\right] \end{array} \right\}.$
Tweedie	$\sum_{i=1}^{n} f_i \frac{\partial^2 \ell_i}{\partial \tau^2}$, where $\frac{\partial^2 \ell_i}{\partial \tau^2} = \begin{cases} -\frac{\omega_i \mu_i^{2-q}}{\exp(\tau)\left(2-q\right)} & \text{for } y_i = 0 \\ \frac{\partial^2 V_i / \partial \tau^2}{V_i} - \left(\frac{\partial V_i / \partial \tau}{V_i}\right)^2 + \frac{\omega_i y_i \mu_i^{1-q}}{\exp(\tau)\left(1-q\right)} - \frac{\omega_i \mu_i^{2-q}}{\exp(\tau)\left(2-q\right)} & \text{for } y_i > 0 \end{cases}$

<u>Notes</u>:

- $\psi'\left(z\right)$ is a trigamma function, which is the derivative of $\psi\left(z\right)$, evaluated at *z*. See Appendix D for details.
- For normal and inverse Gaussian, $\partial^2 \ell / \partial \hat{\tau}^2 = -N/2$ when evaluated at $\hat{\boldsymbol{\beta}}$ and $\hat{\tau}$.
- $\frac{\partial^2 V_i}{\partial \tau^2} = \left(\alpha - 1\right)^2 \sum_{j=1}^{\infty} j^2 V_{ij}$. Again, we will evaluate $\frac{\partial^2 V_i / \partial \tau^2}{V_i}$ directly. See Appendix C for details.
- For normal distribution with identity link function, Hessian matrix is

  $$\mathbf{H} = -\frac{\boldsymbol{X}^T \boldsymbol{\Psi} \boldsymbol{X}}{\hat{\phi}},$$

  and augmented Hessian matrix including the parameter $\tau = \ln \phi$ is

  $$\mathbf{H} = \begin{bmatrix} -(\boldsymbol{X}^T \boldsymbol{\Psi} \boldsymbol{X})/\hat{\phi} & \mathbf{0} \\ \mathbf{0}^T & -\frac{N}{2} \end{bmatrix}.$$

  In addition, the gradient is $\mathbf{0}$.

3.1.3. The iterative process

Note that we will implement the step-halving with Newton Raphson or Fisher scoring method first, but will implement other methods, described in Du and Zheng (2009) and more, in the future.

An iterative process to find the solutions for $\boldsymbol{\beta}$ (which might include ϕ, k for negative binomial or $\boldsymbol{\psi}$ for multinomial) is based on (1) Newton Raphson (for all iterations), (2) Fisher scoring (for all iterations) or (3) a hybrid method. The hybrid method consists of applying Fisher scoring steps for a specified number of iterations before switching to Newton Raphson steps. It is done easily by applying different formula for the Hessian matrix at each iteration. Newton Raphson performs well if the initial values are close to the solution, but the hybrid method can be used to improve the algorithm's robustness to bad initial values. Apart from improved robustness, the Fisher scoring is faster due to the simpler form of the Hessian matrix.

Some definitions are needed for an iterative process:

I	Starting iteration for checking complete separation and quasi-complete separation. It must be 0 or a positive integer. This criterion is not used if the value is 0.
J	The maximum number of steps in step-halving method. It must be a positive integer.
K	The first number of iterations using Fisher scoring, then switching to Newton Raphson. It must be 0 or a positive integer. A value of 0 means using Newton Raphson for all iterations and a value greater or equal to *M* means using Fisher scoring for all iterations.
M	The maximum number of iterations. It must be a non-negative integer. If the value is 0, then initial parameter values become final estimates.
ε_ℓ, ε_p, ε_H	Tolerance levels for three types of convergence criteria (see Section 3.1.3.2 below).
Abs	A 0/1 binary variable; *Abs* = 1 if absolute change is used for convergence criteria and *Abs* = 0 if relative change is used (see Section 3.1.3.2 below).

And the iterative process is outlined as follows:

(1) Input values for *I*, *J*, *K*, *M*, ε_ℓ, ε_p, ε_H and *Abs* for each type of three convergence criteria.

(2) Input initial values $\boldsymbol{\beta}^{(0)}$ or if no initial values are given, compute initial values $\boldsymbol{\beta}^{(0)}$ (see Section 3.1.3.1 below), then calculate log likelihood $\ell^{(0)}$, gradient vector $\mathbf{s}^{(0)}$ and Hessian matrix $\mathbf{H}^{(0)}$ based on $\boldsymbol{\beta}^{(0)}$.

(3) Let $\xi = 1$.

(4) Compute estimates of i^{th} iteration:

$$\boldsymbol{\beta}^{(i)} = \boldsymbol{\beta}^{(i-1)} - \xi\left(\mathbf{H}^{(i-1)}\right)^{-}\mathbf{s}^{(i-1)},$$

where $(\mathbf{H})^{-}$ is a generalized inverse of $\mathbf{H}$. Then compute log likelihood $\ell^{(i)}$ based on $\boldsymbol{\beta}^{(i)}$.

(5) Use step-halving method if $\ell^{(i)} < \ell^{(i-1)}$: reduce ξ by half and repeat step (4). I.e., the set of values of ξ is $\{(1/2)^{j}$: j = 0, …, J – 1}. If J is reached but the log likelihood is not improved, issue a warning message, then stop.

(6) Compute gradient vector $\mathbf{s}^{(i)}$ and Hessian matrix $\mathbf{H}^{(i)}$ based on $\boldsymbol{\beta}^{(i)}$. Note that $\boldsymbol{W_e}$ is used to calculate $\mathbf{H}^{(i)}$ if $i \le K$; $\boldsymbol{W_o}$ is used to calculate $\mathbf{H}^{(i)}$ if $i > K$.

(7) Check if complete or quasi-complete separation of the data is established (see the note below on how to check them) if distribution is binomial or multinomial and the current iteration $i \ge I$. If either complete or quasi-complete separation is detected, issue a warning message, then stop.

(8) Check if all three convergence criteria (see Section 3.1.3.2 below) are met. If they are not but M is reached, issue a warning message, then stop.

(9) If all three convergence criteria are met, check if complete or quasi-complete separation of the data is established if distribution is binomial or multinomial and $i < I$ (because checking for complete or quasi-complete separation has not started yet). If complete or quasi-complete separation is detected, issue a warning message, then stop, otherwise, stop (the process converges for binomial or multinomial successfully). If all three convergence criteria are met for the distributions other than binomial and multinomial, stop (the process converges for other distributions successfully). The final vector of estimates is denoted by $\hat{\boldsymbol{\beta}}$ (and $\hat{\tau}$ and $\hat{\boldsymbol{\psi}}$ for multinomial). Otherwise, go back to step (3). See Figure 1: The Flowchart of the Iterative Process of Parameter Estimation below.

Notes:

- How the scale parameter ϕ is handled in the above iterative process:

1. If ϕ (τ), for normal, inverse Gaussian, gamma and Tweedie distributions, is estimated by the ML method, then ϕ will be estimated jointly with regression parameters $\boldsymbol{\beta}$. I.e., the last element of the gradient vector **s** is with respect to τ
2. If ϕ is set to be a fixed positive value, then ϕ will be held fixed at that value for in each iteration of the above process.
3. If ϕ is specified for all distributions by the deviance or Pearson chi-square divided by degrees of freedom (see Section 4.3.3), then ϕ will be fixed at 1 to obtain the estimates of $\boldsymbol{\beta}$ (and $\boldsymbol{\psi}$ for multinomial) in the whole iterative process. Based on $\hat{\boldsymbol{\beta}}$ (and $\hat{\boldsymbol{\psi}}$ for multinomial), calculate the deviance and Pearson chi-square values and obtain $\hat{\phi}$, then revise some statistics, such as the gradient vector, the Hessian matrix, the covariance matrix, etc. see Section 4.1 for details.

- Complete separation or quasi-complete separation of the data is checked for binomial, nominal multinomial and ordinal multinomial distributions here just like what we did in CSLOGISTIC and CSORDINAL procedures. The method is briefly described as follows, see Fang (2004) case-wise data for details):

  For each iteration after a user-specified number of iterations, i.e., if $i > I$, and for binomial models, calculate (note here v refers to records in the dataset)

$$p_{\min} = \min_v p_v$$

$$p_{\max} = \max_v p_v,$$

$$p_{\min}^* = \min_v \left(\min\left(\mu_v, 1-\mu_v \right) \right),$$

where $p_v = \begin{cases} \mu_v & \text{if } y_v = \text{success } (=1) \\ 1-\mu_v & \text{if } y_v = \text{failfure } (=0) \end{cases}$ (p_v is the probability of the observed target for record v) and $\mu_v = g^{-1}(\boldsymbol{x}_v^{\mathrm{T}}\boldsymbol{\beta} + o_v)$; for multinomial model, the definitions of $p_{\min}$, $p_{\max}$ and $p_{\min}^*$ are modified as follows:

$$p_{\min} = \min_v \pi_{v,y_v}$$

$$p_{\max} = \max_v \pi_{v,y_v},$$

$$p_{\min}^* = \min_v \left(\min_j \pi_{v,j} \right).$$

Note that π_{v,y_v} has been defined before for multinomial models. Then the rules of checking complete separation or quasi-complete separation for binomial or multinomial models would be the same. If $p_{\min} = p_{\max} = 1$ (actually $\min\left(p_{\min}, p_{\max}\right) = p_{\min} > 0.99$ is checked) there is a complete separation. Else if (1) $p_{\max} > 0.99$ or $p^*_{\min} < 0.001$ and if (2) there are very small diagonal elements (absolute value $< \sqrt{10^{-7}} \approx 3.16\times10^{-4}$) in the non-redundant parameter locations in matrix **A**, where **A** is the lower triangular matrix in Cholesky decomposition of –**H**, where **H** is the Hessian matrix, such that $-\mathbf{H} = \mathbf{A}\mathbf{A}^{\mathrm{T}}$, then there is a quasi-complete separation.

The developers will evaluate whether the implementation of complete separation or quasi-complete separation checking makes sense in map/reduce environment.

- Whenever a warning message is issued, the procedure continues and results based on the last iteration are given, though the validity of the model fit is questionable.
- If the hybrid method converges with Fisher scoring step, the process will continue with Newton Raphson steps till it converges again.

3.1.3.1. Initial values

The users can specify their own initial values. The order is the intercept (if there is one), regression parameters (and the scale parameter ϕ if it will be estimated by the ML method for normal, inverse Gaussian and gamma and the ancillary parameter *k* if it is estimated by the ML method for negative binomial) for all distributions except multinomial. For ordinal multinomial, the order is threshold parameters and regression parameters. For nominal multinomial, the order is regression parameters for each category (except the reference category). See Appendices A and B for details. If the users didn’t specify them, we have to compute initial values internally. For all distributions except multinomial, the initial values $\boldsymbol{\beta}^{(0)}$ and/or the scale parameter $\phi^{(0)}$ (if it is estimated by ML method) are calculated as follows:

(1) Set the initial fitted values $\tilde{\mu}_{\boldsymbol{i}} = (y_i m_i + 0.5)/(m_i + 1)$ for a binomial distribution (y_i can be a proportion or 0/1 value) and $\tilde{\mu}_{\boldsymbol{i}} = y_i$ for a non-binomial distribution. From them deriving $\tilde{\eta}_i = g(\tilde{\mu}_i)$, $g'(\tilde{\mu}_i)$ and $V(\tilde{\mu}_i)$. If $\tilde{\eta}_i$ becomes undefined, $\tilde{\eta}_i = 1$.

(2) Calculate the weight matrix $\tilde{\boldsymbol{W}}_e$ with the diagonal element $\tilde{w}_{ei} = \dfrac{f_i\omega_i}{\phi}\cdot\dfrac{1}{V(\tilde{\mu}_i)\left(g'(\tilde{\mu}_i)\right)^2}$, where ϕ is set to 1 or a fixed positive value. If the denominator of $\tilde{w}_{ei}$ becomes 0, $\tilde{w}_{ei}$ = 0.

(3) Assign the adjusted target variable $\boldsymbol{z}$ with the i^{th} observation $z_i = (\tilde{\eta}_i - o_i) + (y_i - \tilde{\mu}_i)g'(\tilde{\mu}_i)$ for a binomial distribution and $z_i = (\tilde{\eta}_i - o_i)$ for a non-binomial distribution.

(4) Calculate the initial parameter values

$$\boldsymbol{\beta}^{(0)} = (\boldsymbol{X}^{\mathrm{T}}\tilde{\boldsymbol{W}}_e\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\tilde{\boldsymbol{W}}_e\boldsymbol{z}, \text{ and/or}$$

$$\begin{aligned} \phi^{(0)} &= \frac{1}{N}\left(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta}^{(0)}\right)^T\widetilde{\boldsymbol{W}}_e\left(\boldsymbol{z}-\boldsymbol{X}\boldsymbol{\beta}^{(0)}\right) \\ &= \frac{1}{N}\left(\boldsymbol{z}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z} - 2\left(\boldsymbol{\beta}^{(0)}\right)^T\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z} + \left(\boldsymbol{\beta}^{(0)}\right)^T\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{X}\boldsymbol{\beta}^{(0)}\right) \end{aligned}$$

For the ancillary parameter k of negative binomial, initial $k = 1$, so $\tau = 0$ for now.

Notes:

- The computation of the initial values can be implemented in map/reduce environment. I.e., assume there are J mappers, the first 3 steps would result $\boldsymbol{z}_j$ and $\widetilde{\boldsymbol{W}}_{e,j}$ as an $n_j \times 1$ adjusted target vector and an $n_j \times n_j$ diagonal matrix in the jth mapper, respectively, along with $\boldsymbol{X}_j$. And $\boldsymbol{X}_j^T\widetilde{\boldsymbol{W}}_{e,j}\boldsymbol{X}_j$, $\boldsymbol{z}_j^T\widetilde{\boldsymbol{W}}_{e,j}\boldsymbol{X}_j$, $\boldsymbol{z}_j^T\widetilde{\boldsymbol{W}}_{e,j}\boldsymbol{z}_j$ can be computed in the jth mapper. Then combine the results from all mappers in the reducer as $\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{X} = \sum_{j=1}^{J}\boldsymbol{X}_j^T\widetilde{\boldsymbol{W}}_{e,j}\boldsymbol{X}_j$, $\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z} = \sum_{j=1}^{J}\boldsymbol{X}_j^T\widetilde{\boldsymbol{W}}_{e,j}\boldsymbol{z}_j$ and $\boldsymbol{z}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z} = \sum_{j=1}^{J}\boldsymbol{z}_j^T\widetilde{\boldsymbol{W}}_{e,j}\boldsymbol{z}_j$. Finally, compute $\boldsymbol{\beta}^{(0)}$ based on $\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{X}$ and $\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z}$, and then $\phi^{(0)}$ based on $\boldsymbol{z}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z}$, $\left(\boldsymbol{\beta}^{(0)}\right)^T\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{z}$, and $\left(\boldsymbol{\beta}^{(0)}\right)^T\boldsymbol{X}^T\widetilde{\boldsymbol{W}}_e\boldsymbol{X}\boldsymbol{\beta}^{(0)}$.

3.1.3.2. Convergence criteria

We consider 3 types of convergence criteria here: log-likelihood convergence, parameter convergence, and Hessian convergence. For each type, we consider both absolute and relative change. Let ε_ℓ, ε_{p} and ε_{H} be given tolerance levels for each type, then the criteria can be written as follows:

(1) Log-likelihood convergence: $\begin{cases} \dfrac{\left|\ell^{(i)}-\ell^{(i-1)}\right|}{\left|\ell^{(i-1)}\right|+10^{-6}} < \varepsilon_\ell & \text{if relative change} \\ \left|\ell^{(i)}-\ell^{(i-1)}\right| < \varepsilon_\ell & \text{if absolute change} \end{cases}$.

(2) Parameter convergence: $\begin{cases} \max\limits_j\left(\dfrac{\left|\beta_j^{(i)}-\beta_j^{(i-1)}\right|}{\left|\beta_j^{(i-1)}\right|+10^{-6}}\right) < \varepsilon_{\mathrm{p}} & \text{if relative change} \\ \max\limits_j\left(\left|\beta_j^{(i)}-\beta_j^{(i-1)}\right|\right) < \varepsilon_{\mathrm{p}} & \text{if absolute change} \end{cases}$.

(3) Hessian convergence:
$$\begin{cases} \dfrac{\left(\mathbf{s}^{(i)}\right)^{\mathrm{T}}\left(\mathbf{H}^{(i)}\right)^{-}\left(\mathbf{s}^{(i)}\right)}{\left|\ell^{(i)}\right|+10^{-6}} < \varepsilon_{\mathrm{H}} & \text{if relative change} \\ \left(\mathbf{s}^{(i)}\right)^{\mathrm{T}}\left(\mathbf{H}^{(i)}\right)^{-}\left(\mathbf{s}^{(i)}\right) < \varepsilon_{\mathrm{H}} & \text{if absolute change} \end{cases}.$$

Notes:

- Depending on a user's choice, either relative or absolute change is considered.
- If the user doesn't specify Hessian convergence criterion, we would check if it is met based on absolute change with ε_{H} = 1.0*e*-4 after specified log-likelihood convergence criterion and/or parameter convergence criterion has been satisfied. If Hessian convergence criterion was not met, a warning message, such as "All default or specified convergence criteria are satisfied, but Hessian convergence criterion is not. The convergence is uncertain." would be displayed.

3.1.3.3. Null model and intercept-only model

For the null model and intercept-only model, we provide an approximation method by considering the tradeoff between the performance and the computational cost.

(a) Null models

If the scale parameter ϕ or the ancillary parameter k is estimated by ML method,

Let $\hat{\mu}_0 = g^{-1}(o_i)$

- For normal distribution,

$$\hat{\phi} = \frac{1}{N}\sum_{i=1}^{n} f_i\omega_i(y_i - \hat{\mu}_0)^2$$

- For inverse Gaussian distribution,

$$\hat{\phi} = \frac{1}{N}\sum_{i=1}^{n} \frac{f_i\omega_i(y_i - \hat{\mu}_0)^2}{y_i\hat{\mu}_0^2}$$

- For gamma and tweedie, there is no closed form solution for $\hat{\phi}$, and it needs a iterative process. Herein, it is approximated by its initial value calculated in Section 3.1.3.1.
- For the ancillary parameter k, it is set to 1.0.

(b) Intercept-only models

- For all distributions except multinomial

Let β_0 be parameters of the intercept-only model (excluding ϕ and k).

There is no closed form solution for $\hat{\beta}_0$ in addition to $\hat{\phi}$ or k . $\hat{\beta}_0$ and $\hat{\phi}$ (or k) are approximated by their initial values calculated in Section 3.1.3.1.

- For ordinal multinomial

  Let $\boldsymbol{B}_0 = \left(\boldsymbol{\psi}^{(0)T}, \mathbf{0}^{\mathrm{T}}\right)^{\mathrm{T}}$ be parameters of the threshold-only model.
  If there is no offset variable,

$$\psi_j^{(0)} = g\left(\frac{\sum_{l=1}^{j} N_l}{N}\right), j = 1, \cdots, J-1,$$

  and if there is an offset variable, there is no close form solution, and it needs a iterative process. Herein, they are approximated by their initial values given in Appendix A.

- For nominal multinomial

  Let $\boldsymbol{\beta}_0 = \left(\boldsymbol{\beta}_1^{(0)\mathrm{T}}, \cdots, \boldsymbol{\beta}_{J-1}^{(0)\mathrm{T}}\right)^{\mathrm{T}}$ be parameters for the intercept-only model.

  If there is no offset variable,

$$\beta_{j1}^{(0)} = \ln\left(\frac{N_j}{N_J}\right), \beta_{jk}^{(0)} = 0, j = 1, \cdots, J-1,$$

  and if there is an offset variable, there is no close form solution, and it needs a iterative process. Herein, they are approximated by their initial values given in Appendix B.

Note that when there is no closed form solution for the model under consideration and the approximate model is used, then a warning message, such as “The parameter estimates may not be accurate for the approximate model being used”, would be displayed.

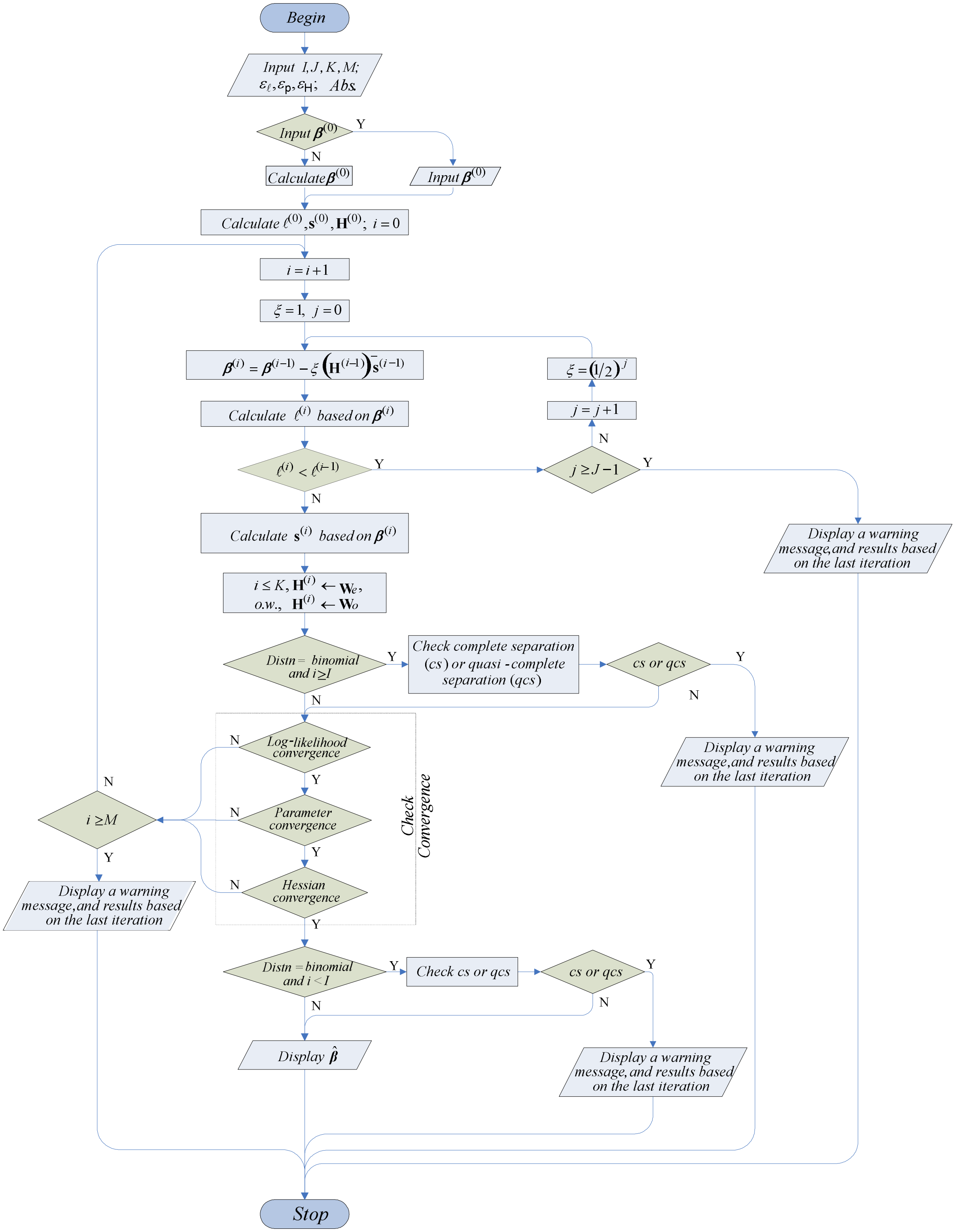

Begin
Input $I, J, K, M;$
$\varepsilon_\ell, \varepsilon_{\mathrm{p}}, \varepsilon_{\mathrm{H}};$ $Abs.$
Input $\boldsymbol{\beta}^{(0)}$
Y
N
Calculate $\boldsymbol{\beta}^{(0)}$
Input $\boldsymbol{\beta}^{(0)}$
Calculate $\ell^{(0)}, \mathbf{s}^{(0)}, \mathbf{H}^{(0)};$ $i = 0$
$i = i + 1$
$\xi = 1,$ $j = 0$
$\boldsymbol{\beta}^{(i)} = \boldsymbol{\beta}^{(i-1)} - \xi \left(\mathbf{H}^{(i-1)}\right)^{-}\mathbf{s}^{(i-1)}$
$\xi = (1/2)^{j}$
$j = j + 1$
Calculate $\ell^{(i)}$ based on $\boldsymbol{\beta}^{(i)}$
$\ell^{(i)} < \ell^{(i-1)}$
Y
N
$j \geq J - 1$
Y
N
Calculate $\mathbf{s}^{(i)}$ based on $\boldsymbol{\beta}^{(i)}$
Display a warning message,and results based on the last iteration
$i \leq K, \mathbf{H}^{(i)} \leftarrow \mathbf{W}_e,$
$o.w.,$ $\mathbf{H}^{(i)} \leftarrow \mathbf{W}_o$
Distn = binomial and $i \geq I$
Y
Check complete separation (cs) or quasi-complete separation (qcs)
cs or qcs
Y
N
N
Log-likelihood convergence
N
Y
Display a warning message,and results based on the last iteration
N
$i \geq M$
N
Parameter convergence
Y
Check Convergence
Y
N
Hessian convergence
Display a warning message,and results based on the last iteration
Y
Distn = binomial and $i < I$
Y
Check cs or qcs
cs or qcs
Y
N
N
Display $\hat{\boldsymbol{\beta}}$
Display a warning message, and results based on the last iteration
Stop

Figure 1: The Flowchart of the Iterative Process of Parameter Estimation in GLE

3.1.4. Parameter estimation on original scale

If the $\boldsymbol{X}$ matrix is transformed, then the final estimates of $\boldsymbol{\beta}$ above are based on transformed scale, denoted it as $\widehat{\boldsymbol{\beta}}^*$. They would be transformed back on original scale, denoted it as $\widehat{\boldsymbol{\beta}}$, as follows:

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{A}\widehat{\boldsymbol{\beta}}^*$$

Note that $\boldsymbol{A}$ could reduce to $\boldsymbol{S}^{-1}$ and hereafter in the document, superscript $*$ is added to a quantity to denote the quantity on transformed scale.

For ordinal multinomial model, we have

$$\widehat{\boldsymbol{B}} = \begin{bmatrix} \widehat{\boldsymbol{\psi}} \\ \widehat{\boldsymbol{\beta}} \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} \widehat{\boldsymbol{\psi}}^* \\ \widehat{\boldsymbol{\beta}}^* \end{bmatrix} = \boldsymbol{T}\widehat{\boldsymbol{B}}^*$$

where $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{I}_{J-1} & \boldsymbol{I}_{J-1} \otimes (\boldsymbol{c}_1^T \boldsymbol{S}_1^{-1}) \\ \boldsymbol{0} & \boldsymbol{S}_1^{-1} \end{bmatrix}$.

For nominal multinomial model, we have

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{T}\widehat{\boldsymbol{\beta}}^*$$

where $\boldsymbol{T} = \oplus_{j=1}^{J-1} \boldsymbol{A}_j$, and $\boldsymbol{A}_j = \boldsymbol{A}$ if the model has an intercept and $\boldsymbol{A}_j = \boldsymbol{S}^{-1}$ if the model has no intercept.

Notes:

- If $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{B}$ is a $p \times q$ matrix, then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is the $mp \times nq$ block matrix,

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix}.$$

- If $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{B}$ is a $p \times q$ matrix, then the direct sum $\boldsymbol{A} \oplus \boldsymbol{B}$ is defined as

$$\boldsymbol{A} \oplus \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & b_{p1} & \cdots & b_{pq} \end{bmatrix}$$

  In general, the direct sum of n matrices is

$$\oplus_{i=1}^{n} \boldsymbol{A}_i = \mathrm{diag}(\boldsymbol{A}_1, \cdots, \boldsymbol{A}_n) = \begin{bmatrix} \boldsymbol{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{A}_n \end{bmatrix}.$$

4. Inference and Model Summary

4.1 Parameter inference

4.1.1 Parameter estimate covariance matrix, correlation matrix and standard error

The parameter estimate covariance matrix, correlation matrix and standard errors can be obtained easily with parameter estimates. Whether or not the scale parameter $\phi(\tau)$ is estimated by ML method, parameter estimate covariance and correlation matrices are listed for $\widehat{\boldsymbol{\beta}}$ only because the covariance between $\widehat{\boldsymbol{\beta}}$ and $\hat{\tau}$ should be zeros. For the ancillary parameter k (τ) of negative binomial is estimated by ML method, parameter estimate covariance and correlation matrices are still listed for $\widehat{\boldsymbol{\beta}}$ only for simplicity purpose even though the covariance between $\widehat{\boldsymbol{\beta}}$ and $\hat{\tau}$ is generally not zero. For ordinal multinomial model, parameter estimate covariance and correlation matrices are listed for $\widehat{\boldsymbol{B}} = (\widehat{\boldsymbol{\psi}}^{\mathrm{T}}, \widehat{\boldsymbol{\beta}}^{\mathrm{T}})^{\mathrm{T}}$.

4.1.1.1 Parameter estimate covariance

Two parameter estimate covariance matrices can be calculated: model-based and robust.

(a) Model-based parameter estimate covariance

The parameter estimate covariance matrix is given by

$$\boldsymbol{\Sigma}_m = -\mathbf{H}^{-}$$

where $\mathbf{H}^{-}$ is the generalized inverse of Hessian matrix $\mathbf{H}$ evaluated at $\widehat{\boldsymbol{\beta}}$ (and $\widehat{\boldsymbol{B}}$ for ordinal multinomial) (and $\widehat{\phi}$ if the scale parameter is estimated for normal, inverse Guassian, gamma and Tweedie distributions by ML method or specified for all distributions by the deviance or Pearson chi-square divided by degrees of freedom).

Notes:

- For normal distribution with identity link function (linear regression model), $\boldsymbol{\Sigma}_m = (\boldsymbol{X}^T\boldsymbol{\Psi}\boldsymbol{X})^-$ where $\boldsymbol{\Psi} = \mathrm{diag}\left(f_1\omega_1,\dots f_n\omega_n\right)$.
- For hybrid method, $\boldsymbol{W}_o$ is used to calculate $\boldsymbol{\Sigma}_m$ even $\widehat{\boldsymbol{\beta}}$ converges within iterations of Fisher scoring steps. Naturally, $\boldsymbol{W}_o$ and $\boldsymbol{W}_e$ are used for Newton Raphson and Fisher scoring method, respectively.
- The corresponding rows and columns for redundant parameter estimates should be set to zero.

(b) Robust parameter estimate covariance

The validity of the parameter estimate covariance matrix based on the Hessian depends on the correct specification of the variance function of the response in addition to the correct specification of the mean regression function of the response. The robust parameter estimate covariance provides a consistent estimate even when the specification of the variance function of the response is incorrect. The robust estimator is also called Huber's estimator because Huber (1967) was the first one described this variance estimate; White's estimator or HCCM (heteroskedasticity consistent covariance matrix) estimator because White (1980) independently showed that this variance estimate is consistent under a linear regression model including heteroskedasticity; or sandwich estimator because the formula has a gradient factor "sandwiched" between two Hessian matrices. The robust (or Huber/White/sandwich) estimator is defined as follows

$$\boldsymbol{\Sigma}_r = \boldsymbol{\Sigma}_m\left(\sum_{i=1}^{n}\left[\frac{\partial \ell_i}{\partial \boldsymbol{\beta}}\right]\left[\frac{\partial \ell_i}{\partial \boldsymbol{\beta}}\right]^{\mathrm{T}}\right)\boldsymbol{\Sigma}_m = \boldsymbol{\Sigma}_m\left(\sum_{i=1}^{n} f_i\cdot\left(\frac{\omega_i(y_i-\mu_i)}{\phi V(\mu_i)g'(\mu_i)}\right)^2\cdot\boldsymbol{x}_i\cdot\boldsymbol{x}_i^{\mathrm{T}}\right)\boldsymbol{\Sigma}_m$$

Notes:

- The robust parameter estimate covariance matrix is justified by asymptotic arguments, but the small sample performance might not be good. For linear regression model, some modifications can be installed to improve small sample performance, but it is not clear if these modifications are applicable to other generalized linear models as well.

For ordinal multinomial model,

$$\boldsymbol{\Sigma}_r = \boldsymbol{\Sigma}_m\left(\sum_{i=1}^{n}\left[\frac{\partial \ell_i}{\partial \boldsymbol{B}}\right]\left[\frac{\partial \ell_i}{\partial \boldsymbol{B}}\right]^{\mathrm{T}}\right)\boldsymbol{\Sigma}_m$$

where $\frac{\partial \ell_i}{\partial \boldsymbol{B}}$ is the first derivative for the i^{th} record and can be found in Appendix A.

For nominal multinomial model,

$$\mathbf{\Sigma}_r = \mathbf{\Sigma}_m \left(\sum_{i=1}^{n} \left[\frac{\partial \ell_i}{\partial \boldsymbol{\beta}} \right] \left[\frac{\partial \ell_i}{\partial \boldsymbol{\beta}} \right]^{\mathrm{T}} \right) \mathbf{\Sigma}_m$$

Where $\frac{\partial \ell_i}{\partial \boldsymbol{\beta}}$ is the first derivative for the ith record and can be found in Appendix B.

4.1.1.2 Parameter estimate correlation

The correlation matrix is calculated from the covariance matrix as usual. Let σ_{ij} be an element of $\mathbf{\Sigma}_m$ or $\mathbf{\Sigma}_r$, then the corresponding element of the correlation matrix is $\frac{\sigma_{ij}}{\sqrt{\sigma_{ii}}\sqrt{\sigma_{jj}}}$. The corresponding rows and columns for redundant parameter estimates should be set to system missing values.

4.1.1.3 Parameter estimate standard error

Let $\hat{\beta}_i$ denote a non-redundant parameter estimate for all distributions except multinomial. Its standard error is the square root of the *i*-th diagonal element of $\mathbf{\Sigma}_m$ or $\mathbf{\Sigma}_r$:

$$\hat{\sigma}_{\beta_i} = \sqrt{\sigma_{ii}}$$

The standard error for redundant parameter estimates is set to a system missing value.

If the scale parameter is estimated by ML method, the standard estimate of $\hat{\tau}$ is

$$\hat{\sigma}_\tau = \sqrt{-\frac{1}{\frac{\partial^2 \ell}{\partial \tau^2}}}$$

where $\frac{\partial^2 \ell}{\partial \tau^2}$ can be found on Table 8. However, people are usually more interested in the original than the transformed τ, so we will only list the estimation result for ϕ. The estimate of ϕ is $\exp(\hat{\tau})$, the standard error estimated of $\hat{\phi}$ is $(\exp(\hat{\tau}) \cdot \hat{\sigma}_\tau)$.

For ordinal multinomial model:

Let $\hat{\psi}_j, j = 1, \cdots, J-1$, be threshold parameter estimates and $\hat{\beta}_i, i = 1, \cdots, p$, denote the non-redundant regression parameter estimates. Their standard errors are the square root of the *i*-th diagonal element of $\mathbf{\Sigma}_m$ or $\mathbf{\Sigma}_r$:

$$\hat{\sigma}_{\psi_j} = \sqrt{\sigma_{jj}} \text{ and } \hat{\sigma}_{\beta_j} = \sqrt{\sigma_{(J-1+i),(J-1+i)}}, \text{ respectively.}$$

For nominal multinomial model,

Let $\hat{\beta}_{jk}$ denote a non-redundant parameter estimate. It standard error is the square root of the $\left((j-1)p+k\right)^{\text{th}}$ diagonal element of $\mathbf{\Sigma}_m$ or $\mathbf{\Sigma}_r$**,**

$$\hat{\sigma}_{\beta_{jk}} = \sqrt{\sigma_{((j-1)p+k,(j-1)p+k)}}$$

Notes

- For normal distribution with identity link function (linear regression model), the standard error of $\hat{\phi}$ is

$$\hat{\sigma}_{\hat{\phi}} = \hat{\phi}\sqrt{\frac{2}{N}}.$$

4.1.1.4 Parameter estimate covariance matrix, correlation matrix and standard error on original scale

If the $\boldsymbol{X}$ matrix is transformed, then the model-based parameter estimate covariance matrices above are also based on transformed scale. They should be transformed back to original scale.

(a) Model-based parameter estimate covariance

Denote the model-based parameter estimate covariance matrices based on original and transformed scale are $\boldsymbol{\Sigma}_m$ and $\boldsymbol{\Sigma}_m^*$, respectively.

$$\boldsymbol{\Sigma}_m = \boldsymbol{A}\boldsymbol{\Sigma}_m^*\boldsymbol{A}^{\mathrm{T}}$$

For ordinal and multinomial models,

$$\boldsymbol{\Sigma}_m = \boldsymbol{T}\boldsymbol{\Sigma}_m^*\boldsymbol{T}^{\mathrm{T}}$$

(b) Robust parameter estimate covariance

Denote the robust parameter estimate covariance matrices based on original and transformed scale are $\boldsymbol{\Sigma}_r$ and $\boldsymbol{\Sigma}_r^*$, respectively.

$$\boldsymbol{\Sigma}_r = \boldsymbol{A}\boldsymbol{\Sigma}_r^*\boldsymbol{A}^{\mathrm{T}}.$$

For ordinal and multinomial models,

$$\boldsymbol{\Sigma}_r = \boldsymbol{T}\boldsymbol{\Sigma}_r^*\boldsymbol{T}^{\mathrm{T}}.$$

(c) Parameter estimate correlation

They are calculated based on $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$ rather than $\boldsymbol{\Sigma}_m^*$ or $\boldsymbol{\Sigma}_r^*$**.**

(d) Parameter estimate standard error

For regression parameters, they are calculated based on $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$. For the scale parameter and ancillary parameter, parameter estimate standard errors are the same no matter which scale is used.

4.1.2 Wald confidence intervals

Wald confidence interval is provided for each non-redundant parameter. Wald confidence intervals are based on the asymptotically normal distribution of the parameter estimators. The parameter estimators includes $\widehat{\boldsymbol{\beta}}$ (and $\widehat{\boldsymbol{\psi}}$ for multinomial), $\hat{\phi}$ $(\hat{\tau})$ if ϕ is estimated by ML.

The $100(1-\alpha)\%$ Wald confidence interval for β_j is given by

$$\left(\hat{\beta}_j - z_{1-\alpha/2}\hat{\sigma}_{\beta_j}, \hat{\beta}_j + z_{1-\alpha/2}\hat{\sigma}_{\beta_j}\right)$$

where z_p is the $(100p)^{\mathrm{th}}$ percentile of the standard normal distribution.

If exponentiated parameter estimates of $\boldsymbol{\beta}$ are required, then the estimate of $\exp(\beta_j)$ is $\exp(\hat{\beta}_j)$, the standard error estimate of $\exp(\hat{\beta}_j)$ is $\left(\exp(\hat{\beta}_j)\cdot\hat{\sigma}_{\beta_j}\right)$ and the corresponding $100(1-\alpha)\%$ Wald confidence interval for $\exp(\beta_j)$ is

$$\left(\exp\left(\hat{\beta}_j - z_{1-\alpha/2}\hat{\sigma}_{\beta_j}\right), \exp\left(\hat{\beta}_j + z_{1-\alpha/2}\hat{\sigma}_{\beta_j}\right)\right)$$

Wald confidence intervals for redundant parameter estimates are set to system missing values.

Similarly, the $100(1-\alpha)\%$ Wald confidence interval for τ is defined as

$$\left(\hat{\tau} - z_{1-\alpha/2}\hat{\sigma}_{\tau}, \hat{\tau} + z_{1-\alpha/2}\hat{\sigma}_{\tau}\right)$$

where $\hat{\tau}$ is the maximum likelihood estimate of τ, $\hat{\sigma}_{\tau}$ is the standard error estimate of $\hat{\tau}$ and the corresponding $100(1-\alpha)\%$ Wald confidence interval for ϕ (or k) is defined as

$$\left(\exp\left(\hat{\tau} - z_{1-\alpha/2}\hat{\sigma}_{\tau}\right), \exp\left(\hat{\tau} + z_{1-\alpha/2}\hat{\sigma}_{\tau}\right)\right).$$

For ordinal multinomial distribution, in addition to $\boldsymbol{\beta}$, the $100(1-\alpha)\%$ Wald confidence interval for ψ_j is given by

$$\left(\hat{\psi}_j - z_{1-\alpha/2}\hat{\sigma}_{\psi_j}, \hat{\psi}_j + z_{1-\alpha/2}\hat{\sigma}_{\psi_j}\right).$$

The estimate of $\exp(\psi_j)$ is $\exp(\hat{\psi}_j)$, the standard error estimate of $\exp(\hat{\psi}_j)$ is $\left(\exp(\hat{\psi}_j)\cdot\hat{\sigma}_{\psi_j}\right)$ and the corresponding $100(1-\alpha)\%$ Wald confidence interval for $\exp(\psi_j)$

$$\left(\exp\left(\hat{\psi}_j - z_{1-\alpha/2}\hat{\sigma}_{\psi_j}\right), \exp\left(\hat{\psi}_j + z_{1-\alpha/2}\hat{\sigma}_{\psi_j}\right)\right).$$

For nominal multinomial distribution, the $100(1-\alpha)\%$ Wald confidence interval for β_{jk} is given by

$$\left(\hat{\beta}_{jk} - z_{1-\alpha/2}\hat{\sigma}_{\beta_{jk}}, \hat{\beta}_{jk} + z_{1-\alpha/2}\hat{\sigma}_{\beta_{jk}}\right)$$

The estimate of $\exp(\beta_{jk})$ is $\exp(\hat{\beta}_{jk})$, the standard error estimate of $\exp(\hat{\beta}_{jk})$ is $\left(\exp(\hat{\beta}_{jk})\cdot\hat{\sigma}_{\beta_{jk}}\right)$ and the corresponding $100(1-\alpha)\%$ Wald confidence interval for $\exp(\beta_{jk})$ is

$$\left(\exp\left(\hat{\beta}_{jk} - z_{1-\alpha/2}\hat{\sigma}_{\beta_{jk}}\right), \exp\left(\hat{\beta}_{jk} + z_{1-\alpha/2}\hat{\sigma}_{\beta_{jk}}\right)\right)$$

Note that Wald confidence intervals are based on estimates on the original scale.

4.1.3 Chi-square statistics

The hypothesis $H_{0i}: \beta_i = 0$ is tested for each non-redundant parameter using the chi-square statistic

$$c_i = \left(\frac{\hat{\beta}_i}{\hat{\sigma}_{\beta_i}}\right)^2$$

which has an asymptotic chi-square distribution with 1 degree of freedom, χ_1^2.

Note that the chi-square statistic will not be calculated for the scale parameters ϕ (τ), even it is estimated by ML method.

For ordinal multinomial distribution, the hypothesis $H_{0j}: \psi_j = 0, j = 1, \cdots, J-1$, and $H_{0i}: \beta_j = 0, i = 1, \cdots, p$, are tested for threshold parameters and regression parameters using the chi-square statistic

$$c_{\psi_j} = \left(\frac{\hat{\psi}_j}{\hat{\sigma}_{\psi_j}}\right)^2 \text{ and } c_{\beta_i} = \left(\frac{\hat{\beta}_i}{\hat{\sigma}_{\beta_i}}\right)^2, \text{ respectively.}$$

Similarly, c_{ψ_j} and c_{β_i} has an asymptotic chi-square distribution with 1 degree.

For nominal multinomial distribution, the test statistics for the hypothesis $H_{0,jk}: \beta_{jk} = 0, j = 1, \cdots, J - 1, k = 1, \cdots, p$, is

$$c_{\beta_{jk}} = \left(\frac{\hat{\beta}_{jk}}{\hat{\sigma}_{\beta_{jk}}}\right)^2$$

which has an asymptotic chi-square distribution with 1 degree of freedom.

Chi-square statistics and their corresponding p-value are set to system missing values for redundant parameter estimates.

Note that Chi-square statistics are based on estimates on the original scale.

4.1.4 P-values

The general form for calculating p-values for the tests above and below, given a test statistic T and a corresponding cumulative distribution function G as specified above, is defined as $p = 1 - G(T)$. For example, the p-value for χ_1^2 test $H_0: \beta = 0$ is $p = 1 - prob(\chi_1^2 \leq c)$.

4.2 Tests

After estimating parameters and calculating relevant statistics, several tests for the given model are performed: (1) Lagrange multiplier (LM) test for fixed ϕ value or k value for negative binomial distribution; (2) model fitting test; (3) model effect tests; (4) custom tests; and (5) estimating marginal means (EMMEANS).

4.2.1 Lagrange multiplier test

If the scale parameter ϕ for normal, inverse Gaussian, gamma and Tweedie distributions is set to a fixed value or specified by the deviance or Pearson chi-square divided by the degrees of freedom (the latter case, ϕ can be considered as a fixed value), or an ancillary parameter k is set to a fixed value for negative binomial, then the LM test is offered to assess the validity of the value. For a fixed ϕ value which can be any positive value or a fixed k value other than 0, the test statistic is defined as

$$T_{LM} = \frac{s^2}{A}$$

where $s = \partial\ell/\partial\tau$ (Table 7) and $A = -\left(\frac{\partial^2\ell}{\partial\tau^2}\right) - \left(-\frac{\partial^2\ell}{\partial\tau\partial\boldsymbol{\beta}^T}\right)\left(-\frac{\partial^2\ell}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)^{-}\left(-\frac{\partial^2\ell}{\partial\boldsymbol{\beta}\partial\tau}\right)$ (Table 8) evaluated at $\widehat{\boldsymbol{\beta}}$ and fixed ϕ or k value ($\tau = \ln(\phi)$, or $\ln(k)$). Then T_{LM} is an asymptotic chi-square with 1 degree of freedom. The p-value can be calculated accordingly.

If the ancillary parameter k for negative binomial is set to a fixed value, the LM test is provided to assess the validity of the value.

For k is set to 0, the LM test statistic is based on following auxiliary OLS regression (Cameron and Trivedi, 1998).

$$\frac{(y_i - \hat{\mu}_i)^2 - y_i}{\hat{\mu}_i} = \alpha\hat{\mu}_i + \varepsilon_i$$

where $\hat{\mu}_i = g^{-1}(\boldsymbol{x}_i^T\boldsymbol{\beta} + o_i)$ and ε_i is an error term. Let the response of the above OLS regression $[(y_i - \hat{\mu}_i)^2 - y_i/\hat{\mu}_i]$ be z_i and the explanatory variable $\hat{\mu}_i$ be w_i. The estimate of the above regression parameter α and the standard error of the estimate of α are

$$\hat{\alpha} = \frac{\sum_i^n f_i w_i z_i}{\sum_i^n f_i w_i^2} \quad \text{and} \quad \hat{\sigma}_\alpha = \sqrt{\frac{\frac{1}{N-1}\sum_i^n f_i(z_i - \hat{\alpha} w_i)^2}{\sum_i^n f_i w_i^2}}$$

Then the LM test statistic is *z* statistic

$$z = \frac{\hat{\alpha}}{\hat{\sigma}_\alpha}$$

and it has an asymptotically standard normal distribution under the null hypothesis of equidispersion in a Poisson model ($H_0\colon k = 0$). The alternative hypothesis can be one-sided overdispersion ($H_a\colon k > 0$), underdispersion ($H_a\colon k < 0$) or two-sided non-directional ($H_a\colon k \neq 0$) with the variance function of $V(\mu) = \mu + k\mu^2$. The calculation of p-value depends on the alternative. For $H_a\colon k > 0$, p-value $= 1 - \Phi(z)$, where $\Phi(\cdot)$ is the cumulative probability of a standard normal distribution; for $H_a\colon k < 0$, p-value $= \Phi(z)$; and for $H_a\colon k \neq 0$, p-value$= 2\big(1 - \Phi(|z|)\big)$. We will show all three p-values.

Implementation note:

The *z* statistic can be calculated in one data pass under map/reduce environment as follows:

$$z = \sqrt{\frac{(N-1)S_2^2}{S_1 S_3 - S_2^2}}$$

where $S_1 = \sum_{i=1}^{n} f_i z_i^2$, $S_2 = \sum_{i=1}^{n} f_i z_i \mu_i$ and $S_3 = \sum_{i=1}^{n} f_i \hat{\mu}_i^2$.

In each mapper, compute $\hat{\mu}_i, z_i$ and also accumulate S_1, S_2 and S_3, then in the reducer, combine all parts of S_1, S_2 and S_3 from all mappers to compute the *z* statistic.

4.2.2 Model fitting test

The model fitting omnibus test is based on –2 log-likelihood values for the model under consideration and the initial model. For the model under consideration, the value of –2 log-likelihood is

$$-2\ell(\widehat{\boldsymbol{\beta}}).$$

Let initial model be the intercept-only model if intercept is in the considered model or the null model otherwise.

- For the intercept-only model, let the value of –2 log-likelihood is

$$-2\ell(\widehat{\boldsymbol{\beta}}_0).$$

- For the null model, let the value of –2 log-likelihood is

$$-2\ell(\mathbf{0}).$$

(a) The omnibus (or global) test statistic for all distribution except multinomial distribution is

$$S = 2\left(\ell(\widehat{\boldsymbol{\beta}}) - \ell(\widehat{\boldsymbol{\beta}}_0)\right) \quad \text{for the intercept only model or}$$
$$S = 2\left(\ell(\widehat{\boldsymbol{\beta}}) - \ell(\mathbf{0})\right) \quad \text{for the null model}$$

S has an asymptotic chi-square distribution with r degrees of freedom, equal to the difference in the number of valid parameters between the model under consideration and the initial model. $r = p_x - 1$ for the intercept-only model; $r = p_x$ for the null model**.** The *p*-values then can be calculated accordingly.

(b) For ordinal multinomial model,

- The value of –2 log-likelihood for the model under consideration is

$$-2\ell(\widehat{\boldsymbol{B}}).$$

- The value of –2 log-likelihood for the thresholds-only model is

$-2\ell(\widehat{\boldsymbol{B}}_0)$.

where $\widehat{\boldsymbol{B}}_0 = \left(\widehat{\boldsymbol{\psi}}^{(0)T}, \mathbf{0}^{\mathrm{T}}\right)^{\mathrm{T}}$ is the parameters estimated for thresholds-only model.

Then the omnibus test statistic is

$$S = 2\left(\ell(\widehat{\boldsymbol{B}}) - \ell(\widehat{\boldsymbol{B}}_0)\right),$$

and it is asymptotically chi-square distributed with p_x degrees of freedom.

(c) For nominal multinomial model,

- The value of –2 log-likelihood for the model under consideration is

  $-2\ell(\widehat{\boldsymbol{\beta}})$.

- The value of –2 log-likelihood for the intercept-only model is

  $-2\ell(\widehat{\boldsymbol{\beta}}_0)$.

  where $\widehat{\boldsymbol{\beta}}_0 = \left(\widehat{\boldsymbol{\beta}}_1^{(0)\mathrm{T}}, \cdots, \widehat{\boldsymbol{\beta}}_{J-1}^{(0)\mathrm{T}}\right)^{\mathrm{T}}$ is the parameters estimated for intercept-only model the value of –2 log-likelihood for the null model is

  $-2\ell(\mathbf{0})$,

  where $\ell(\mathbf{0}) = \ln\left(\frac{1}{J}\right)\sum_{i=1}^{n}\frac{f_i\omega_i}{\phi} + c$, and c is computed based on subpopulations (see Section 4.3.3.2 for details.)

Then the omnibus test statistics is

$$\begin{array}{ll} S = 2\left(\ell(\widehat{\boldsymbol{\beta}}) - \ell(\widehat{\boldsymbol{\beta}}_0)\right) & \text{for the intercept only model or} \\ S = 2\left(\ell(\widehat{\boldsymbol{\beta}}) - \ell(\mathbf{0})\right) & \text{for the null model} \end{array}$$

and it is asymptotically chi-square distributed with r degrees of freedom. $r = \sum_{j=1}^{J-1}\left(p_x^j - 1\right)$ for the intercept-only model, where p_x^j is the number of non-redundant parameters in $\boldsymbol{\beta}_j$; $r = \sum_{j=1}^{J-1} p_x^j$ for the null model**.**

When calculating the value of –2 log-likelihood of initial model we need to setup the rules to handle the scale parameter ϕ or the ancillary parameter k in the initial model and they depend on how it is handled in the model under consideration.

(1) If the scale parameter ϕ or the ancillary parameter k is estimated by the ML method in the model under consideration, then it will also be estimated by the ML method in the initial model.

(2) If the scale parameter ϕ or the ancillary parameter k is held fixed in the model under consideration, then the same value is fixed in the initial model.

(3) If the scale parameter ϕ is specified by the deviance or Pearson chi-square divided by degrees of freedom in the model under consideration, then that value will be held fixed in the initial model. Note that the log likelihood for the model under consideration, would be adjusted, i.e., based on $\phi = \hat{\phi}$, so the log likelihoods for both models (the model under consideration and initial model) are calculated based on the same scale parameter value.

The details of the calculation of initial model are given in Section 3.1.3.3. Please note that for a part of null and intercept-only models, there are no closed form solutions, thus approximate models will be used. Thus, when the initial model is different from the model under consideration and the approximate initial model is used, then a warning message, such as “The omnibus test may not be accurate for the approximate initial model being used”, would be displayed.

4.2.3 Tests for model effects

For each regression effect specified in the model, two analyses can be conducted: type I analysis and type III analysis. The can request to do one of them, both of them or none.

4.2.3.1 Type I analysis

Type I analysis consists of fitting a sequence of models, starting with the null model as the baseline model (for all distributions except ordinal multinomial), adding one additional effect, which can be an intercept term (if there is one), covariates, factors and interactions, of the model on each step. For ordinal multinomial model, the baseline model will be thresholds-only model. So it depends on the order of effects specified in the model. On the other hand, type III analysis will not depend on the order of effects. The reason for using the null model as the baseline model is to obtain the chi-square statistic for the first parameter $\boldsymbol{\beta}_1$ which might be for an intercept or the first predictor variable.

(a) All distributions except multinomial distributions

For each effect specified in the model, type I test matrix $\boldsymbol{L}_i$ is constructed and H_0: $\boldsymbol{L}_i\boldsymbol{\beta} = \boldsymbol{0}$ is tested. The Wald statistic is defined by

$$S = \left(\boldsymbol{L}_i\widehat{\boldsymbol{\beta}}\right)^{\boldsymbol{T}}(\boldsymbol{L}_i\boldsymbol{\Sigma}\boldsymbol{L}_i^T)^{-}\boldsymbol{L}_i\widehat{\boldsymbol{\beta}}.$$

$\boldsymbol{L}_i$ is a $r \times p$ full row rank hypothesis matrix and is constructed based on the generating matrix $\boldsymbol{H}_\omega = (\boldsymbol{X}^T\boldsymbol{\Omega}\boldsymbol{X})^{-}\boldsymbol{X}^T\boldsymbol{\Omega}\boldsymbol{X}$, where $\boldsymbol{\Omega}$ is the scale weight matrix with the i^{th} diagonal element being ω_i and such that $\boldsymbol{L}_i\boldsymbol{\beta}$ is estimable. $\widehat{\boldsymbol{\beta}}$ is the maximum likelihood estimate and $\boldsymbol{\Sigma}$ is the estimated covariance matrix ($\boldsymbol{\Sigma}$ could be $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$). The asymptotic distribution of S is $\chi^2_{r_C}$, where $r_C = \text{rank}(\boldsymbol{L}_i\boldsymbol{\Sigma}\boldsymbol{L}_i^T)$, If $r_C < r$, $(\boldsymbol{L}_i\boldsymbol{\Sigma}\boldsymbol{L}_i^T)^{-}$ is a generalized inverse such that Wald tests are effective for restricted set of hypothesis $\boldsymbol{L}_i^C\boldsymbol{\beta}$ containing a particular subset C of independent rows from H_0. See Fang and Spisic (2004) for details. Then the p-values can be calculated accordingly.

Note that for type I analysis, $\boldsymbol{L}_i$ depends on the order of effects specified in the model, but for type III analysis, it does not. If such a matrix cannot be constructed, the effect is not testable. See Chiu (1995a, b) and Zhong (2006a) for computational details on construction of type I and III test matrices.

(b) Ordinal multinomial distributions

For ordinal multinomial model, first consider partition more general test matrix $\boldsymbol{L} = \left(\boldsymbol{L}(\boldsymbol{\psi}), \boldsymbol{L}(\boldsymbol{\beta})\right)$, where $\boldsymbol{L}(\boldsymbol{\psi}) = \left(\boldsymbol{l}_1, \cdots, \boldsymbol{l}_{J-1}\right)$ consists of columns corresponding to threshold parameters and $\boldsymbol{L}(\boldsymbol{\beta})$ be the part of $\boldsymbol{L}$ corresponding to regression parameters. Consider matrix $\boldsymbol{L}_0 = \left(\boldsymbol{l}_0, \boldsymbol{L}(\boldsymbol{\beta})\right)$ where the column vectors corresponding to threshold parameters are replaced by their sum $\boldsymbol{l}_0 = \sum_{j=1}^{J-1}\boldsymbol{l}_j$. Then $\boldsymbol{LB}$ is estimable if and only if $\boldsymbol{L}_0 = \boldsymbol{L}_0\boldsymbol{H}_\omega$, where $\boldsymbol{H}_\omega = (\boldsymbol{X}_1^T\boldsymbol{\Omega}\boldsymbol{X}_1)^{-}\boldsymbol{X}_1^T\boldsymbol{\Omega}\boldsymbol{X}_1$ is a $(1+p) \times (1+p)$ matrix constructed using $\boldsymbol{X}_1 = (\boldsymbol{1}, -\boldsymbol{X})$. The Wald statistic for testing $\boldsymbol{LB} = \boldsymbol{0}$, where $\boldsymbol{L}$ is a $r \times (J - 1 + p)$ full row rank hypothesis matrix is defined by

$$S = \left(\boldsymbol{L}\widehat{\boldsymbol{B}}\right)^T(\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^{\boldsymbol{T}})^{-}\boldsymbol{L}\widehat{\boldsymbol{B}}$$

where $\widehat{\boldsymbol{B}} = \left(\widehat{\boldsymbol{\psi}}^T, \widehat{\boldsymbol{\beta}}^T\right)^T$ is the maximum likelihood estimate and $\boldsymbol{\Sigma}$ is the estimated covariance ($\boldsymbol{\Sigma}$ could be $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$). The asymptotic distribution of S is $\chi^2_{r_C}$, where $r_C = \text{rank}(\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{L}^T)$.

For each effect specified in the model excluding threshold parameters, type I test matrix $\boldsymbol{L}_i$ is constructed and H_0: $\boldsymbol{L}_i\boldsymbol{B} = 0$ is tested. Construction of matrix $\boldsymbol{L}_i$ is based on matrix $\boldsymbol{H}_\omega = (\boldsymbol{X}_1^T\boldsymbol{\Omega}\boldsymbol{X}_1)^{-}\boldsymbol{X}_1^T\boldsymbol{\Omega}\boldsymbol{X}_1$ and such that $\boldsymbol{L}_i\boldsymbol{B}$ is estimable. Thus, the way to construct $\boldsymbol{L}_i$ (type I and III) for ordinal multinomial is the same as that for other distributions. Note that the threshold-parameter effect is not tested for both type I and III analyses.

(c) Nominal multinomial distributions

For each effect specified in the model, $\boldsymbol{L}_i$ is constructed based on the generating matrix $\boldsymbol{H}_\omega = (\boldsymbol{X}^T\boldsymbol{\Omega X})^{-}\boldsymbol{X}^T\boldsymbol{\Omega X}$, where $\boldsymbol{\Omega}$ is the scale weight matrix with the ith diagonal element being ω_i and such that $\boldsymbol{L}_i\boldsymbol{\beta}$ is estimable.

$$S = \left(\boldsymbol{L}_i'\widehat{\boldsymbol{\beta}}\right)^{\boldsymbol{T}}(\boldsymbol{L}_i'\boldsymbol{\Sigma L}_i'^{T})^{-}\boldsymbol{L}_i'\widehat{\boldsymbol{\beta}}$$

where $\boldsymbol{L}_i' = \boldsymbol{I}_{J-1} \otimes \boldsymbol{L}_i$ and $r_C = \text{rank}(\boldsymbol{L}_i'\boldsymbol{\Sigma L}_i'^{T})$; $\boldsymbol{\Sigma}$ could be $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$. The asymptotic distribution of S is $\chi^2_{r_C}$.

4.2.3.2 Type III analysis

The computation of Wald statistics for type III analysis is similar to that for type I analysis. The only difference is that type III L matrix is constructed.

4.2.4 Custom tests

Contrasts defined as linear combination of regression parameters can be tested. For a user specified $\boldsymbol{L}$ and $\boldsymbol{K}$, the hypothesis H_0: $\boldsymbol{L\beta} = \boldsymbol{K}$ is tested only when each row of the $\boldsymbol{L}$ matrix is checked for estimablility (i.e. check if $\boldsymbol{LH}_\omega = \boldsymbol{L}$ where $\boldsymbol{H}_\omega = (\boldsymbol{X}^T\boldsymbol{\Omega X})^{-}\boldsymbol{X}^T\boldsymbol{\Omega X}$, and $\boldsymbol{\Omega}$ is the scale weight matrix with the ith diagonal element is ω_i). Then test statistics, exponential estimation and multiple test p-value adjustment are the three subsections to be discussed. For checking on estimability for ordinal and nominal multinomial model, please see Section 4.2.3.1 for details.

4.2.4.1 Test statistics

The test statistics used is Wald statistics (see Section 4.2.3). Then the *p*-values are calculated accordingly.

4.2.4.2 Exponential estimation

If $\boldsymbol{L}$ is a $1 \times p$ row vector, we can calculate the estimate of $\boldsymbol{L\beta}$, its approximate standard error and its Wald confidence interval. In the meantime, for logistic regression or log-linear models, we can also calculate $\exp(\boldsymbol{L\beta})$, its standard error, and its confidence interval. Note for other models, $\exp(\boldsymbol{L\beta})$ might not make sense. The following table is shown the formulae:

Table 11: Estimate, Standard Error and Wald Conference Interval for $\boldsymbol{L\beta}$ and $\exp(\boldsymbol{L\beta})$

	$\boldsymbol{L\beta}$	$\exp(\boldsymbol{L\beta})$
Estimate	$\boldsymbol{L\hat{\beta}}$	$\exp(\boldsymbol{L\hat{\beta}})$
Std. Error	$\hat{\sigma}_{\boldsymbol{L\beta}} = \sqrt{\boldsymbol{L\Sigma L^T}}$	$\left(\exp(\boldsymbol{L\hat{\beta}}) \cdot \hat{\sigma}_{\boldsymbol{L\beta}}\right)$
Wald confidence interval	$\left(\boldsymbol{L\hat{\beta}} - z_{1-\alpha/2}\hat{\sigma}_{\boldsymbol{L\beta}}, \quad \boldsymbol{L\hat{\beta}} + z_{1-\alpha/2}\hat{\sigma}_{\boldsymbol{L\beta}}\right)$	$\left(\exp\left(\boldsymbol{L\hat{\beta}} - z_{1-\alpha/2}\hat{\sigma}_{\boldsymbol{L\beta}}\right), \quad \exp\left(\boldsymbol{L\hat{\beta}} + z_{1-\alpha/2}\hat{\sigma}_{\boldsymbol{L\beta}}\right)\right)$

4.2.4.3 Multiple test *p*-value adjustment

The above hypothesis $H_0: \boldsymbol{L\beta} = \boldsymbol{K}$ can be tested using the multiple row hypotheses testing technique. Let $\boldsymbol{l}_i^T$ be the *i*th row vector of matrix $\boldsymbol{L}$ and k_i be the *i*th element of vector $\boldsymbol{K}$. The *i*th row hypothesis is $H_{0i}: \boldsymbol{l}_i^T \boldsymbol{\beta} = k_i$. Testing H_0 is the same as testing multiple non-redundant row hypotheses $\{H_{0i}^*\}_{i=1}^R$ simultaneously, where *R* is the number of non-redundant row hypotheses, and H_{0i}^* represents the *i*th non-redundant hypothesis. A hypothesis H_{0i} is redundant if there exists another hypothesis $H_{0j}, j \neq i$ such that $\boldsymbol{l}_i = c\boldsymbol{l}_j, k_i = \mathrm{c}k_j,\ c \neq 0$.

For each individual hypothesis H_{0i}, test statistics can be calculated. Let p_i denotes the *p*-value for testing H_{0i}, and p_i^* denotes the adjusted *p*-value. The conclusion from the multiple testing is, at level α (the family-wise type I error),

reject $H_{0i}: \boldsymbol{l}_i^T \boldsymbol{\beta} = k_i$, if $p_i^* < \alpha$;

reject $H_0: \boldsymbol{L\beta} = \boldsymbol{K}$, if $\min_i(p_i^*) < \alpha$.

There are different methods to adjust *p*-values. Five methods are provided here. Please note that if the adjusted *p*-value is bigger than 1, it is set to 1 in all the methods.

(a) LSD (Least Significant Difference**)**

The adjusted *p*-values are the same as the original *p*-values: $p_i^* = p_i$.

(b) Bonferroni

The adjusted *p*-values are $p_i^* = Rp_i$.

(c) Sidak

The adjusted *p*-values are $p_i^* = 1 - (1 - p_i)^R$.

(d) Sequential Bonferroni

In sequential test, the p-values are first ordered from the smallest to the biggest, and then adjusted depending on the order. Let the ordered p-values for the non-redundant row hypotheses be $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(R)}$ with corresponding non-redundant hypotheses being be $H_{0(1)} \leq H_{0(2)} \leq \cdots \leq H_{0(R)}$.

The adjusted p-value of $p_{(i)}$ is $p^*_{(i)} = \begin{cases} Rp_{(1)} & \text{if } i = 1 \\ \max\left((R-i+1)p_{(i)}, p^*_{(i-1)}\right) & \text{if } i \geq 2 \end{cases}$.

Note: if a row hypothesis is made redundant by $H^*_{0(i)}$, the p-value and adjusted p-value of this row are the same as that of $H^*_{0(i)}$. This applies to both sequential Bonferroni and Sidak tests.

(e) Sequential Sidak

The adjusted p-value of $p_{(i)}$ is $p^*_{(i)} = \begin{cases} 1-\left(1-p_{(1)}\right)^R & \text{if } i = 1 \\ \max\left(1-\left(1-p_{(i)}\right)^{R-i+1}, p^*_{(i-1)}\right) & \text{if } i \geq 2 \end{cases}$.

See Fang and Spisic (2004) for comparison of adjustment methods.

Note that if confidence intervals are also calculated for the above hypothesis, then adjusting confidence intervals is required to correspond to adjusted p-values. The only item needed to be adjusted in the confidence intervals is the critical value from the standard normal distribution. Assume that the original critical value is $z_{1-\alpha/2}$ and the adjusted critical value is z^*.

(a) LSD (Least Significant Difference)

The adjusted critical value is $z^* = z_{1-\frac{\alpha}{2}}$.

(b) Bonferroni

The adjusted critical value is $z^* = z_{1-\frac{\alpha}{2R}}$.

(c) Sidak

The adjusted critical value is $z^* = z_{1-\frac{1-(1-\alpha)^{1/R}}{2}}$.

(d) Sequential Bonferroni

The adjusted z^* values will correspond to the ordered adjusted p-values $p_{(1)}, p_{(2)}, \cdots, p_{(R)}$ as follows:

$$z_{(i)}^* = \begin{cases} z_{1-\frac{\alpha}{2R}} & \text{if } i = 1 \\ \min\left(z_{1-\frac{\alpha}{2(R-i+1)}}, z_{(i-1)}^*\right) & \text{if } i \geq 2 \end{cases}.$$

(e) Sequential Sidak

$$z_{(i)}^* = \begin{cases} z_{1-\frac{1-(1-\alpha)^{1/R}}{2}} & \text{if } i = 1 \\ \min\left(z_{1-\frac{1-(1-\alpha)^{1/(R-i+1)}}{2}}, z_{(i-1)}^*\right) & \text{if } i \geq 2 \end{cases}.$$

4.2.5 EMMEANS

There are two types of estimated marginal means (EMMEANS) calculated here. One corresponds to the specified factors for the linear predictor of the model and the other corresponds to those for the response of the model.

EMMEANS are based on the estimated cell means. For a given fixed set of factors, or their interactions, we estimate marginal means as the mean value averaged over all cells generated by the rest of the factors in the model. Covariates may be fixed at any specified value. If not specified, the value for each covariate is set to its overall mean estimate.

For ordinal and nominal multinomial model, EMMEANS are not available.

4.2.5.1 EMMEANS for the linear predictor

(a) Calculating EMMEANS for the linear predictor

EMMEANS for the linear predictor are based on the link function transformation. They are computed for the linear predictor. Since the given model with respect to the linear predictor is a linear model (i.e. the model is $\boldsymbol{\eta} = \boldsymbol{X\beta} + \text{offset}$), so the way to construct $\boldsymbol{L}$ is the same as that for the GLM procedure. Each EMMEAN for the linear predictor is constructed in the form $\boldsymbol{L}\widehat{\boldsymbol{\beta}}$ such that $\boldsymbol{L\beta}$ is estimable.

Briefly, for a given set of factors in the model, a vector of EMMEANS for the linear predictor is created for all combined levels of the factors. Assume there are r levels. This $r \times 1$ vector can be expressed in the form $\widehat{\boldsymbol{v}} = \boldsymbol{L}\widehat{\boldsymbol{\beta}}$ where each row of $\boldsymbol{L}$ matrix is generated as described above. Variance matrix of $\widehat{\boldsymbol{v}}$ is then computed by the following formula

$$\boldsymbol{V}(\widehat{\boldsymbol{v}}) = \boldsymbol{L\Sigma L}^T.$$

Note that $\boldsymbol{\Sigma}$ could be $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$. The standard error for the *j*th element of $\hat{\boldsymbol{v}}$ is the square root of the *j*th diagonal element of $\boldsymbol{V}(\hat{\boldsymbol{v}})$. Let the *j*th element of $\hat{\boldsymbol{v}}$ and its standard error be $\hat{v}_j$ and $\hat{\sigma}_{v_j}$, respectively, then the corresponding $100(1 - \alpha)\%$ Wald confidence interval for $v_j, j = 1, \cdots, r$ is given by

$$\left(\hat{v}_j - z_{1-\alpha/2}\hat{\sigma}_{v_j}, \hat{v}_j + z_{1-\alpha/2}\hat{\sigma}_{v_j}\right).$$

(b) Comparing EMMEANS for the linear predictor

We can compare EMMEANS for the linear predictor based on a selected contrast type which a set of contrasts for the factor is created. Let this set of contrasts define matrix $\boldsymbol{C}$ used for testing the following hypothesis $H_0: \boldsymbol{Cv} = \boldsymbol{0}$ (an overall test). A Wald statistic is used for testing given set of contrasts for the factor as follows:

$$\mathrm{S} = (\boldsymbol{C}\hat{\boldsymbol{v}})^T(\boldsymbol{C}\boldsymbol{V}(\hat{\boldsymbol{v}})\boldsymbol{C}^T)^{-}(\boldsymbol{C}\hat{\boldsymbol{v}}).$$

Asymptotic distribution of the Wald statistic is chi-square with r_l degrees of freedom, where $r_l = \mathrm{rank}(\boldsymbol{C}\boldsymbol{V}(\hat{\boldsymbol{v}})\boldsymbol{C}^T)$. The *p*-value can be calculated accordingly. Note that the adjusted *p*-value based on multiple test *p*-value adjustments (see Section 4.2.4.3) won't be given.

Each row $\boldsymbol{c}_i^T$ of matrix $\boldsymbol{C}$ is also tested separately (individual tests). Estimate for the *i*th row is given by $\boldsymbol{c}_i^T\hat{\boldsymbol{v}}$ and its standard error by $\sqrt{\boldsymbol{c}_i^T\boldsymbol{V}(\hat{\boldsymbol{v}})\boldsymbol{c}_i}$. The corresponding $100(1 - \alpha)\%$ Wald confidence interval for $\boldsymbol{c}_i^T\boldsymbol{v}$ is given by

$$\left(\boldsymbol{c}_i^T\hat{\boldsymbol{v}} - z_{1-\alpha/2}\sqrt{\boldsymbol{c}_i^T\boldsymbol{V}(\hat{\boldsymbol{v}})\boldsymbol{c}_i}, \quad \boldsymbol{c}_i^T\hat{\boldsymbol{v}} + z_{1-\alpha/2}\sqrt{\boldsymbol{c}_i^T\boldsymbol{V}(\hat{\boldsymbol{v}})\boldsymbol{c}_i}\right).$$

The Wald statistic for $H_0: \boldsymbol{c}_i^T\boldsymbol{v} = 0$ is

$$S_i = \left(\frac{\boldsymbol{c}_i^T\hat{\boldsymbol{v}}}{\boldsymbol{c}_i^T\boldsymbol{V}(\hat{\boldsymbol{v}})\boldsymbol{c}_i}\right)^2.$$

And it has an asymptotic chi-square distribution with 1 degree of freedom. The *p*-values can be calculated accordingly. In addition, the adjusted *p*-values can also computed, see Section 4.2.4.3 for details.

Note:

- The usual contrast types used in $\boldsymbol{C}$ are included
  - Deviation

- Simple
- Helmert
- Difference
- Polynomial
- Repeated

See Appendix of SPSS Advanced Statistics 7.5 (1997) for definitions of these contrasts. Note the definition of deviation is revised: **each level of the factor is compared to the grand mean**.

- In addition, we would like to offer pair-wise contrast (the differences between EMMEANS for each pair of levels for the effect), $\boldsymbol{C}$ can be constructed similarly as that in GLM procedure.

4.2.5.2 EMMEANS for the response

EMMEANS for the response are based on the original scale of the dependent variable except for the binomial response with events/trials format (see note below). They can be defined as the estimator of the expected response for a subject conditional on his/her belonging to a specified effect and having the averages of covariates.

(a) Calculating EMMEANS for the response

The way to construct EMMEANS for the response is based on EMMEANS for the linear predictor. Let $\widehat{\boldsymbol{M}}_c$ be EMMEANS for the response and it is defined as

$$\widehat{\boldsymbol{M}}_c = g^{-1}\left(\boldsymbol{L}\widehat{\boldsymbol{\beta}}\right) = g^{-1}(\widehat{\boldsymbol{v}}).$$

The variance of EMMEANS for the response is

$$\boldsymbol{V}\left(\widehat{\boldsymbol{M}}_c\right) = \text{diag}\left(\frac{\partial g^{-1}\left(\hat{v}_j\right)}{\partial \hat{v}_j}\right)\boldsymbol{L\Sigma L}^T \text{diag}\left(\frac{\partial g^{-1}\left(\hat{v}_j\right)}{\partial \hat{v}_j}\right)$$

Where $\text{diag}\left(\partial g^{-1}\left(\hat{v}_j\right)/\partial \hat{v}_j\right)$ a $r \times r$ matrix and $\partial g^{-1}\left(\hat{v}_j\right)/\partial \hat{v}_j$ is the derivative of the inverse of the link with respect to the j^{th} value in $\widehat{\boldsymbol{v}}$ and $\partial g^{-1}\left(\hat{v}_j\right)/\partial \hat{v}_j = 1/g'\left(\widehat{M}_{cj}\right)$ where $g'\left(\widehat{M}_{cj}\right)$ is from Table 5. The standard error for the j^{th} element of $\widehat{\boldsymbol{M}}_c$ and the corresponding confidence interval are calculated similar to those of $\widehat{\boldsymbol{v}}$**,** see Section 4.2.5.1-(a) for details.

Note:

$\widehat{\boldsymbol{M}}_c$ is EMMEANS for the proportion, not for the number of events when r and m (events/trials) variables are used for the binomial distribution. See P. 62 for discussion about binomial response with events/trials format.

(b) Comparing EMMEANS for the response

It is similar to comparing EMMEANS for the linear predictor, just replace $\widehat{\boldsymbol{v}}$ with $\widehat{\boldsymbol{M}}_c$ and $\boldsymbol{V}(\widehat{\boldsymbol{v}})$ with $\boldsymbol{V}\left(\widehat{\boldsymbol{M}}_c\right)$. See Section 4.2.5.1-(b) for details.

4.2.5 Tests on original scale

(a) Lagrange multiplier test (Section 4.2.1)

(b) Model fitting test (Section 4.2.2)

All statistics calculated are the same on either original or transformed scale. Since parameters have been estimated based on transformed scale and a lot of values are available, those statistics should be calculated based on transformed scale.

(c) Tests for model effects (Section 4.2.3)

(d) EMMEANS and custom tests (Section 4.2.4)

For each effect specified in the model, type I or III test matrix $\boldsymbol{L}$ is constructed from the generating matrix, $\boldsymbol{H}_\omega = (\boldsymbol{X}^T\boldsymbol{\Omega X})^-\boldsymbol{X}^T\boldsymbol{\Omega X}$. We may have trouble to calculate $\boldsymbol{H}_\omega$ directly. Use the transformed variables, we will first calculate $\boldsymbol{H}_\omega^* = (\boldsymbol{X}^{*T}\boldsymbol{\Omega X}^*)^-\boldsymbol{X}^{*T}\boldsymbol{\Omega X}^*$. Since $\boldsymbol{H}_\omega = \boldsymbol{A}\boldsymbol{H}_\omega^*\boldsymbol{A}^{-1}$, we can obtain $\boldsymbol{H}_\omega$ from $\boldsymbol{H}_\omega^*$, then construct type I or III test matrix $\boldsymbol{L}_i$ for the *i*th effect based on original scale from $\boldsymbol{H}_\omega$.

For ordinal multinomial, use $\boldsymbol{H}_\omega = (\boldsymbol{X}_1^T\boldsymbol{\Omega X}_1)^-\boldsymbol{X}_1^T\boldsymbol{\Omega X}_1$, $\boldsymbol{H}_{\boldsymbol{\omega}}^* = (\boldsymbol{X}_1^{*T}\boldsymbol{\Omega X}_1^*)^-\boldsymbol{X}_1^{*T}\boldsymbol{\Omega X}_1^*$ and $\boldsymbol{H}_\omega = \boldsymbol{T}\boldsymbol{H}_\omega^*\boldsymbol{T}^{-1}$ to construct type I or III test matrix $\boldsymbol{L}$.

For nominal multinomial, use $\boldsymbol{H}_\omega = (\boldsymbol{X}^T\boldsymbol{\Omega X})^-\boldsymbol{X}^T\boldsymbol{\Omega X}$, $\boldsymbol{H}_\omega^* = (\boldsymbol{X}^{*T}\boldsymbol{\Omega X}^*)^-\boldsymbol{X}^{*T}\boldsymbol{\Omega X}^*$ and $\boldsymbol{H}_\omega = \boldsymbol{T}\boldsymbol{H}_\omega^*\boldsymbol{T}^{-1}$ to construct type I or III test matrix $\boldsymbol{L}$.

4.3 Goodness of fit

To assess goodness of fit of a given generalized linear model, we calculate three statistics: deviance, Pearson chi-square, and information criteria.

Note that all statistics are the same on either or transformed scale. Since parameters have been estimated based on the transformed scale and a lot of values are available, those statistics should be calculated based on the transformed scale.

4.3.1 Deviance

The theoretical definition of deviance is as follows:

$$D = 2\phi\big(\ell(\boldsymbol{y};\boldsymbol{y}) - \ell(\hat{\boldsymbol{\mu}};\boldsymbol{y})\big)$$

where $\ell(\hat{\boldsymbol{\mu}};\boldsymbol{y})$ is the log likelihood function expressed as the function of the predicted mean values of $\hat{\boldsymbol{\mu}}$ (calculated based on the parameter estimates) given the response variable $\boldsymbol{y}$ and $\ell(\boldsymbol{y};\boldsymbol{y})$ is the log likelihood function by replacing $\hat{\boldsymbol{\mu}}$ with $\boldsymbol{y}$. The formula used for the deviance is $\sum_{i=1}^{n} f_i d_i$ where the form of d_i for the distributions is given in the following table:

Table 9: The Form of d_i for Probability Distributions

Distribution	d_i
Normal	$\omega_i\left(y_i-\mu_i\right)^2$
Inverse Gaussian	$\dfrac{\omega_i}{y_i\mu_i^2}\left(y_i-\mu_i\right)^2$
Gamma	$2\omega_i\left\{-\ln\left(\dfrac{y_i}{\mu_i}\right)+\dfrac{y_i-\mu_i}{\mu_i}\right\}$
Negative binomial	$2\omega_i\left\{y_i\ln\left(\dfrac{y_i}{\mu_i}\right)-\left(y_i+1/k\right)\ln\left(\dfrac{y_i+1/k}{\mu_i+1/k}\right)\right\}$
Poisson	$2\omega_i\left\{y_i\ln\left(\dfrac{y_i}{\mu_i}\right)-\left(y_i-\mu_i\right)\right\}$
Binomial(*m*)	$2\omega_i^*\left\{y_i\ln\left(\dfrac{y_i}{\mu_i}\right)+\left(1-y_i\right)\ln\left(\dfrac{1-y_i}{1-\mu_i}\right)\right\}$
Tweedie	$2\omega_i\left\{\dfrac{y_i^{2-q}-\left(2-q\right)y_i\mu_i^{1-q}+\left(1-q\right)\mu_i^{2-q}}{\left(1-q\right)\left(2-q\right)}\right\}$

Note:

- When $\boldsymbol{y}$ is a binary dependent variable with 0/1 values (binomial distribution), and categorical variable (multinomial distribution), the deviance and Pearson chi-square are calculated based on the subpopulations, see Section 4.3.3.2 below.
- When y = 0 for negative binomial and Poisson distributions and y = 0 (for r = 0) or 1 (for r = m) for binomial distribution with $\boldsymbol{r/m}$ format, separate values are given the deviance. Let d_i be the deviance value for individual case i when y_i = 0 for negative binomial and Poisson and 0/1 for binomial.

Distribution	d_i
Negative binomial	$2\omega_i \dfrac{\ln(1+k\mu_i)}{k} \quad \text{if } y_i = 0$
Poisson	$2\omega_i\mu_i \quad \text{if } y_i = 0$
Binomial(m)	$\begin{cases} -2\omega_i^* \ln(1-\mu_i) & \text{if } y_i = 0 \text{ or } r_i = 0 \\ -2\omega_i^* \ln(\mu_i) & \text{if } y_i = 1 \text{ or } r_i = m_i \end{cases}$

4.3.2 Pearson chi-square

Pearson chi-square statistic is defined as follows

$$\chi^2 = \sum_{i=1}^{n} f_i \gamma_i$$

where $\gamma_i = \frac{\omega_i^*(y_i-\mu_i)^2}{V(\mu_i)}$ for binomial distribution and $\gamma_i = \frac{\omega_i(y_i-\mu_i)^2}{V(\mu_i)}$ for other distributions.

4.3.3 Scaled deviance and Pearson chi-square

The scaled deviance is $D^* = D/\phi$ and the scaled Pearson chi-square is $\chi^{2*} = \chi^2/\phi$ if ϕ is known from estimating as a parameter or setting as a fixed value.

Since the scaled deviance and Pearson chi-square statistics, have a limiting chi-square distribution with degrees of freedom equal to the number of observations (effective sample size) minus the number of non-redundant regression parameters estimated, i.e. d.f. = N – p_x, the deviance or Pearson chi-square divided by its degrees of freedom can be used as an estimate of the scale parameter ϕ for both continuous and discrete distributions.

$$\hat{\phi} = \frac{D}{N - p_x} \text{ or } \hat{\phi} = \frac{\chi^2}{N - p_x}$$

If the ancillary parameter k of negative binomial is estimated by the ML method, the scale parameter ϕ is measured by the deviance or Pearson chi-square divided by its degrees of freedom, then the degrees of freedom is $N - p_x - 1$ not usual $N - p_x$ because k is the extra parameter estimated by ML method.

Note that the values of the deviance and Pearson chi-square divided by the degrees of freedom (they might be called D/df and Pearson/df, respectively) will be computed no matter how the scale parameter is treated.

If the scale parameter is measured by the deviance or Pearson chi-square, first we assume $\phi = 1$, estimate $\hat{\boldsymbol{\beta}}$, calculate the deviance and Pearson chi-square values and obtain $\hat{\phi}$ from the above formula. Then the scaled version of both statistics is obtained by dividing the deviance and Pearson chi-square by $\hat{\phi}$. In the meantime, some statistics need to be revised. The gradient vector and the Hessian matrix are divided by $\hat{\phi}$ and the covariance matrix is multiplied by $\hat{\phi}$. Accordingly the estimated standard errors are also adjusted, the Wald confidence intervals and significance tests will be affected even the parameter estimates are not affected by $\hat{\phi}$.

Note that two log likelihood values would be displayed: original one (based on $\phi = 1$) and adjusted one (based on $\phi = \hat{\phi}$ which is plugged into the log likelihood function of the corresponding distribution).

4.3.3.1 Overdispersion

For the Poisson, binomial distributions and multinomial distribution, if the estimated scale parameter $\hat{\phi}$ is not near the assumed value of one, then the data may be overdispersed if the value is greater than one or underdispersed if the value is less than one. Overdispersion is more common in practice. The problem with overdispersion is that it may cause standard errors of the estimated parameters to be underestimated. A variable may appear to be a significant predictor, when in fact it is not.

4.3.3.2 Deviance and Pearson chi-square for binomial distribution with 0/1 binary response variable and multinomial distribution

When ***r*** and ***m*** (event/trial) variables are used for the binomial distribution, each case represents *m* Bernoulli trials. When ***y*** is a binary dependent variable with 0/1 values, each case represents a single trial. The trial can be repeated for several times with the same setting (i.e. the same values for all predictors). For example, suppose the first 10 y values are 2 1s and 8 0s and $\boldsymbol{x}$ values are the same (if recorded in events/trials format, these 10 cases is recorded as 1 case with r = 2 and m = 10), then these 10 cases should be considered from the same subpopulation. Cases with common values in the variable list that includes all predictors are regarded as coming from the same subpopulation. When the binomial distribution with binary response is used, we should calculate the deviance and Pearson chi-square based on the subpopulations. If we calculate them based on the cases, the results might not be useful.

If subpopulations are specified for the binomial distribution with 0/1 binary response variable, the data should be reconstructed from the single trial format to the events/trials format. Assume the following notations for reconstructed data:

n_s	Number of subpopulations.
r_{j1}	Sum of the product of the frequencies and the scale weights associated with y = 1 in the j^{th} subpopulation. So r_{j0} is that with y = 0 in the j^{th} subpopulation.
m_j	Total weighted observations and $m_j = r_{j1} + r_{j0}$.
y_{j1}	The proportion of 1s in the j^{th} subpopulation and $y_{j1} = r_{j1}/ m_j$.
μ_j	The fitted probability in the j^{th} subpopulation ($\hat{\mu}_j$ would be the same for each case in the j^{th} subpopulation because values for all predictors are the same for each case.)

The deviance and Pearson chi-square are defined as follows:

$$D = 2\sum_{j=1}^{n_s} m_j \left\{ y_{j1}\ln\left(\frac{y_{j1}}{\mu_j}\right) + \left(1 - y_{j1}\right)\ln\left\{\frac{1 - y_{j1}}{1 - \mu_j}\right\}\right\}$$

and

$$\chi^2 = \sum_{j=1}^{n_s} \frac{m_j\left(y_{j1} - \mu_j\right)^2}{\mu_j\left(1 - \mu_j\right)}$$

The degrees of freedom equal to the number of subpopulations minus the number of non-redundant regression parameters estimated, i.e. $\mathrm{d.f.} = n_s - p_x$ then the values of the deviance and Pearson chi-square divided by the degrees of freedom can be computed accordingly, and the corresponding estimate of the scale parameter ϕ will be

$$\hat{\phi} = \frac{D}{n_s - p_x} \text{ and } \hat{\phi} = \frac{\chi^2}{n_s - p_x}$$

For ordinal and nominal multinomial models, similarly, the data will be reconstructed based on subpopulations. Assume the following notations for reconstructed multinomial data:

n_s	Number of subpopulations.
$r_{i,j}$	Sum of the product of the frequencies and the scale weights associated with the jth category in the ith subpopulation.
m_i	Total weighted observations for the ith subpopulation and $m_i = \sum_{j=1}^{J} r_{i,j}$
$\hat{\pi}_{i,j}$	The fitted probability for the jth category in the ith subpopulation.

The deviance and Pearson chi-square are defined as follows:

$$D = 2\sum_{i=1}^{n_s}\sum_{j=1}^{J} r_{ij}\ln\left(\frac{r_{ij}}{m_i\hat{\pi}_{i,j}}\right) \text{ and } \chi^2 = \sum_{i=1}^{n_s}\sum_{j=1}^{J} \frac{\left(r_{ij} - m_i\hat{\pi}_{i,j}\right)^2}{m_i\hat{\pi}_{i,j}}$$

The degrees of freedom equal to $n_s(J-1) - d$ where $d = J - 1 + p_x$ for the ordinal multinomial distribution; $d = \sum_{j=1}^{J-1} p_x^j$ for the nominal multinomial distribution, then the values of the deviance and Pearson chi-square divided by the degrees of freedom can be computed accordingly, and the corresponding estimate of the scale parameter ϕ will be

$$\hat{\phi} = \frac{D}{n_s(J-1) - d} \text{ and } \hat{\phi} = \frac{\chi^2}{n_s(J-1) - d}$$

Notes

- For the situation of a large volume of data (“Big Data”), the number of subpopulations may be very large when all predictors are used to define subpopulations. Thus, in the Map-Reduce environment, it may cause a network traffic jam. The three alternating methods will be considered below based on their priorities from high to low.
  (1) A record is defined as a subpopulation;
  (2) All factors in predictors are used to define subpopulations; if there are no factors in predictors, a record forms a subpopulation;
  (3) All predictors first are binned into k bins; the subpopulations are defined on all predictors binned. k is set to 5 by default.

- The value of the constant c for binomial models is calculated as follow

$$c = \sum_{j=1}^{n_S} \ln\left(\frac{m_j!}{r_{j0}!\, r_{j1}!}\right).$$

The value of the constant c for ordinal and nominal multinomial models is calculated as follow,

$$c = \sum_{i=1}^{n_S} \ln\left(\frac{m_i!}{r_{i1}! \cdots r_{iJ}!}\right).$$

4.3.4 Information Criteria

Information criteria are used when comparing different models for the same data, the following criteria are given in smaller is better form. If we let ℓ be the log likelihood evaluated at $\hat{\boldsymbol{\beta}}$, the formula for various criteria are given as below. Note that for all distributions except multinomial, $d = p_x$ if only $\boldsymbol{\beta}$ is included; $d = p_x + 1$ if $\boldsymbol{\beta}$ and ϕ for normal, inverse Gaussian, gamma and Tweedie distributions or $\boldsymbol{\beta}$ and k for negative binomial distribution are included; $d = J - 1 + p_x$ for ordinal multinomial distribution; $d = \sum_{j=1}^{J-1} p_x^j$ for the nominal multinomial distribution.

(1) Akaike information criteria (AIC)

$$-2\ell + 2d$$

(2) Finite sample corrected AIC (AICC)

$$-2\ell + \frac{2dN}{(N - d - 1)}$$

(3) Bayesian information criteria (BIC)

$$-2\ell + d\ln(N)$$

(4) Consistent AIC (CAIC)

$$-2\ell + d(\ln(N) + 1)$$

Notes:

- ℓ (the full log likelihood) can be replaced with ℓ_k (the kernel of the log likelihood) depending on the user’s choice.
- If the scale parameter is specified by the deviance or Pearson chi-square, the log likelihood, ℓ or ℓ_k would be original one, i.e., based on $\phi = 1$, for fair comparison among different models.

- When *r* and *m* (event/trial) variables are used for the binomial distribution, then *N* used here would be the sum of the trials frequencies, i.e. $N = \sum_{i=1}^{n} f_i \, m_i$. In this way, the same value results whether the data are in raw, binary form (using single-trial syntax) or in summarized, binomial form (events/trials syntax).

5. Scoring

Scoring is defined as assigning one or more values to a case in a data set. Two types are considered here: predicted values and model diagnostics.

Note that if the target is not transformed, then all predicted and diagnostics values calculated are the same on either original or transformed scale. However, if the target is transformed, then predicted values of the linear predictors and the means (they are the same here) and their confidence intervals would be a different on original or transformed scale. If calculated on transformed scale, those values should be added $\bar{y}$. To avoid confusion, all values should be calculated on original scale.

5.1 Predicted values

Due to the non-linear link functions, the predicted values will be computed for the linear predictor and the mean of the response separately. Also, since estimated standard errors of predicted values of linear predictor are calculated, the confidence intervals for the mean are obtained easily.

Notice that the predicted values can be computed for the case not used in the model-building phrase. That is the response variable can be missing and the predicted values are still computed as long all the predictor variables have non-missing values in the given model. An additional requirement is that given predictor variable values could be properly parameterized by using only the existing model parameters. See Woods (2004), “Guidelines for Scoring under Various Data and Model Conditions,” for details.

5.1.1 Predicted values of the linear predictors

A predicted value of the linear predictor η_i corresponding to $\boldsymbol{x}_i$ is given by

$$\hat{\eta}_i = \boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}} + o_i.$$

For ordinal multinomial model, a predicted value of the linear predictor for category *j* $\eta_{i,j}$ corresponding to $\boldsymbol{x}_i$ is given by

$$\hat{\eta}_{i,j} = \hat{\psi}_j - \boldsymbol{x}_i^T\widehat{\boldsymbol{\beta}} + o_i, j = 1, \dots, J - 1.$$

For nominal multinomial model, a predicted value of the linear predictor for category j $\eta_{i,j}$ corresponding to $\boldsymbol{x}_i$ is given by

$$\hat{\eta}_{i,j} = \boldsymbol{x}_i^T\widehat{\boldsymbol{\beta}}_j + o_i, j = 1, \dots, J - 1.$$

5.1.2 Estimated standard errors of predicted values of linear predictor

The estimated standard error of $\hat{\eta}_i$ is given by

$$\hat{\sigma}_{\eta_i} = \sqrt{\boldsymbol{x}_i^T \Sigma \boldsymbol{x}_i}$$

where $\boldsymbol{\Sigma}$ could be $\boldsymbol{\Sigma}_m$ or $\boldsymbol{\Sigma}_r$.

For ordinal multinomial model, the estimated standard error of $\hat{\eta}_{i,j}$ is given by

$$\hat{\sigma}_{\eta_{i,j}} = \sqrt{(1, -\boldsymbol{x}_i^T)\boldsymbol{\Sigma}_j \begin{pmatrix} 1 \\ -\boldsymbol{x}_i \end{pmatrix}}, j = 1, \dots, J - 1,$$

where $\boldsymbol{\Sigma}_j$ is a reduced parameter estimates covariance (1 + p)× (1 + p) matrix from $\boldsymbol{\Sigma}$. Suppose $\boldsymbol{\Sigma}$ for ordinal multinomial models has the following form:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\psi,\psi} & \boldsymbol{\Sigma}_{\psi,\boldsymbol{\beta}} \\ \boldsymbol{\Sigma}_{\boldsymbol{\beta},\psi} & \boldsymbol{\Sigma}_{\boldsymbol{\beta},\boldsymbol{\beta}} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \sigma_{1,1} & \cdots & \sigma_{1,(J-1)} \\ \vdots & \ddots & \vdots \\ \sigma_{(J-1),1} & \cdots & \sigma_{(J-1,J-1)} \end{bmatrix} & \begin{bmatrix} \sigma_{1,J} & \cdots & \sigma_{1,(J-1+p)} \\ \vdots & \ddots & \vdots \\ \sigma_{(J-1),J} & \cdots & \sigma_{(J-1,J-1+p)} \end{bmatrix} \\ \begin{bmatrix} \sigma_{J,1} & \cdots & \sigma_{J,(J-1)} \\ \vdots & \ddots & \vdots \\ \sigma_{(J-1+p),1} & \cdots & \sigma_{(J-1+p,J-1)} \end{bmatrix} & \begin{bmatrix} \sigma_{J,J} & \cdots & \sigma_{J,(J-1+p)} \\ \vdots & \ddots & \vdots \\ \sigma_{(J-1+p),J} & \cdots & \sigma_{(J-1+p,J-1+p)} \end{bmatrix} \end{bmatrix}$$

then $\boldsymbol{\Sigma}_j$ will have the following form as it takes the corresponding elements in the *j*-th row and column of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\beta},\boldsymbol{\beta}}$:

$$\boldsymbol{\Sigma}_j = \begin{bmatrix} \sigma_{j,j} & \begin{bmatrix} \sigma_{j,J}, \cdots, \sigma_{j,(J-1+p)} \end{bmatrix} \\ \begin{bmatrix} \sigma_{J,j} \\ \vdots \\ \sigma_{(J-1+p),j} \end{bmatrix} & \boldsymbol{\Sigma}_{\boldsymbol{\beta},\boldsymbol{\beta}} \end{bmatrix}$$

For nominal multinomial model, the estimated standard error of $\hat{\eta}_{i,j}$ is given by

$$\hat{\sigma}_{\eta_{i,j}} = \sqrt{\boldsymbol{x}_i^T \boldsymbol{\Sigma}_j \boldsymbol{x}_i}, j = 1, \dots, J-1,$$

where $\boldsymbol{\Sigma}_j$ is part of covariance matrix $\boldsymbol{\Sigma}$ corresponding to the covariance matrix of $\widehat{\boldsymbol{\beta}}_j$.

5.1.3 Predicted values of the means

A predicted value, or fitted value, of the mean μ_i corresponding to $\boldsymbol{x}_i$ is given by

$$\hat{\mu}_i = g^{-1}\left(\boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}} + o_i\right)$$

where g^{-1} is the inverse of the link function. For binomial distribution with 0/1 binary response variable, $\hat{\mu}_i$ is the predicted probability of category 1.

For ordinal multinomial model, a predicted value, or fitted value, of the cumulative response probability for category j, $\gamma_{i,j}$ corresponding to $\boldsymbol{x}_i$ is given by

$$\hat{\gamma}_{i,j} = g^{-1}\left(\hat{\psi}_j - \boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}} + o_i\right), j = 1, \dots, J-1 \text{ with } \hat{\gamma}_{i,J} = 1.$$

For nominal multinomial model, the predicted value of the probability for category j corresponding $\boldsymbol{x}_i$ is given by

$$\hat{\pi}_{i,j} = g^{-1}\left(\hat{\eta}_{i,j}\right) = \begin{cases} \dfrac{\exp\left(\hat{\eta}_{i,j}\right)}{1 + \sum_{k=1}^{J-1} \exp\left(\hat{\eta}_{i,k}\right)}, & j = 1, \cdots, J-1, \\ \dfrac{1}{1 + \sum_{k=1}^{J-1} \exp\left(\hat{\eta}_{i,k}\right)}, & j = J. \end{cases}$$

5.1.4 Confidence intervals for the means

Approximate $100(1-\alpha)\%$ confidence intervals for the mean μ_i can be computed as follows

$$g^{-1}\left(\boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}} + o_i \pm z_{1-\alpha/2} \hat{\sigma}_\eta\right)$$

For ordinal multinomial model, approximate $100(1-\alpha)\%$ confidence intervals for the cumulative response probability $\hat{\gamma}_{i,j}$ can be computed as follows

$$g^{-1}\left(\hat{\psi}_j - \boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}} + o_i \pm z_{1-\alpha/2} \hat{\sigma}_{\eta_{i,j}}\right), j = 1, \cdots, J-1.$$

If either endpoint in the argument is outside the valid range for the inverse link function, the corresponding confidence interval endpoint is set to a system missing value.

For nominal multinomial model, approximate $100(1-\alpha)\%$ confidence intervals for the probability, $\hat{\pi}_{i,j}$ can be computed as follows

$$\hat{\pi}_{i,j} \pm z_{1-\alpha/2}\hat{\sigma}_{\pi_{i,j}}, j = 1, \cdots, J.$$

where $\hat{\sigma}_{\pi_{i,j}}$ can be computed by

$$\hat{\sigma}_{\pi_{i,j}} = \left[\frac{\partial\hat{\pi}_{i,j}}{\partial\hat{\eta}_{i,1}}, \cdots, \frac{\partial\hat{\pi}_{i,j}}{\partial\hat{\eta}_{i,J-1}}\right] \mathrm{Cov}(\hat{\boldsymbol{\eta}}_i) \left[\frac{\partial\hat{\pi}_{i,j}}{\partial\hat{\eta}_{i,1}}, \cdots, \frac{\partial\hat{\pi}_{i,j}}{\partial\hat{\eta}_{i,J-1}}\right]^{\mathrm{T}},$$

$$\frac{\partial\hat{\pi}_{i,j}}{\partial\hat{\eta}_{i,k}} = \begin{cases} \hat{\pi}_{i,j}\left(1 - \hat{\pi}_{i,j}\right) & j = k \\ -\hat{\pi}_{i,j}\hat{\pi}_{i,k} & j \neq k' \end{cases}$$

$$\mathrm{Cov}(\hat{\boldsymbol{\eta}}_i) = \mathrm{Cov}\left(\begin{bmatrix} \hat{\eta}_{i,1} \\ \vdots \\ \hat{\eta}_{i,J-1} \end{bmatrix}\right) = \left(\boldsymbol{I}_{J-1} \otimes \boldsymbol{x}_i^T\right)\boldsymbol{\Sigma}\left(\boldsymbol{I}_{J-1} \otimes \boldsymbol{x}_i\right)$$

and $\boldsymbol{I}_{J-1}$ is a $(J-1) \times (J-1)$ identity matrix and $\boldsymbol{\Sigma}$ could be $\boldsymbol{\Sigma}_m$ **or** $\boldsymbol{\Sigma}_r$**.**

5.1.5 Predicted category for binomial and multinomial distributions

For binomial distribution with 0/1 binary response variable, the predicted category $c(\boldsymbol{x}_i)$ is

$$c(\boldsymbol{x}_i) = \begin{cases} 1 \text{ (or success)} & \text{if } \mu_i \geq 0.5 \\ 0 \text{ (or failure)} & \text{otherwise} \end{cases}.$$

For ordinal and nominal multinomial model, the predicted category $c(\boldsymbol{x}_i)$ is the one with the highest predicted probability, i.e.,

$$c(\boldsymbol{x}_i) = \arg\max_j \hat{\pi}_{i,j}$$

If there is a tie in determining $c(\boldsymbol{x}_i)$, then tie will be broken by choosing the category with

1) Higher $N_j = \sum_{i=1}^{n} f_i y_{i,j}$.
2) If it ties in 1), choose the one with lower category number.

5.1.6 Classification table for binomial and multinomial distributions

Suppose that $c(j, j')$ is the sum of the frequency for the observations whose actual target category is j (as row) and predicted target category is j' (as column), $j, j' = 1, \cdots, J$ (note that $J = 2$ for binomial), then

$$c(j, j') = \sum_{i=1}^{n} f_i I(y_i = j, c(\boldsymbol{x}_i) = j')$$

where $I(\cdot)$ is indicator function.

Suppose that $p_{j,j'}$is the (j, j')th element of the classification table, which is row percentage, then

$$p_{j,j'} = \left(\frac{c(j, j')}{\sum_{k=1}^{J} c(j, k)} \right) \times 100\%$$

The percentage of total correct predictions of the model is

$$p_{total} = \left(\frac{\sum_{j=1}^{J} c(j, j)}{\sum_{j=1}^{J} \sum_{j'=1}^{J} c(j, j')} \right) \times 100\%.$$

5.2 Model diagnostics

In addition to predicted values, we can calculate some values which would be good for model diagnostics for all distributions except multinomial. They include leverage values, residuals and cook's distance values.

5.2.1 Leverage values

The leverage value h_i is defined as the i-th diagonal element of the hat matrix

$$\boldsymbol{H} = \boldsymbol{W}_e^{1/2} \boldsymbol{X} (\boldsymbol{X}^T \boldsymbol{W}_e \boldsymbol{X})^{-} \boldsymbol{X}^T \boldsymbol{W}_e^{1/2}$$

where the i-th diagonal element for $\boldsymbol{W}_e$ is

$$w_{e,i} = \frac{\omega_i}{\phi} \cdot \frac{1}{V(\mu_i)(g'(\mu_i))^2}.$$

5.2.2 Residuals

We will offer 5 different residuals:

(a) Raw residual

The raw residual is defined as

$$r_i^R = y_i - \hat{\mu}_i$$

where y_i is the i-th response and $\hat{\mu}_i$ is the corresponding predicted mean. Note for binomial response with a binary format, y values are 0 for the reference category and 1 for the category we are modeling.

(b) Pearson residual

The Pearson residual is the square root of the i-th contribution to the Pearson chi-square, with the sign of the raw residual.

$$r_i^P = \text{sign}(y_i - \hat{\mu}_i)\sqrt{\gamma_i} = (y_i - \hat{\mu}_i)\sqrt{\frac{\omega_i}{V(\hat{\mu}_i)}}.$$

(c) Deviance residual

The deviance residual is defined as the square root of the contribution of the i-th observation to the deviance, with the sign of the raw residual.

$$r_i^D = \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}.$$

where d_i is the contribution of the i-th case to the deviance, see Table 9, and $\text{sign}(y_i - \hat{\mu}_i)$ is 1 if $y_i - \hat{\mu}_i$ is positive and -1 if $y_i - \hat{\mu}_i$ is negative.

(d) Standardized (and studentized) Pearson residual

The standardized (and studentized) Pearson residual is that the Pearson residual is multiplied by the factor $(\phi(1 - h_i))^{-1/2}$

$$r_i^{SP} = (y_i - \hat{\mu}_i)\sqrt{\frac{\omega_i}{\phi V(\hat{\mu}_i)(1 - h_i)}} = r_i^P\sqrt{\frac{1}{\phi(1 - h_i)}}.$$

(e) Standardized (and studentized) deviance residual

The standardized (and studentized) deviance residual is that the deviance residual is multiplied by the factor $(\phi(1-h_i))^{-1/2}$

$$r_i^{SD} = \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}\sqrt{\frac{1}{\phi(1-h_i)}} = r_i^D\sqrt{\frac{1}{\phi(1-h_i)}}.$$

(f) Likelihood residual

The likelihood residuals are defined by

$$r_i^L = \text{sign}(y_i - \hat{\mu}_i)\sqrt{h_i(r_i^{SP})^2 + (1-h_i)(r_i^{SD})^2}.$$

5.2.3 Cook's distance

Cook's distance measures the change to the solution that results from omitting each observation. The formula is

$$C_i = \frac{1}{p_x} \cdot \frac{h_i}{1-h_i}(r_i^{SP})^2.$$

Note on calculating scoring for binomial response with events/trials format

When r/m *format* for the binomial distribution is used, the response we used is the binomial proportion $y = r/m$, but to many people, the response for binomial distribution should be the number of events (r). Thus for r/m binomial distribution, the predicted value of the mean we are going to list is the expected number of trials, not the expected proportion. Then some of the above formulae in Section 5 should be modified. We will list the modified ones below and those unmodified ones are still the same as before.

Some notations for events/trials format we used before calculating scoring:

r_i	# of events
m_i	# of trials
y_i	proportion ($y_i = r_i/m_i$)

μ_i	expected proportion obtained from parameter estimation

- A predicted value of the mean: $\hat{\mu}_i = g^{-1}\left(\boldsymbol{x}_i^T\widehat{\boldsymbol{\beta}} + o_i\right) \times m_i$.

- Approximate 100(1–α)% confidence interval for the mean: $g^{-1}\left(\boldsymbol{x}_i^T\widehat{\boldsymbol{\beta}} + o_i \pm z_{1-\alpha/2}\hat{\sigma}_\eta\right) \times m_i$.

- The raw residual: $r_i^R = y_i m_i - \hat{\mu}_i$.

- The Pearson residual: $r_i^P = (y_i m_i - \hat{\mu}_i)\sqrt{\frac{\omega_i}{U_i}}$, where $U_i = \hat{\mu}_i\left(1 - (\hat{\mu}_i/m_i)\right)$ (base on # of events)

$$r_i^P = (y_i - \mu_i)\sqrt{\frac{\omega_i}{V(\mu)}}, \text{ where } V(\mu_i) = \frac{\mu_i(1-\mu_i)}{m_i} \quad \text{(based on proportion)}$$

- The deviance residual: $r_i^D = \text{sign}(y_i m_i - \hat{\mu}_i)\sqrt{d_i}$, where d_i is from Table 9 (based on # of events)

$$r_i^D = \text{sign}(y_i - \mu_i)\sqrt{d_i} \qquad \text{(based on proportion)}$$

Note:

- Unlike other distributions which μ_i and $\hat{\mu}_i$ are interchangeable, we need to distinguish μ_i and $\hat{\mu}_i$ for binomial distribution with events/trials format:

  μ_i: the expected proportion used before calculating scoring;

  $\hat{\mu}_i$: the expected number of events for calculating scoring (the predicted value of the mean).

  However, the Pearson residual and deviance residual are the same no matter they are based on # of events or proportion.

Appendix A - Ordinal Multinomial Distribution

For multinomial distribution, the GENLIN procedure supports only the ordinal multinomial model (or threshold model). The model form is not the same as the above traditional generalized linear model and would be consistent with other SPSS procedures, such as PLUM and CSORDINAL. The target variable ***y*** is assumed to be ordinal, its values have an intrinsic linear ordering and correspond to consecutive integers from 1 to *J*. The design matrix ***X*** includes model predictors, but not an intercept. We need some new notations to define the model form:

J	The number of values for the ordinal target variable, $J \geq 2$.
y_i	Ordinal target variable for the record *i*. Its category values are denoted consecutive integers from 1 to *J*.
$y_{i,j}$	Indicator variable of record *i* for category j , i.e. $y_{i,j} = \begin{cases} 1 & \text{if } y_i = j \\ 0 & \text{otherwise} \end{cases}$.
$\boldsymbol{X}$	Design matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^{\mathrm{T}}$, where $\boldsymbol{x}_i^{\mathrm{T}} = (x_{i1}, \ldots, x_{ip})$, is for record i, the superscript T means transpose of a matrix or vector. Note that ***X*** includes model predictors, but not an intercept.
$\boldsymbol{\psi}$	*J* – 1 × 1 vector of threshold parameters , $\boldsymbol{\psi} = (\psi_1, \psi_2, \ldots, \psi_{J-1})^{\mathrm{T}}$ and $\psi_1 < \psi_2 < \cdots < \psi_{J-1}$.
$\boldsymbol{\beta}$	*p* × 1 vector of regression parameters associated with model predictors, $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^{\mathrm{T}}$.
$\boldsymbol{B}$	(*J* – 1 + *p*) × 1 vector of all parameters, $\boldsymbol{B} = (\boldsymbol{\psi}^{\mathrm{T}}, \boldsymbol{\beta}^{\mathrm{T}})^{\mathrm{T}}$.
$\gamma_{i,j}$	Conditional cumulative target probability for category *j* given observed independent variable vector $\boldsymbol{x_i}$, i.e., $\gamma_{i,j} = P(y_i \leq j \mid \boldsymbol{x_i})$.
$\pi_{i,j}$	Conditional target probability for category *j* given observed independent variable vector $\boldsymbol{x_i}$, i.e., $\pi_{i,j} = P(y_i = j \mid \boldsymbol{x_i})$ and $\pi_{i,j} = \gamma_{i,j} - \gamma_{i,j-1}$ for $j = 1, \ldots, J$.
$\eta_{i,j}$	Linear predictor value of record *i* for category *j*. It is related to $\gamma_{i,j}$ through a cumulative link function.

The form for ordinal target y is

$$\eta_{i,j} = g(\gamma_{i,j}) = \psi_j - \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{\beta} + o_i, \quad y_i \sim F.$$

Note:

- To check the dependencies here in the design matrix, columns of $(1,-\boldsymbol{X})^{\mathrm{T}}\boldsymbol{\Psi}(1,-\boldsymbol{X})$, where $\boldsymbol{\Psi} = \mathrm{diag}(f_1\omega_1,\dots f_n\omega_n)$, are examined by using the sweep operator.

Log likelihood function

Given a record $\boldsymbol{x}_i$, y_i follows a multinomial distribution. The kernel log likelihood function is

$$\ell_k = \sum_{i=1}^{n}\frac{f_i\omega_i}{\phi}\sum_{j=1}^{J} y_{i,j}\ln\left(\pi_{i,j}\right),\ \text{where}\ y_{i,j} = \begin{cases}1 & \text{if } y_i = j\\ 0 & \text{otherwise}\end{cases},$$

and the full log likelihood function $\ell = \ell_k + c$, where c is computed based on subpopulations (see Section 4.3.3.2 for details.)

Table A.1: Cumulative Link Function Name, Form, Inverse Form and Range of the Predicted Cumulative Probability

Link function name	$\eta = g(\gamma)$	Inverse $\gamma = g^{-1}(\eta)$	Range of $\hat{\gamma}$
Cumulative logit	$\ln\left(\frac{\gamma}{1-\gamma}\right)$	$\frac{\exp(\eta)}{1+\exp(\eta)}$	$\hat{\gamma}\in(0,1)$
Cumulative probit	$\Phi^{-1}(\gamma)$, where $\Phi(\xi) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\xi} e^{-z^2/2}dz$	$\Phi(\eta)$	$\hat{\gamma}\in(0,1)$
Cumulative complementary log-log	$\ln(-\ln(1-\gamma))$	$1-\exp(-\exp(\eta))$	$\hat{\gamma}\in(0,1)$
Cumulative negative log-log	$-\ln(-\ln(\gamma))$	$\exp(-\exp(-\eta))$	$\hat{\gamma}\in(0,1)$
Cumulative Cauchit	$\tan(\pi(\gamma-0.5))^*$	$0.5+\arctan(\eta)/\pi^*$	$\hat{\gamma}\in(0,1)$

* π in the formula is denoted pi, not the target probability.

Table A.2: The Inverse First and Second Derivatives of Cumulative Link Function

Link function name	Inverse first derivative $\frac{\partial \gamma}{\partial \eta} = \Delta$	Inverse second derivative $\frac{\partial^2 \gamma}{\partial \eta^2}$
Cumulative logit	$\gamma(1-\gamma)$	$\Delta(1-2\gamma)$
Cumulative probit	$\phi\left(\Phi^{-1}(\gamma)\right)$, where $\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$	$-\Delta \times \Phi^{-1}(\gamma)$
Cumulative complementary log-log	$(\gamma-1)\ln(1-\gamma)$	$\Delta\left(1+\ln(1-\gamma)\right)$
Cumulative negative log-log	$-\gamma \ln(\gamma)$	$-\Delta\left(1+\ln(\gamma)\right)$
Cumulative Cauchit	$\cos^2\left(\pi(\gamma-0.5)\right)/\pi^{*}$	$\Delta \times \sin(2\pi\gamma)^{*}$

* π in the formula is denoted pi, not the target probability.

First derivatives

$$\mathbf{s} = \left[\frac{\partial \ell}{\partial \boldsymbol{B}}\right]_{(J-1+p)\times 1} = \begin{bmatrix} \frac{\partial \ell}{\partial \boldsymbol{\psi}} \\ \frac{\partial \ell}{\partial \boldsymbol{\beta}} \end{bmatrix} = \mathbf{0}.$$

$$\mathbf{s} = \left[\frac{\partial \ell}{\partial \psi_1}, \cdots, \frac{\partial \ell}{\partial \psi_{J-1}}, \frac{\partial \ell}{\partial \beta_1}, \cdots, \frac{\partial \ell}{\partial \beta_p}\right]^{\mathrm{T}}, \text{ i.e.,}$$

$$\frac{\partial \ell}{\partial \psi_j} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} \left(\frac{y_{i,j}}{\pi_{i,j}} - \frac{y_{i,j+1}}{\pi_{i,j+1}}\right), \quad j = 1, \ldots, J-1 \text{ and}$$

$$\frac{\partial \ell}{\partial \beta_t} = -\sum_{i=1}^{n} \sum_{j=1}^{J} \frac{f_i \omega_i}{\phi} \left(\frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} - \frac{\partial \gamma_{i,j-1}}{\partial \eta_{i,j-1}}\right) \frac{y_{i,j}}{\pi_{i,j}} x_{it}, \quad t = 1, \ldots, p,$$

where $\pi_{i,j} = \gamma_{i,j} - \gamma_{i,j-1}$ for $j = 1,\ldots,J$ and $\gamma_{i,j} = \begin{cases} 0 & j = 0 \\ g^{-1}\left(\psi_j - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta} + o_i\right) & j = 1,\ldots,J-1, \\ 1 & j = J \end{cases}$ which is from Table A.1 and $\dfrac{\partial\gamma_{i,j}}{\partial\eta_{i,j}}$ is defined in Table A.2 for $j = 1,\ldots,J-1$ and by the definition $\dfrac{\partial\gamma_{i,0}}{\partial\eta_{i,0}} = \dfrac{\partial\gamma_{i,J}}{\partial\eta_{i,J}} = 0$.

Note if $\partial\gamma_{i,j} = 0$ or $\partial\gamma_{i,j} = 1$ then $\dfrac{\partial\gamma_{i,j}}{\partial\eta_{i,j}} = 0$ for all cumulative link functions.

Second derivatives

$$\mathbf{H} = \left[\frac{\partial^2\ell}{\partial\boldsymbol{B}\,\partial\boldsymbol{B}^{\mathrm{T}}}\right]_{(J-1+p)\times(J-1+p)} = \begin{bmatrix} \dfrac{\partial^2\ell}{\partial\boldsymbol{\psi}\,\partial\boldsymbol{\psi}^{\mathrm{T}}} & \dfrac{\partial^2\ell}{\partial\boldsymbol{\psi}\,\partial\boldsymbol{\beta}^{\mathrm{T}}} \\ \dfrac{\partial^2\ell}{\partial\boldsymbol{\beta}\,\partial\boldsymbol{\psi}^{\mathrm{T}}} & \dfrac{\partial^2\ell}{\partial\boldsymbol{\beta}\,\partial\boldsymbol{\beta}^{\mathrm{T}}} \end{bmatrix}.$$

The elements of **H** have two forms: (1) the expected first derivatives of the estimating equation **s** which is applied to Fisher scoring and (2) the first derivatives of the estimating equation **s** which is applied to Newton Raphson.

(1) Expected second derivatives have the following expressions:

$$\frac{\partial^2\ell}{\partial\psi_{j-1}\partial\psi_j} = \sum_{i=1}^{n}\frac{f_i\omega_i}{\phi}\frac{\partial\gamma_{i,j-1}}{\partial\eta_{i,j-1}}\frac{\partial\gamma_{i,j}}{\partial\eta_{i,j}}\frac{1}{\pi_{i,j}}, \quad j = 2,\ldots,J-1,$$

$$\frac{\partial^2\ell}{\partial\psi_j^2} = -\sum_{i=1}^{n}\frac{f_i\omega_i}{\phi}\left(\frac{\partial\gamma_{i,j}}{\partial\eta_{i,j}}\right)^2\left(\frac{1}{\pi_{i,j}} + \frac{1}{\pi_{i,j+1}}\right), \quad j = 1,\ldots,J-1,$$

$$\frac{\partial^2\ell}{\partial\psi_l\partial\psi_j} = 0, \quad \text{for } |l\text{ - }j| > 1,$$

$$\frac{\partial^2\ell}{\partial\psi_j\partial\beta_t} = \sum_{i=1}^{n}\frac{f_i\omega_i}{\phi}\left[\left(\frac{\partial\gamma_{i,j}}{\partial\eta_{i,j}} - \frac{\partial\gamma_{i,j-1}}{\partial\eta_{i,j-1}}\right)\frac{1}{\pi_{i,j}} - \left(\frac{\partial\gamma_{i,j+1}}{\partial\eta_{i,j+1}} - \frac{\partial\gamma_{i,j}}{\partial\eta_{i,j}}\right)\frac{1}{\pi_{i,j+1}}\right]\frac{\partial\gamma_{i,j}}{\partial\eta_{i,j}}x_{it},$$
$$j = 1,\ldots,J-1, \quad t = 1,\ldots,p,$$

$$\frac{\partial^2 \ell}{\partial \beta_t \partial \beta_u} = -\sum_{i=1}^{n}\sum_{j=1}^{J} \frac{f_i \omega_i}{\phi} \left(\frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} - \frac{\partial \gamma_{i,j-1}}{\partial \eta_{i,j-1}} \right)^2 \frac{1}{\pi_{i,j}} x_{it} x_{iu}, \quad t, u = 1, \ldots, p.$$

(2) Second derivatives have the following expressions:

$$\frac{\partial^2 \ell}{\partial \psi_{j-1} \partial \psi_j} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \frac{\partial \gamma_{i,j-1}}{\partial \eta_{i,j-1}} \frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} \frac{y_{i,j}}{\pi_{i,j}^2}, \quad j = 2, \ldots, J-1,$$

$$\frac{\partial^2 \ell}{\partial \psi_j^2} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \left[\frac{\partial^2 \gamma_{i,j}}{\partial \eta_{i,j}^2} \left(\frac{y_{i,j}}{\pi_{i,j}} - \frac{y_{i,j+1}}{\pi_{i,j+1}} \right) - \left(\frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} \right)^2 \left(\frac{y_{i,j}}{\pi_{i,j}^2} + \frac{y_{i,j+1}}{\pi_{i,j+1}^2} \right) \right], \quad j = 1, \ldots, J-1,$$

$$\frac{\partial^2 \ell}{\partial \psi_l \partial \psi_j} = 0, \quad \text{for } |l \text{ - } j| > 1,$$

$$\begin{aligned} \frac{\partial \ell}{\partial \psi_j \partial \beta_t} = &-\sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \left[\frac{\partial^2 \gamma_{i,j}}{\partial \eta_{i,j}^2} \pi_{i,j} - \frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} \left(\frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} - \frac{\partial \gamma_{i,j-1}}{\partial \eta_{i,j-1}} \right) \right] \frac{y_{i,j}}{\pi_{i,j}^2} x_{it} + \\ &\sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \left[\frac{\partial^2 \gamma_{i,j}}{\partial \eta_{i,j}^2} \pi_{i,j+1} - \frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} \left(\frac{\partial \gamma_{i,j+1}}{\partial \eta_{i,j+1}} - \frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} \right) \right] \frac{y_{i,j+1}}{\pi_{i,j+1}^2} x_{it}, \\ &j = 1, \ldots, J-1, \quad t = 1, \ldots, p, \end{aligned}$$

$$\begin{aligned} \frac{\partial \ell}{\partial \beta_t \partial \beta_u} = &\sum_{i=1}^{n}\sum_{j=1}^{J} \frac{f_i \omega_i}{\phi} \left[\left(\frac{\partial^2 \gamma_{i,j}}{\partial \eta_{i,j}^2} - \frac{\partial^2 \gamma_{i,j-1}}{\partial \eta_{i,j-1}^2} \right) \pi_{i,j} - \left(\frac{\partial \gamma_{i,j}}{\partial \eta_{i,j}} - \frac{\partial \gamma_{i,j-1}}{\partial \eta_{i,j-1}} \right)^2 \right] \frac{y_{i,j}}{\pi_{i,j}^2} x_{it} x_{iu}, \\ &t, u = 1, \ldots, p, \end{aligned}$$

where $\frac{\partial^2 \gamma_{i,j}}{\partial \eta_{i,j}^2}$ is defined in Table 5.2 for $j = 1, \ldots, J-1$ and by the definition $\frac{\partial^2 \gamma_{i,0}}{\partial \eta_{i,0}^2} = \frac{\partial^2 \gamma_{i,J}}{\partial \eta_{i,J}^2} = 0$.

Initial values

Let $N_j = \sum_{i=1}^{n} f_i y_{i,j}$ be the number of responses in category *j*, for $j = 1, \dots, J$, and $N = \sum_{i=1}^{n} f_i$ be the effective sample size. Initial values for threshold parameters without and with offset variable, o_i, are then computed according to the following formulae:

$$\psi_j^{(0)} = g\left(\frac{\sum_{l=1}^{j} N_l}{N}\right) \text{ and } \psi_j^{(0)} = g\left(\frac{\sum_{l=1}^{j} N_l}{N}\right) - \bar{o}_j \text{ for } j = 1, \dots, J-1, \text{ respectively;}$$

where $\bar{o}_j = \sum_{i=1}^{n}\sum_{l=1}^{j} f_i y_{i,l} o_i \Big/ \sum_{i=1}^{n}\sum_{l=1}^{j} f_i y_{i,l}$. Initial values for all regression parameters are set to zero, i.e. $\beta_t^{(0)} = 0$, for $t = 1, \dots, p$.

Notes:

- Similarly, the computation of ℓ_k, ℓ, **s**, **H** as well as N_j, N, $\bar{o}_j$ in initial values can be implemented in map-reduce environment.

Appendix B - Nominal Multinomial Distribution

Like ordinal multinomial distribution, the form of nominal multinomial model is not same as the other traditional generalized linear model. So we need to introduce some new notations.

y_i	Nominal categorical target variable for the record i. Its category values are denoted as 1, 2, etc.
J	The total number of categories for target variable.
$y_{i,j}$	Indicator variable for category j, i.e. $y_{i,j} = 1$ if $y_i = j$, otherwise $y_{i,j} = 0$.
$\boldsymbol{X}$	Design matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^{\mathrm{T}}$. The ith row is $\boldsymbol{x}_i^{\mathrm{T}} = (x_{i1}, \dots, x_{ip})$, where superscript T means transpose of a matrix or vector, $i = 1, \dots, n$ with $x_{i1} = 1$ if model has an intercept.
$\pi_{i,j}$	The target probability for category *j* given observed independent variable vector $\boldsymbol{x}_i$, i.e. $\pi_{i,j} = \Pr(y_i = j)$.

$\eta_{i,j}$	Linear predictor value of record i for category j .
$\boldsymbol{\beta}_j$	p × 1 vector of unknown parameters for the category j , $j = 1,\cdots, J-1$. The first element in $\boldsymbol{\beta}_j$ is the intercept for the category j , if there is one.
$\boldsymbol{\beta}$	$\boldsymbol{\beta} = (\boldsymbol{\beta}_1^{\mathrm{T}},\cdots,\boldsymbol{\beta}_{J-1}^{\mathrm{T}})^{\mathrm{T}}$

The form of a generalized linear model for nominal target y is

$$\eta_{i,j} = g(\pi_{i,j}) = \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_j + o_i,\ y_i \sim F$$

where $\eta_{i,j}$ is linear predictor value of record i for category j ; o_i is the offset variable value of the record i and $g(\cdot)$ is logit link function such that :

$$g(\pi_{i,j}) = \log\left(\frac{\pi_{i,j}}{\pi_{i,J}}\right), \quad j = 1,\cdots, J-1$$

Or

$$\pi_{i,j} = g^{-1}(\eta_{i,j}) = \begin{cases} \dfrac{\exp\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_j + o_i\right)}{1+\sum_{k=1}^{J-1}\exp\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_k + o_i\right)}, & j = 1,\cdots, J-1, \\ \dfrac{1}{1+\sum_{k=1}^{J-1}\exp\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}_k + o_i\right)}, & j = J \end{cases}$$

where $\boldsymbol{\beta}_j = (\beta_{j1},\cdots,\beta_{jp})^{\mathrm{T}}$ is the regression parameter vector for target category j . There are $p(J-1)$ regression parameters in total $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^{\mathrm{T}},\cdots,\boldsymbol{\beta}_{J-1}^{\mathrm{T}})^{\mathrm{T}}$.

Log likelihood function

Given a record $\boldsymbol{x}_i$, y_i follows a multinomial distribution. The log likelihood function for probability distribution is

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \sum_{j=1}^{J} y_{i,j} \ln(\pi_{i,j}) + c,$$

where c is computed based on subpopulations (see Section 4.3.3.2 for details.)

First derivatives

The first derivative for $\boldsymbol{\beta}_j$ is

$$s(\boldsymbol{\beta}_j) = \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_j} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} (y_{i,j} - \pi_{i,j}) \boldsymbol{x}_i, \ \ j = 1, \cdots, J-1$$

So the first derivative for $\boldsymbol{\beta}$ is

$$\boldsymbol{s} = (s(\boldsymbol{\beta}_1)^{\mathrm{T}}, \cdots, s(\boldsymbol{\beta}_{J-1})^{\mathrm{T}})^{\mathrm{T}} = \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} (\mathbf{y}_i - \boldsymbol{\pi}_i) \otimes \boldsymbol{x}_i$$

where $\mathbf{y}_i = (y_{i,1}, \cdots, y_{i,J-1})^T$ and $\boldsymbol{\pi}_i = [\pi_{i,1}, \cdots, \pi_{i,J-1}]^T$. $\otimes$ is the Kronecker product such that $A \otimes B$ produce a matrix with A 's element a_{ij} being replaced by a matrix $a_{ij} B$.

Second derivatives

The second derivative (Hessian) matrix, $\mathbf{H}$, is a $p(J-1) \times p(J-1)$ matrix with the form:

$$\mathbf{H} = \left[\frac{\partial^2 \ell}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^{\mathrm{T}}} \right]_{p(J-1) \times p(J-1)} = \begin{bmatrix} \frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_1 \partial \boldsymbol{\beta}_1^{\mathrm{T}}} & \cdots & \frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_1 \partial \boldsymbol{\beta}_{J\text{-}1}^{\mathrm{T}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_{J\text{-}1} \partial \boldsymbol{\beta}_1^{\mathrm{T}}} & \cdots & \frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_{J\text{-}1} \partial \boldsymbol{\beta}_{J\text{-}1}^{\mathrm{T}}} \end{bmatrix}$$

where the $(k, j)^{\text{th}}$ block element of $\mathbf{H}$, for $k, j = 1, \cdots, J-1$ is

$$\frac{\partial^2 \ell}{\partial \boldsymbol{\beta}_k \partial \boldsymbol{\beta}_j^{\mathrm{T}}} = \begin{cases} \sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \pi_{i,k} \pi_{i,j} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} & k \neq j \\ -\sum_{i=1}^{n} \frac{f_i \omega_i}{\phi} \pi_{i,j} (1-\pi_{i,j}) \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} & k = j \end{cases}$$

Initial values

For all non-intercept regression parameters, set their initial values to be zero. For intercepts, if there are any, set for $j = 1, \cdots, J - 1$,

$$\beta_{j1}^{(0)} = \log\left(\frac{N_j}{N_J}\right) \qquad \text{if there is no offset variable, and}$$

$$\beta_{j1}^{(0)} = \log\left(\frac{N_j}{N_J}\right) - \bar{o}_j \qquad \text{if there is an offset variable,}$$

where $N_j = \sum_{i=1}^{n} f_i y_{i,j}$, and $\bar{o}_j = \frac{\sum_{i=1}^{n} f_i y_{i,j} o_i}{\sum_{i=1}^{n} f_i y_{i,j}}$.

<u>Notes</u>:

- Similarly, the computation of $\ell_k, \boldsymbol{\ell}, \mathbf{s}, \mathbf{H}$ as well as N_j, $\bar{o}_j$ in initial values can be implemented in map-reduce environment.

Appendix C - Tweedie Distribution

- V_i in Tweedie distribution is an infinite series as follows:

$$V_i = \sum_{j=1}^{\infty} V_{ij} \text{ and } V_{ij} = \frac{\omega_i^{j(1-\alpha)} y_i^{-j\alpha} (q-1)^{j\alpha}}{\phi^{j(1-\alpha)} (2-q)^j \Gamma(-j\alpha) j!},$$

where $\alpha = \frac{2-q}{1-q}$, and note α is negative for 1 < q < 2. To evaluate the infinite summation for V_i, the value of j is determined for which V_{ij} reaches a maximum (we evaluate $\max_j \ln(V_{ij})$ here) and sum the necessary terms of the series in that region. The method proposed by Dunn and Smyth (2005) is adopted here and summarized as follows:

(1) Approximate the gamma functions in $\ln\left(V_{ij}\right)$ by using Stirling's approximation as

$$\ln\left(j!\right)=\ln\Gamma\left(1+j\right)\approx\left(j+\frac{1}{2}\right)\ln\left(j\right)-j+\frac{1}{2}\ln\left(2\pi\right),$$

$$\ln\Gamma\left(-j\alpha\right)\approx\ln\Gamma\left(1-j\alpha\right)\approx\left(-j\alpha+\frac{1}{2}\right)\ln\left(-j\alpha\right)-\left(-j\alpha\right)+\frac{1}{2}\ln\left(2\pi\right),$$

and $\ln\left(V_{ij}\right)$ becomes

$$\begin{aligned}\ln\left(V_{ij}\right)&=j\ln\left(z_i\right)-\ln\left(j!\right)-\ln\Gamma\left(-j\alpha\right)\\&\approx j\left[\ln\left(z_i\right)+\left(1-\alpha\right)\left(1-\ln\left(j\right)\right)+\alpha\ln\left(-\alpha\right)\right]-\ln\left(j\right)-\frac{1}{2}\ln\left(-\alpha\right)-\ln\left(2\pi\right),\end{aligned}$$

where $z_i=\dfrac{\omega_i^{1-\alpha}y_i^{-\alpha}\left(q-1\right)^{\alpha}}{\phi^{1-\alpha}\left(2-q\right)}$.

(2) Treat *j* as continuous and $\ln\left(V_{ij}\right)$ is differentiated with respect to *j*

$$\begin{aligned}\frac{\partial\ln\left(V_{ij}\right)}{\partial j}&\approx\ln\left(z_i\right)+\alpha\ln\left(-\alpha\right)-\left(1-\alpha\right)\ln\left(j\right)-\frac{1}{j}\\&\approx\ln\left(z_i\right)+\alpha\ln\left(-\alpha\right)-\left(1-\alpha\right)\ln\left(j\right),\end{aligned}$$

since the term 1/*j* is ignored for *j* large.

(3) Set the above derivative to zero to obtain the value of *j* at which $\ln\left(V_{ij}\right)$ reaches a maximum

$$j_{\max}=\frac{\omega_i y_i^{2-q}}{\phi\left(2-q\right)}.$$

If y_i is large, ω_i large, ϕ small or q near 2, $j_{\max}$ would be large. The approximate maximum value of $\ln\left(V_{ij}\right)$ is

$$\ln\left(V_{i,j_{\max}}\right) = j_{\max}\left(1-\alpha\right) - \ln\left(j_{\max}\right) - \frac{1}{2}\ln\left(-\alpha\right) - \ln\left(2\pi\right).$$

(4) Find the lower and upper bounds of j to approximate V_i with $\tilde{V}_i = \sum_{j=j_L}^{j_U} V_{ij}$. We simply search $1 \le j_L < j_{\max}$ and $j_U > j_{\max}$ such that $\ln\left(V_{i,j_L}\right) < \ln\left(V_{i,j_{\max}}\right) - 37$ and $\ln\left(V_{i,j_U}\right) < \ln\left(V_{i,j_{\max}}\right) - 37$, respectively.

(5) Compute $\ln\left(\tilde{V}_i\right)$ in the following way to avoid the possibility of floating point overflow:

$$\ln\left(\tilde{V}_i\right) = \ln\left(V_{i,j_{\max}}\right) + \ln\sum_{j=j_L}^{j_U}\exp\left(\ln\left(V_{ij}\right) - \ln\left(V_{i,j_{\max}}\right)\right).$$

- The value of j at which the series $\sum_{j=1}^{\infty} jV_{ij}$ and $\sum_{j=1}^{\infty} j^2 V_{ij}$ reach their maximums can be still approximated by $j_{\max} = \frac{\omega_i y_i^{2-q}}{\phi\left(2-q\right)}$. Then

$$\ln\left(j_{\max}V_{i,j_{\max}}\right) = j_{\max}\left(1-\alpha\right) - \frac{1}{2}\ln\left(-\alpha\right) - \ln\left(2\pi\right) \text{ and}$$

$$\ln\left(j_{\max}^2 V_{i,j_{\max}}\right) = j_{\max}\left(1-\alpha\right) + \ln\left(j_{\max}\right) - \frac{1}{2}\ln\left(-\alpha\right) - \ln\left(2\pi\right).$$

Note that there are n $j_{\max}$ values corresponding to n complete records. Thus, j_L and j_U should be different for each record. In addition, j_L and j_U should be different for $\sum_{j=1}^{\infty} V_{ij}$, $\sum_{j=1}^{\infty} jV_{ij}$ and $\sum_{j=1}^{\infty} j^2 V_{ij}$ as well. However, Dunn and Smyth (2005) have found it useful to choose common j_L and j_U for all records and all summations. Basic idea is that searching j_L and j_U based on $\sum_{j=1}^{\infty} jV_{ij}$ for each record then the minimum of j_L and maximum of j_U from all records would be the common j_L and j_U, respectively.

1. Search j_L and j_U:

```
For $(i = 1, n)\{$

    $\ln(z_i) = -\alpha\ln(y_i) + (1-\alpha)\ln(\omega_i) + \alpha\ln(q-1) - \ln(2-q) - (1-\alpha)\tau$

    $j_i = \omega_i * y_i^{2-q} / [\exp(\tau) * (2-q)]$

}

$\ln(z_{\max}^{+}) = \max_{i=1,\ldots,n} \ln(z_i)$;   $\ln(z_{\max}^{-}) = \min_{i=1,\ldots,n} \ln(z_i)$

$cpart^{+} = \ln(z_{\max}^{+}) + (1-\alpha) + \alpha * \ln(-\alpha)$;  $cpart^{-} = \ln(z_{\max}^{-}) + (1-\alpha) + \alpha * \ln(-\alpha)$

$j_{\max}^{+} = \max_{i=1,\ldots,n}\{j_i\}$;   $j_{\max}^{-} = \min_{i=1,\ldots,n}\{j_i\}$

$\ln V_{\max}^{+} = j_{\max}^{+} * (1-\alpha)$;   $\ln V_{\max}^{-} = j_{\max}^{-} * (1-\alpha)$

$\text{limit} = 37$

$j = \max(1, j_{\max}^{+})$;   $est \ln V = \ln V_{\max}^{+}$

While $(est \ln V > (\ln V_{\max}^{+} - \text{limit}))\{$

    $j = j + 1$  (or  $j = j + 2$ as Dunn did which might speed the result)

    $est \ln V = j * (cpart^{+} - (1-\alpha) * \ln(j))$

}

$j_U = j$

$j = \max(1, j_{\max}^{-})$;   $est \ln V = \ln V_{\max}^{-}$
```

```
While $\left(est\ln V > \left(\ln V^{-}_{\max} - \text{limit}\right)\right)\{$

    $j = \max\left(1, j-1\right)$   (or   $j = \max\left(1, j-2\right)$ as Dunn did which might speed the result)

    $est\ln V = j*\left(cpart^{-} - \left(1-\alpha\right)*\ln\left(j\right)\right)$

}

$j_L = j$
```

2. Compute $\ln\left(\tilde{V}_i\right) = \ln\sum_{j=j_L}^{j_U} V_{ij}$, $\tilde{V}_i$, $j\tilde{V}_i = \sum_{j=j_L}^{j_U} jV_{ij}$ and $j^2\tilde{V}_i = \sum_{j=j_L}^{j_U} j^2V_{ij}$ in the following way to avoid the possibility of floating point overflow:

```
For $\left(j = j_L, j_U\right)\{$

    $\ln\left(V_{ij}\right) = j\ln\left(z_i\right) - \ln\Gamma\left(1+j\right) - \ln\Gamma\left(-j\alpha\right)$

    $\ln\left(jV_{ij}\right) = \ln\left(V_{ij}\right) + \ln\left(j\right)$

    $\ln\left(j^2V_{ij}\right) = \ln\left(V_{ij}\right) + 2*\ln\left(j\right)$

}

$\ln\left(V_{i,j_{\max}}\right) = \max_{j=j_L,\ldots,j_U}\ln\left(V_{ij}\right)$; $\ln\left(jV_{ij}\right)_{\max} = \max_{j=j_L,\ldots,j_U}\ln\left(jV_{ij}\right)$; $\ln\left(j^2V_{ij}\right)_{\max} = \max_{j=j_L,\ldots,j_U}\ln\left(j^2V_{ij}\right)$

$sum1_i = 0;\ sum2_i = 0;\ sum3_i = 0$

For $\left(j = j_L, j_U\right)\{$

    $v_{ij} = \exp\left(\ln\left(V_{ij}\right) - \ln\left(V_{i,j_{\max}}\right)\right)$; $sum1_i = sum1_i + v_{ij}$
```

$$jv_{ij} = \exp\left(\ln\left(jV_{ij}\right) - \ln\left(jV_{ij}\right)_{\max}\right); \quad sum2_i = sum2_i + jv_{ij}$$

$$j^2v_{ij} = \exp\left(\ln\left(j^2V_{ij}\right) - \ln\left(j^2V_{ij}\right)_{\max}\right); \quad sum3_i = sum3_i + j^2v_{ij}$$

}

$$\ln\left(\tilde{V}_i\right) = \ln\left(V_{i,j_{\max}}\right) + \ln\left(sum1_i\right)$$

$$\frac{j\tilde{V}_i}{\tilde{V}_i} = \exp\left(\ln\left(jV_{ij}\right)_{\max} - \ln\left(V_{i,j_{\max}}\right)\right) * \frac{sum2_i}{sum1_i}$$

$$\frac{j^2\tilde{V}_i}{\tilde{V}_i} = \exp\left(\ln\left(j^2V_{ij}\right)_{\max} - \ln\left(V_{i,j_{\max}}\right)\right) * \frac{sum3_i}{sum1_i}$$

$$\frac{\partial V_i / \partial \tau}{V_i} = \left(\alpha - 1\right) * \frac{j\tilde{V}_i}{\tilde{V}_i}$$

$$\frac{\partial^2 V_i / \partial \tau^2}{V_i} = \left(\alpha - 1\right)^2 * \frac{j^2\tilde{V}_i}{\tilde{V}_i}$$

Appendix D - Digamma and Trigamma Function

This part is based on Zhong (2006b).

This document descries the computational algorithm of the digamma and Trigamma function based on the formulas in Abramowitz and Stegun (1972).

z	A complex number
x	A real number
$\Gamma(z)$	The gamma function
$\psi(z)$	Digamma function

$\psi'(z)$	Trigamma function
B_n	The Bernoulli number

The gamma function, $\Gamma(z)$, is defined by the following integral,

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt, \quad real(z) > 0$$

$\ln(\Gamma(z))$ is a log-gamma function evaluated at *z*.

$\psi(z)$ is digamma function, which is the derivative of logarithm of a gamma function evaluated at *z*,

$$\psi(z) = \frac{\partial \ln(\Gamma(z))}{\partial z} = \frac{\Gamma'(z)}{\Gamma(z)}$$

$\psi'(z)$ is a Trigamma function, which is the derivative of $\psi(z)$, evaluated at *z*.

Digamma Function

The two main mathematical properties of the digamma function,

(1) Recurrence formulas

$$\psi(z+1) = \psi(z) + \frac{1}{z}$$

(2) Asymptotic formulas

$$\begin{aligned}\psi(z) &\sim \ln(z) - \frac{1}{2z} - \sum_{n=1}^{\infty} \frac{B_{2n}}{2nz^{2n}} \\ &= \ln(z) - \frac{1}{2z} - \frac{1}{12z^2} + \frac{1}{120z^4} - \frac{1}{252z^6} + \cdots \left(z \to \infty \text{ in } |\arg z| < \infty\right)\end{aligned}$$

The Bernoulli number B_n can be defined by the contour integral,

$$B_n = \frac{n!}{2\pi\, i} \oint \frac{z}{e^z - 1} \frac{dz}{z^{n+1}}$$

where the contour encloses the origin, has radius less than 2π, and is traversed in a counterclockwise direction (Arfken, 1985). The first few Bernoulli number B_n are

$$\begin{aligned}
B_0 &= 1\\
B_1 &= -\tfrac{1}{2}\\
B_2 &= \tfrac{1}{6}\\
B_4 &= -\tfrac{1}{30}\\
B_6 &= \tfrac{1}{42}\\
B_8 &= -\tfrac{1}{30}\\
B_{10} &= \tfrac{5}{66}\\
B_{12} &= -\tfrac{691}{2730}\\
B_{14} &= \tfrac{7}{6}\\
B_{16} &= -\tfrac{3617}{510}\\
B_{18} &= \tfrac{43867}{798}\\
B_{20} &= -\tfrac{174611}{330}\\
B_{22} &= \tfrac{854513}{138}
\end{aligned}$$

Therefore, we have following formula to calculate the digamma function,

$$\psi(x) = \psi(x+m) - \sum_{i=1}^{m} \frac{1}{x+i-1}$$

where *m* is a positive integer.

According two formulas above, we have the following computational algorithm of digamma function for real number *x*.

Algorithm 1: Digamma(x)

```
If ( $abs(x) \le 10^{-5}$ ) then
    $p = -0.577215664901532860606512 - \frac{1}{x}$ .
    Return ($p$).
End if.
$m$ = 10.
$x = x + m$.
$p = \frac{1}{x^2}$
$p = \left(\left(\left(\left(\left(\frac{691}{32760}p - \frac{1}{132}\right)p + \frac{1}{240}\right)p - \frac{1}{252}\right)p + \frac{1}{120}\right)p - \frac{1}{12}\right)p$ .
$p = p + \ln(x) - \frac{1}{2x}$.
```

```
For $i$ = 1 to $m$ do
    $p = p - \frac{1}{x-i}$
End for.
Return ($p$).
```

The algorithm 1 has a computed precision of $|\varepsilon| < 10^{-12}$, but in practice, appears to 15 significant digits for all positive real argument.

Trigamma Function

The *n*th derivative of $\psi(z)$ is called the polygamma function, denoted $\psi^{(n)}(z)$,

$$\psi^{(n)}(z) = \frac{d^n}{dz^n}\psi(z) = \frac{d^{n+1}}{dz^{n+1}}\ln\Gamma(x),\ n = 1,2,3,\cdots$$

Trigamma function, $\psi'(z)$, has two main mathematical properties,

(1) Recurrence formulas

$$\psi'(z+1) = \psi'(z) - z^{-2}$$

(2) Asymptotic formulas

$$\begin{aligned}\psi'(z) &\sim \frac{1}{z} + \frac{1}{2z^2} + \sum\nolimits_{k=1}^{\infty}\frac{B_{2k}}{z^{2k+1}} \\ &= \frac{1}{z} + \frac{1}{2z^2} + \frac{1}{6z^3} - \frac{1}{30z^5} + \frac{1}{42z^7} - \frac{1}{30z^9} + \cdots \\ &\qquad\qquad\left(z \to \infty \text{ in } |\arg z| < \infty\right)\end{aligned}$$

Similarly, we have following formula to calculate the digamma function,

$$\psi(x) = \psi(x+m) + \sum\nolimits_{i=1}^{m}\frac{1}{(x+i-1)^2}$$

where m is a positive integer.

According to two formula above, we have the following computational algorithm of trigamma function for real number *x*.

Algorithm 2: Trigamma(x)

```
If ( $abs(x) <= 10^{-4}$ ) then
    $p = \frac{1}{x \times x}$ .
    Return ($p$).
End for.
$m$ = 10.
$x = x + m$.
$p = \frac{1}{x^2}$ .
$p = \left(\left(\left(\left(\left(\left(-\frac{691}{2730}p+\frac{5}{66}\right)p-\frac{1}{30}\right)p+\frac{1}{42}\right)p-\frac{1}{30}\right)p+\frac{1}{6}\right)p+1\right)\frac{1}{x}+\frac{1}{2}p$ .
For $i$ = 1 to $m$ do
    $p = p + \frac{1}{(x-i)^2}$ .
End for
Return ($p$).
```

The algorithm 2 has a computed precision of $|\varepsilon| < 10^{-13}$, but in practice, appears to 15 significant digits for all positive real argument.

References – Phase I

[1]. Abramowitz, M. and Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing. New York: Dover. Chapter 6: Gamma and Related Functions.

[2]. Arfken, G. (1985). *Mathematical Methods for Physicists*, 3rd printing. Orlando, FL: Academic Press. Chapter 10: Digamma and Polygamma Functions.

[3]. Cameron, A. C. and Trivedi, P. K. (1998), *Regression Analysis of Count Data*, Cambridge: Cambridge University Press.

[4]. Chiu, T. (1995a), “The four types of sum of square for univariate β-model,” S*PSS Internal Document.*

[5]. Chiu, T. (1995b), “Calculation of the four types of sums of squares,” *SPSS Internal Document*.

[6]. Chu, J. and Zhong, W. (2005), “Algorithm: Generalized Linear Models and Generalized Estimating Equation,” *SPSS Internal Document.*

[7]. Chu, J. (2009), “Algorithm: Data Transformation in GENLIN,” *SPSS Internal Document.*

[8]. Chu, J. and Han, S. (2011), “Algorithm: Linear Engine”, *IBM SPSS Internal Document*.

[9]. Du, Z. and Zheng, P. (2009) “Algorithm: Unconstrained and Linearly Constrained Optimization,” *SPSS Internal Document.*

[10]. Dunn, P. K. and Smyth, G. K. (2005), “Series Evaluation of Tweedie Exponential Dispersion Model Densities,” *Statistics and Computing*, 15, 267–280.

[11]. Dunn, P. K. and Smyth, G. K. (2001), “Tweedie Family Densities: Methods of Evaluation,” *Proceedings of the 16th International Workshop on Statistical Modelling*, Odense, Denmark, 2–6 July.

[12]. Fang, D. P (2004), “Logistic Regression in Complex Sampling,” *SPSS Internal Document.*

[13]. Hardin, J. W. and Hilbe, J. M. (2001), *Generalized Linear Models and Extension*, College Station, TX: Stata Press.

[14]. Lam, M. L. (1995a), “Constructing the Design Matrix for the *β*-Model”, *SPSS Internal Document*.

[15]. Lam, M. L. (1995b), “Algorithm: the symmetric sweep operator”, *SPSS Internal Document*.

[16]. McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, Second Edition, London: Chapman and Hall.

[17]. Zhong, W. (2006a), “Algorithm: The construction of Type I and III L matrix”, *SPSS Internal Document.*

[18]. Zhong, W. (2006b), “Algorithm: Diagamma and trigamma”, *SPSS Internal Document.*

[19]. Dagli, A. (2012), “Simple random sampling in Map-Reduce”, *IBM SPSS Internal Document.*

6. Introduction – Phase II

Generalized Linear Engine Phase II (GLE Phase II) adds five main functions based on GLE Phase I (Chu and Zhong, 2012).

- Automatic two-way interaction detection.
- Model selection, including distribution, link function and effects.
- Influential outliers for all distributions except multinomial distribution.
- Diagnostic plots for all distributions except multinomial distribution.
- Grouping analysis for all distributions, and influential target category analysis for multinomial distributions and unusual categories detection for other distributions.

Section 7 describes automatic interaction detection. Section 8 describes model selection. Scoring and model diagnostics are presented in Section 9 and 10. In addition, Appendix A gives grouping analysis and unusual category detection.

7. Automatic two-way interaction detection

This section gives a method to detect two-way factor interaction $X_1 * X_2$ given specific probability distribution and link function, where X_1 and X_2 are two factors. In order to achieve this goal, log-likelihood ratio test between reduced model and full model is used. Here the reduced model means a GZLM in which only predictors X_1 and X_2 are involved, and full model means the model contains X_1 , X_2 and $X_1 * X_2$.

Since the computation will be complex for multinomial distribution, the log-likelihood ratio test for the distributions except the multinomial distribution is provided from Sections 7.1 to Section 7.4. Then Section 7.5 and 7.6 introduce nominal and ordinal multinomial distribution, respectively.

However, even with this original limitation, it might not be possible to check all candidate pairs of two factors for the model selection methods in Section 8. The reason is, if there are large number of main effects in $\boldsymbol{X}$, the whole process might require too much memory (so user might receive “run out of memory” message and no output at all) or too much computational cost (so user might wait for a long time to receive output). Hence, we provide a two-way-test pair search strategy to restrict number of the pairs in those which are more likely to be selected to the final model in the model selection method. See Section 7.7 for details.

7.1 Notations

The notations below are just used for distributions except multinomial distribution:

R	The total number of categories for factor X_1.
S	The total number of categories for factor X_2.
n_{ij}	The number of records in the combination $X_1 = i$ and $X_2 = j$.
y_{ijk}	The target value for the kth record in the combination $X_1 = i$ and $X_2 = j$. If the distribution if binomial(m), then $y_{ijk} = \frac{r_{ijk}}{m_{ijk}}$, where r_{ijk} and m_{ijk} are the events value and trials value, respectively.
f_{ijk}	The frequency weight for the kth record in the combination $X_1 = i$ and $X_2 = j$.
N_{ij}	The total number of cases in the combination $X_1 = i$ and $X_2 = j$.
$\bar{y}_{ij}$	The target mean in the combination $X_1 = i$ and $X_2 = j$.
α_i	The parameter of $X_1 = i$.
β_j	The parameter of $X_2 = j$.
μ_{ij}	The expectation of target in the combination of $X_1 = i$ and $X_2 = j$.

7.2 Basic statistics

The below basic statistics are needed to collect:

- The total number of records: $N_{ij} = \sum_{k=1}^{n_{ij}} f_{ijk}$
- Target mean: $\bar{y}_{ij} = \frac{\sum_{k=1}^{n_{ij}} f_{ijk} y_{ijk}}{N_{ij}}$
- The sum of square term of target: $c_{ij} = \sum_{k=1}^{n_{ij}} f_{ijk}\left(y_{ijk} - \bar{y}_{ij}\right)^2$

7.3 Two-way interaction test

The interaction test based on pseudo log-likelihood ratio test can be described as following steps:

1. Compute pseudo log-likelihood function, ℓ_{full} , for full model. Please see Section 7.4 for details.
2. Compute pseudo log-likelihood function, $\ell_{reduced}$ for the model. Please see section 7.4 for detail.
3. Estimate the scale parameter based on full model:

$$\hat{\phi} = \frac{1}{df} \sum_{i=1}^{R} \sum_{j=1}^{S} \frac{c_{ij}}{V(\bar{y}_{ij})}$$

where $df = \sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij} - R * S + c$, here c is the number of invalid categorical combinations.

4. Compute the log-likelihood ratio statistics

$$\chi^2 = \frac{2(\ell_{full} - \ell_{reduced})}{\hat{\phi}}$$

Compute the p value

$$p = 1 - Pr\,(\chi^2_{df} \le \chi^2)$$

where χ^2_{df} is the random variable following chi-square distribution with degree freedom $df = (R-1)*(S-1) - c$, where c is the number of invalid categorical combinations.

5. If $p \le \alpha$, where α is a significant level(the default is 0.05) then the interaction is significant.

7.4 Pseudo log-likelihood value computation

The pseudo log-likelihood functions for interaction detection are listed in the Table 7.1. Please note that compared with the kernels of log-likelihood function, some terms are omitted because these terms are the same for the full model and reduced model.

Table 7.1. Distribution and pseudo log-likelihood function

Target distribution	Pseudo log-likelihood
Normal	$\ell = -\frac{1}{2} \sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij} \left(\bar{y}_{ij} - \mu_{ij}\right)^2$
Inverse Gaussian	$\ell = -\frac{1}{2} \sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij} \left(\frac{\bar{y}_{ij} - 2\mu_{ij}}{\mu_{ij}^2}\right)$
Gamma	$\ell = -\sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij} \left(\ln \mu_{ij} + \frac{\bar{y}_{ij}}{\mu_{ij}}\right)$

Negative binomial	$\ell = \sum_{\mathrm{i=1}}^{\mathrm{R}} \sum_{\mathrm{j=1}}^{\mathrm{S}} N_{ij} \left(\bar{y}_{ij} \ln\left(k * \mu_{ij}\right) - \left(\bar{y}_{ij} + \frac{1}{k}\right) \ln\left(1 + k * \mu_{ij}\right) \right)$, where k is a parameter which will be specified by user. If user do not specify it, we automatically set it as 1.
Poisson	$\ell = \sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij} \left(\bar{y}_{ij} \ln\left(\mu_{ij}\right) - \mu_{ij} \right)$
Binomial	$\ell = \sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij} \left(\bar{y}_{ij} \ln\left(\mu_{ij}\right) + \left(1 - \bar{y}_{ij}\right) \ln\left(1 - \mu_{ij}\right) \right)$
Tweedie	$\ell = \sum_{\mathrm{i=1}}^{\mathrm{R}} \sum_{\mathrm{j=1}}^{\mathrm{S}} \mathrm{N_{ij}} \left(\frac{\bar{\mathrm{y}}_{\mathrm{ij}} * \mu_{\mathrm{ij}}^{1-\mathrm{q}}}{1-\mathrm{q}} + \frac{\mu_{\mathrm{ij}}^{2-\mathrm{q}}}{2-\mathrm{q}} \right)$, where q is a parameter which will be specified by user. If user do not specify it, we automatically set it as 1.5.

For the full model, ℓ_{full} can be got by above formula directly by replacing μ_{ij} with $\bar{y}_{ij}$.

For reduced model, the pseudo log-likelihood value will be computed by following iterative process:

1. Input values for T_1 (maximum number of iterations in the outer iterative process, tentatively set to 100), ε_1(tolerance level of stopping criterion in the outer iterative process, tentatively set to 10^{-6}), T_2 (maximum number of iterations in the inner iterative process, tentatively set to 5), ε_2(tolerance level of stopping criterion in the inner iterative process, tentatively set to 10^{-6})
2. Set initial values of $\alpha_i^{(0)} = 1.0E - 6$ and$\beta_j^{(0)} = 1.0E - 6$. Then compute expectation value, $\mu_{ij}^{(0)} = g^{-1}\left(\alpha_i^{(0)} + \beta_j^{(0)}\right)$, and initial pseudo log-likelihood value $\ell_{reduced}^{(0)}$ by plugging $\mu_{ij}^{(0)}$ into formulae in Table 1 .
3. Set the iteration number $t_1 = 1$.
4. Compute the weight, $w_{ij}^{(t_1-1)}$, and gradient, $s_{ij}^{(t_1-1)}$ in each combination of $X_1 = i, i = 1, \cdots, R$, and $X_2 = j, j = 1, \cdots, S$:

$$w_{ij}^{(t_1-1)} = \frac{N_{ij}}{V\left(\mu_{ij}^{(t_1-1)}\right)\left(g'\left(\mu_{ij}^{(t_1-1)}\right)\right)^2} + N_{ij}\left(\bar{y}_{ij} - \mu_{ij}^{(t_1-1)}\right) \times \frac{V\left(\mu_{ij}^{(t_1-1)}\right) g''\left(\mu_{ij}^{(t_1-1)}\right) + V'\left(\mu_{ij}^{(t_1-1)}\right) g'\left(\mu_{ij}^{(t_1-1)}\right)}{\left(V\left(\mu_{ij}^{(t_1-1)}\right)\right)^2 \left(g'\left(\mu_{ij}^{(t_1-1)}\right)\right)^3}$$

and

$$s_{ij}^{(t_1-1)} = \frac{1}{w_{ij}^{(t_1-1)}} \times \frac{N_{ij}\left(\bar{y}_{ij} - \mu_{ij}^{(t_1-1)}\right)}{V\left(\mu_{ij}^{(t_1-1)}\right) g'\left(\mu_{ij}^{(t_1-1)}\right)}$$

5. Compute the parameters increment α_i^* and β_j^* based on $w_{ij}^{(t_1-1)}$ and $s_{ij}^{(t_1-1)}$ with the following iterative process:

a) Create a $R \times S$ contingency table with the $w_{ij}^{(t_1-1)}$ and $s_{ij}^{(t_1-1)}$for each combination of $X_1 = i, i = 1, \cdots, R$ and $X_2 = j, j = 1, \cdots, S$:

X_2 X_1	1	2	$\cdots$	S
1	$w_{11}^{(t_1-1)}$, $s_{11}^{(t_1-1)}$	$w_{12}^{(t_1-1)}$, $s_{12}^{(t_1-1)}$	$\cdots$	$w_{1S}^{(t_1-1)}$, $s_{1S}^{(t_1-1)}$
2	$w_{21}^{(t_1-1)}$, $s_{21}^{(t_1-1)}$	$w_{22}^{(t_1-1)}$, $s_{22}^{(t_1-1)}$	$\cdots$	$w_{2S}^{(t_1-1)}$, $s_{2S}^{(t_1-1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
R	$w_{R1}^{(t_1-1)}$, $s_{R1}^{(t_1-1)}$	$w_{R2}^{(t_1-1)}$, $s_{R2}^{(t_1-1)}$	$\cdots$	$w_{RS}^{(t_1-1)}$, $s_{RS}^{(t_1-1)}$

b) Initial $\alpha_i^* = 0, i = 1, \cdots, R$ and $\beta_j^* = 0, j = 1, \cdots, S$, and the iteration number $t_2 = 1$.

c) Compute marginal mean of gradient for each row i, $i = 1, \cdots, R$

$$s_{i\cdot}^{(t_1-1)} = \frac{\sum_{j=1}^{S} s_{ij}^{(t_1-1)} \times w_{ij}^{(t_1-1)}}{\sum_{j=1}^{S} w_{ij}^{(t_1-1)}}$$

Update the α_i^* for each i, $i = 1, \cdots, R$:

$$\alpha_i^* = \alpha_i^* + s_{i\cdot}^{(t_1-1)}$$

Update the $s_{ij}^{(t_1-1)}$ for $i = 1, \cdots, R$ and $j = 1, \cdots, S$ in the table

$$s_{ij}^{(t_1-1)} = s_{ij}^{(t_1-1)} - s_{i\cdot}^{(t_1-1)}$$

d) Based on the updated table, compute the marginal mean of gradient for each column j, $j = 1, \cdots, S$:

$$s_{\cdot j}^{(t_1-1)} = \frac{\sum_{i=1}^{R} s_{ij}^{(t_1-1)} \times w_{ij}^{(t_1-1)}}{\sum_{i=1}^{R} w_{ij}^{(t_1-1)}}$$

Update the β_j^* for each j, $j = 1, \cdots, S$:

$$\beta_j^* = \beta_j^* + s_{\cdot j}^{(t_1-1)}$$

Update the $s_{ij}^{(t_1-1)}$ for $i = 1, \cdots, R$ and $j = 1, \cdots, S$ in the table

$$s_{ij}^{(t_1-1)} = s_{ij}^{(t_1-1)} - s_{\cdot j}^{(t_1-1)}$$

e) If $\max\left\{\left|s_{i\cdot}^{(t_1-1)}\right|, \left|s_{\cdot j}^{(t_1-1)}\right|\right\} \le \varepsilon_2$ or $t_2 > T_2$, then stop and output the parameter α_i^* and β_j^*. Otherwise let $t_2 = t_2 + 1$, and go to step c).

6. Update parameter estimates for iteration

$$\alpha_i^{(t_1)} = \alpha_i^{(t_1-1)} + \alpha_i^*$$
$$\beta_i^{(t_1)} = \beta_i^{(t_1-1)} + \beta_j^*$$

then update expectation value

$$\mu_{ij}^{(t_1)} = g^{-}\left(\alpha_i^{(t_1)} + \beta_j^{(t_1)}\right)$$

And pseudo log-likelihood value $\ell_{reduced}^{(t_1)}$ based on $\mu_{ij}^{(t_1)}$ using the formula listed in Table 1.

7. If $\ell_{reduced}^{(t_1)} < \ell_{reduced}^{(t_1-1)}$, then stop and output $\ell_{reduced}^{(t_1-1)}$.
8. If $|\ell_{reduced}^{(t_1)} - \ell_{reduced}^{(t_1-1)}| < \varepsilon_1$ or $t_1 > T_1$, then stop and output $\ell_{reduced}^{(t_1)}$, otherwise, $t_1 = t_1 + 1$, go back to step 4.

7.5 Nominal multinomial distribution

This sub-section discusses the interaction detection for nominal multinomial distribution using log-likelihood ratio test.

The following notations are used for nominal multinomial distributio*n*

J	The total number of categories for target variable.
R	The total number of categories for factor X_1.
S	The total number of categories for factor X_2.
n_{ij}	The number of records in the combination $X_1 = i$ and $X_2 = j$.
f_{ijm}	The frequency weight for the mth record in the combination $X_1 = i$ and $X_2 = j$.
y_{ijm}	The target value for the mth record in the combination $X_1 = i$ and $X_2 = j$.
$\hat{\boldsymbol{\pi}}_{ij}$	$\hat{\boldsymbol{\pi}}_{ij} = \left(\hat{\pi}_{ij,1}, \cdots, \hat{\pi}_{ij,J-1}\right)^T$, where $\hat{\pi}_{ij,k}$ is the estimated probability of the kth target category when $X_1 = i$ and $X_2 = j$ and the superscript T means transpose of a matrix or vector. Please note that $\hat{\pi}_{ij,J} = 1 - \sum_{k=1}^{J-1} \hat{\pi}_{ij,k}$ and is not included in the vector $\hat{\boldsymbol{\pi}}_{ij}$.
$\boldsymbol{\alpha}_i$	$\boldsymbol{\alpha}_i = \left(\alpha_{i1}, \cdots, \alpha_{iJ-1}\right)^T$, where $\alpha_{ik}$is the parameter of $X_1 = i$ for the kth target category.
$\boldsymbol{\beta}_j$	$\boldsymbol{\beta}_j = \left(\beta_{j1}, \cdots, \beta_{jJ-1}\right)^T$, where β_{jk} is the parameter of $X_2 = j$ for the kth target category.

The interaction detection for nominal multinomial distribution is similar to that in the previous sections. Therefore, just some implementation notes are given as following:

Implementation notes:

(1) The following basic statistics are needed to collect:

- The total number of records for the k$^{\text{th}}$ target category in the combination $X_1 = i$ and $X_2 = j$:

$$N_{ij,k} = \sum_{m=1}^{n_{ij}} f_{ijm} * I(y_{ijm} = k)$$

where $(\mathrm{y_{ijm}} = \mathrm{k}) = \begin{cases} 1, & \mathrm{y_{ijm}} = \mathrm{k} \\ 0, & otherwise \end{cases}$.

- The observed probability of the k$^{\text{th}}$ target category in the combination $X_1 = i$ and $X_2 = j$:

$$\bar{y}_{ij,k} = \frac{N_{ij,k}}{N_{ij}}$$

- The total number of records in the combination $\mathrm{X_1 = i}$ and $\mathrm{X_2 = j}$:

$$N_{ij} = \sum_{k=1}^{J} N_{ij,k}$$

(2) The scale parameter is estimated as 1.

(3) The degree of freedom of the log-likelihood ratio test is $(\mathrm{R} * \mathrm{S} - \mathrm{c}) * (\mathrm{J} - 1)$, where c is the number of invalid categorical combinations.

(4) The parameters, α_i and β_j , are extended to vector, $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$.

(5) Since only the logit link function will used for nominal multinomial distribution, the log-likelihood value will be computed as

$$\ell = \sum_{i=1}^{R} \sum_{j=1}^{S} \sum_{k=1}^{J} N_{ij,k} * \ln(\pi_{ij,k})$$

For the full model, the $\pi_{ij,k}$ will be estimated as $\hat{\pi}_{ij,k} = \bar{y}_{ij,k}$ for $k = 1, \cdots, J-1$, and $\hat{\pi}_{ij,k} = 1 - \sum_{k=1}^{J-1} \bar{y}_{ij,k}$. And for the reduced model, the $\pi_{ij,k}$ will be estimated by the parameter estimates and link function, i.e. suppose we have the parameter estimates $\hat{\boldsymbol{\alpha}}_i = (\hat{\alpha}_{i1}, \cdots, \hat{\alpha}_{iJ-1})^T$ and $\hat{\boldsymbol{\beta}}_j = (\hat{\beta}_{j1}, \cdots, \hat{\beta}_{jJ-1})^T$, then

$$\hat{\pi}_{ij,k} = \begin{cases} \dfrac{\exp(\hat{\alpha}_{ik} + \hat{\beta}_{jk})}{1 + \sum_{k=1}^{J-1} \exp(\hat{\alpha}_{ik} + \hat{\beta}_{jk})}, & k = 1, \cdots, J-1 \\ \dfrac{1}{1 + \sum_{k=1}^{J-1} \exp(\hat{\alpha}_{ik} + \hat{\beta}_{jk})} & k = J \end{cases}$$

(6) The weight, $w_{ij}^{(t_1-1)}$, and mean of score, $s_{ij}^{(t_1-1)}$ in Section 2.4 will be matrix $\boldsymbol{w}_{ij}^{(t_1-1)}$ and vector $\boldsymbol{s}_{ij}^{(t_1-1)}$ which can be computed as:

$$\boldsymbol{w}_{ij}^{(t_1-1)} = N_{ij} * \left(diag\left(\boldsymbol{\pi}_{ij}^{(t_1-1)}\right) - \boldsymbol{\pi}_{ij}^{(t_1-1)} * \left(\boldsymbol{\pi}_{ij}^{(t_1-1)}\right)^T\right)$$

and

$$\boldsymbol{s}_{ij}^{(t_1-1)} = \left(\boldsymbol{w}_{ij}^{(t_1-1)}\right)^{-1} * N_{ij} * \left(\bar{\boldsymbol{y}}_{ij} - \boldsymbol{\pi}_{ij}^{(t_1-1)}\right)$$

where $\bar{y}_{ij} = \left(\bar{y}_{ij,1}, \cdots, \bar{y}_{ij,J-1}\right)^{\mathrm{T}}$.

(7) The marginal mean for each row and column in Section 2.4 can be computed as

$$\boldsymbol{s}_{i\cdot}^{(t_1-1)} = \left(\sum_{j=1}^{S} \boldsymbol{w}_{ij}^{(t_1-1)}\right)^{-1} * \left(\sum_{j=1}^{S} \boldsymbol{w}_{ij}^{(t_1-1)} * \boldsymbol{s}_{ij}^{(t_1-1)}\right)$$

and

$$\boldsymbol{s}_{\cdot j}^{(t_1-1)} = \left(\sum_{i=1}^{R} \boldsymbol{w}_{ij}^{(t_1-1)}\right)^{-1} * \left(\sum_{i=1}^{R} \boldsymbol{w}_{ij}^{(t_1-1)} * \boldsymbol{s}_{ij}^{(t_1-1)}\right)$$

respectively.

7.6 Ordinal multinomial distribution

This sub-section discusses the interaction detection for ordinal multinomial distribution using log-likelihood ratio test.

The following notations are used if the distribution is ordinal multinomial.

J	The total number of categories for target variable.
R	The total number of categories for predictor X_1.
S	The total number of categories for predictor X_2.
n_{ij}	The number of records in the combination $X_1 = i$ and $X_2 = j$.
y_{ijm}	The target value for the mth record in the combination $X_1 = i$ and $X_2 = j$.
f_{ijm}	The frequency weight for the mth record in the combination $X_1 = i$ and $X_2 = j$.
$\gamma_{ij,k}$	Conditional cumulative target probability for the kth category in the combination $\mathrm{X}_1 = \mathrm{i}$ and $\mathrm{X}_2 = \mathrm{j}$.
$\pi_{ij,\mathrm{k}}$	Conditional target probability for for the kth category in the combination $X_1 = i$ and $X_2 = j$, $\pi_{ij,\mathrm{k}} = \gamma_{ij,k} - \gamma_{ij,k-1}$.

$\boldsymbol{\Psi}$	$(J-1)\times 1$ vector of threshold parameter, $\boldsymbol{\Psi} = \left(\psi_1,\cdots,\psi_{J-1}\right)^T$ and $\psi_1 < \psi_2 < \cdots < \psi_{J-1}$.
α_i	The parameter of $X_1 = i$.
$\boldsymbol{\alpha}$	$(R-1)\times 1$ parameter vector , $\boldsymbol{\alpha} = (\alpha_1,\cdots,\alpha_{R-1})^T$.
β_j	The parameter of $X_2 = j$.
$\boldsymbol{\beta}$	$(S-1)\times 1$ parameter vector , $\boldsymbol{\beta} = (\beta_1,\cdots,\beta_{S-1})^T$.
λ_{ij}	The parameter of the combination of $X_1 = i$ and $X_2 = j$.
$\boldsymbol{\lambda}$	$R \times S$ parameter vector, $\boldsymbol{\lambda} = (\lambda_{11},\cdots,\lambda_{1S},\cdots\lambda_{R1},\cdots,\lambda_{RS})^T$.
$\eta_{ij,k}$	Linear predictor value for the k^{th} target category in the combination $X_1 = i$ and $X_2 = j$.

Basic statistics

The following basic statistics are needed to collect:

- The total number of records for the k^{th} target category in the combination $X_1 = i$ and $X_2 = j$:

$$N_{ij,k} = \sum\nolimits_{m=1}^{n_{ij}} f_{ijm} * I(y_{ijm} = k)$$

where $(y_{ijm} = k) = \begin{cases} 1, & y_{ijm} = k \\ 0, & otherwise \end{cases}$.

- The observed probability of the k^{th} target category in the combination $X_1 = i$ and $X_2 = j$:

$$\bar{y}_{ij,k} = \frac{N_{ij,k}}{N_{ij}}$$

- The total number of records in the combination $X_1 = i$ and $X_2 = j$:

$$N_{ij} = \sum\nolimits_{k=1}^{J} N_{ij,k}$$

Interaction detection

The interaction detection is also based on the log-likelihood ration test. Please note that

- The reduced model is

$$\eta_{ij,k} = g\left(\gamma_{ij,k}\right) = \psi_k - \alpha_i - \beta_j$$

where $k = 1,\cdots,J-1,\ i = 1,\cdots,R$ and $j = 1,\cdots,S$. And the full model is

$$\eta_{ij,k} = g\left(\gamma_{ij,k}\right) = \psi_k - \lambda_{ij}$$

where $k = 1, \cdots, J-1,\ \ i = 1, \cdots, R$ and $j = 1, \cdots, S$.

- The scale parameter is 1.
- The degree of freedom of log-likelihood ratio test is $df = (R-1) * (S-1) - c$, where c is the number of invalid categorical combinations.

Log-likelihood value

The log-likelihood value is

$$\ell = \sum_{i=1}^{R} \sum_{j=1}^{S} \sum_{k=1}^{J} N_{ij,k} * \ln\left(\pi_{ij,k}\right)$$

For both full model and reduced model, the log-likelihood value will be computed by following iterative process:

1. Input values for T_1 (maximum number of iterations, tentatively set to 100), ε_1(tolerance level of stopping criterion, tentatively set to 10^{-6}) .
2. Set initial values:

   $\alpha_i^{(0)} = 0, i = 1, \cdots, R;$

   $\beta_j^{(0)} = 0, j = 1, \cdots, S;$

   and

   $$\psi_k^{(0)} = g\left(\frac{\sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij,k}}{\sum_{i=1}^{R} \sum_{j=1}^{S} N_{ij}}\right), k = 1, \cdots, J-1$$

3. Compute $\pi_{ij,k}^{(0)} = \gamma_{ij,k}^{(0)} - \gamma_{ij,k-1}^{(0)}$ for $k = 1, \cdots, J-1$ and

   $$\gamma_{ij,k}^{(0)} = \begin{cases} 0 & k = 0 \\ g^{-1}\left(\psi_k^{(0)} - \alpha_i^{(0)} - \beta_j^{(0)}\right) & k = 1, \cdots, J-1 \\ 1 & k = J \end{cases}$$

   Then compute log-likelihood value $\ell^{(0)}$ based on $\pi_{ij,k}^{(0)}$ and $N_{ij,k}$ using the formula of log-likelihood value for ordinal distribution.

4. Set the iteration number $t_1 = 1$.
5. Compute estimates of t_1th iteration

   $$\mathbf{B}^{(t_1)} = \mathbf{B}^{(t_1-1)} - \left(\boldsymbol{H}^{(t_1-1)}\right)^{-} \boldsymbol{s}^{(t_1-1)}$$

   where $\mathbf{B} = (\boldsymbol{\Psi}^{\boldsymbol{T}}, \boldsymbol{\alpha}^{\boldsymbol{T}}, \boldsymbol{\beta}^{\boldsymbol{T}})^{\boldsymbol{T}}$ if model is reduced model and $\mathbf{B} = (\boldsymbol{\Psi}^{\boldsymbol{T}}, \boldsymbol{\lambda}^{\boldsymbol{T}})^{\boldsymbol{T}}$ if model is full model. The hessian matrix $\boldsymbol{H}$ and gradient vector **s** will be computed later, and $(\boldsymbol{H})^{-}$ is the generalized inverse of $\boldsymbol{H}$**.**

6. Similar to the step 3, compute $\pi_{ij,k}^{(t_1)}$ and log-likelihood value $\ell^{(t_1)}$.
7. If $\ell_{reduced}^{(t_1)} < \ell_{reduced}^{(t_1-1)}$, then stop and output $\ell_{reduced}^{(t_1-1)}$.
8. If $|\ell^{(t_1)} - \ell^{(t_1-1)}\ | < \varepsilon_1$ or $t_1 > T_1$, then stop and output $\ell^{(\ t_1)}$, otherwise, $t_1 = t_1 + 1$, go back to step 5.

Gradient vector and hessian matrix for reduced model

The gradient vector $\boldsymbol{s}$ can be computed as following:

$$\boldsymbol{s} = \left[\frac{\partial\ell}{\partial\psi_1}, \cdots, \frac{\partial\ell}{\partial\psi_{J-1}}, \frac{\partial\ell}{\partial\alpha_1}, \cdots, \frac{\partial\ell}{\partial\alpha_{R-1}}, \frac{\partial\ell}{\partial\beta_1}, \cdots, \frac{\partial\ell}{\partial\beta_{J-1}}\right]^T,$$

where

$$\frac{\partial\ell}{\partial\psi_k} = \sum_{i=1}^{R}\sum_{j=1}^{S}\frac{\partial\gamma_{ij,k}}{\partial\eta_{ij,k}}\left(\frac{N_{ij,k}}{\pi_{ij,k}} - \frac{N_{ij,k+1}}{\pi_{ij,k+1}}\right), k = 1, \cdots, J-1,$$

$$\frac{\partial\ell}{\partial\alpha_i} = -\sum_{j=1}^{S}\sum_{k=1}^{J}\left(\frac{\partial\gamma_{ij,k}}{\partial\eta_{ij,k}} - \frac{\partial\gamma_{ij,k-1}}{\partial\eta_{ij,k-1}}\right)\frac{N_{ij,k}}{\pi_{ij,k}},\ \ i = 1, \cdots, R-1,$$

$$\frac{\partial\ell}{\partial\beta_j} = -\sum_{i=1}^{R}\sum_{k=1}^{J}\left(\frac{\partial\gamma_{ij,k}}{\partial\eta_{ij,k}} - \frac{\partial\gamma_{ij,k-1}}{\partial\eta_{ij,k-1}}\right)\frac{N_{ij,k}}{\pi_{ij,k}},\ \ j = 1, \cdots, S-1,$$

And the hessian matrix is

$$H = \begin{pmatrix} \frac{\partial^2\ell}{\partial\boldsymbol{\Psi}\partial\boldsymbol{\Psi}^T} & \frac{\partial^2\ell}{\partial\boldsymbol{\Psi}\partial\boldsymbol{\alpha}^T} & \frac{\partial^2\ell}{\partial\boldsymbol{\Psi}\partial\boldsymbol{\beta}^T} \\ \frac{\partial^2\ell}{\partial\boldsymbol{\alpha}\partial\boldsymbol{\Psi}^T} & \frac{\partial^2\ell}{\partial\boldsymbol{\alpha}\partial\boldsymbol{\alpha}^T} & \frac{\partial^2\ell}{\partial\boldsymbol{\alpha}\partial\boldsymbol{\beta}^T} \\ \frac{\partial^2\ell}{\partial\boldsymbol{\beta}\partial\boldsymbol{\Psi}^T} & \frac{\partial^2\ell}{\partial\boldsymbol{\beta}\partial\boldsymbol{\alpha}^T} & \frac{\partial^2\ell}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} \end{pmatrix},$$

where

$$\frac{\partial^2\ell}{\partial\psi_{k-1}\partial\psi_k} = \sum_{i=1}^{R}\sum_{j=1}^{S}\frac{\partial\gamma_{ij,k-1}}{\partial\eta_{ij,k-1}}\frac{\partial\gamma_{ij,k}}{\partial\eta_{ij,k}}\frac{N_{ij,k}}{\pi_{ij,k}^2}, k = 2, \cdots, J-1,$$

$$\frac{\partial^2\ell}{\partial\psi_k^2} = \sum_{i=1}^{R}\sum_{j=1}^{S}\left[\frac{\partial^2\gamma_{ij,k}}{\partial\eta_{ij,k}^2}\left(\frac{N_{ij,k}}{\pi_{ij,k}} - \frac{N_{ij,k+1}}{\pi_{ij,k+1}}\right) - \left(\frac{\partial\gamma_{ij,k}}{\partial\eta_{ij,k}}\right)^2\left(\frac{N_{ij,k}}{\pi_{ij,k}^2} + \frac{N_{ij,k+1}}{\pi_{ij,k+1}^2}\right)\right], k = 1, \cdots, J-1,$$

$$\frac{\partial^2\ell}{\partial\psi_k\partial\psi_m} = 0,\ \ \text{for } |k-m| > 1,$$

$$\frac{\partial^2\ell}{\partial\alpha_i^2} = \sum_{j=1}^{S}\sum_{k=1}^{J}\left[\left(\frac{\partial^2\gamma_{ij,k}}{\partial\eta_{ij,k}^2} - \frac{\partial^2\gamma_{ij,k-1}}{\partial\eta_{ij,k-1}^2}\right)\pi_{ij,\mathrm{k}} - \left(\frac{\partial\gamma_{ij,k}}{\partial\eta_{ij,k}} - \frac{\partial\gamma_{ij,k-1}}{\partial\eta_{ij,k-1}}\right)^2\right]\left(\frac{N_{ij,k}}{\pi_{ij,k}^2}\right), i = 1, \cdots, R-1,$$

$$\frac{\partial^2 \ell}{\partial \alpha_i \partial \alpha_j} = 0, \text{for } i \neq j,$$

$$\frac{\partial^2 \ell}{\partial \beta_j^2} = \sum_{i=1}^{R} \sum_{k=1}^{J} \left[\left(\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} - \frac{\partial^2 \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}^2} \right) \pi_{ij,\mathrm{k}} - \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right)^2 \right] \left(\frac{N_{ij,k}}{\pi_{ij,k}^2} \right), j = 1, \cdots, S-1,$$

$$\frac{\partial^2 \ell}{\partial \beta_i \partial \beta_j} = 0, \text{for } i \neq j,$$

$$\frac{\partial^2 \ell}{\partial \psi_k \partial \alpha_i} = -\sum_{j=1}^{S} \left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \pi_{ij,\mathrm{k}} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right) \right] \frac{N_{ij,k}}{\pi_{ij,k}^2}$$
$$+ \sum_{j=1}^{S} \left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \pi_{ij,\mathrm{k}+1} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{\partial \gamma_{ij,k+1}}{\partial \eta_{ij,k+1}} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \right) \right] \frac{N_{ij,k+1}}{\pi_{ij,k+1}^2},$$

$$k = 1, \cdots, J-1, \;\; i = 1, \cdots, R-1,$$

$$\frac{\partial^2 \ell}{\partial \psi_k \partial \beta_j} = -\sum_{i=1}^{R} \left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \pi_{ij,\mathrm{k}} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right) \right] \frac{N_{ij,k}}{\pi_{ij,k}^2}$$
$$+ \sum_{i=1}^{R} \left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \pi_{ij,\mathrm{k}+1} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{\partial \gamma_{ij,k+1}}{\partial \eta_{ij,k+1}} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \right) \right] \frac{N_{ij,k+1}}{\pi_{ij,k+1}^2},$$

$$k = 1, \cdots, J-1, \;\; j = 1, \cdots, S-1,$$

$$\frac{\partial^2 \ell}{\partial \alpha_i \partial \beta_j} = \sum_{k=1}^{J} \left[\left(\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} - \frac{\partial^2 \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}^2} \right) \pi_{ij,\mathrm{k}} - \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right)^2 \right] \frac{N_{ij,k}}{\pi_{ij,k}^2},$$

$$i = 1, \cdots, R-1, j = 1, \cdots, S-1$$

Gradient vector and hessian matrix for full model

The gradient vector $\boldsymbol{s}$ can be computed as following:

$$\boldsymbol{s} = \left[\frac{\partial \ell}{\partial \psi_1}, \cdots, \frac{\partial \ell}{\partial \psi_{J-1}}, \frac{\partial \ell}{\partial \lambda_{11}}, \cdots, \frac{\partial \ell}{\partial \lambda_{1S}}, \cdots, \frac{\partial \ell}{\partial \lambda_{21}}, \cdots, \frac{\partial \ell}{\partial \lambda_{RS}} \right]^T,$$

where

$$\frac{\partial \ell}{\partial \psi_k} = \sum_{i=1}^{R} \sum_{j=1}^{S} \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{N_{ij,k}}{\pi_{ij,k}} - \frac{N_{ij,k+1}}{\pi_{ij,k+1}} \right), k = 1, \cdots, J-1,$$

$$\frac{\partial \ell}{\partial \lambda_{ij}} = -\sum_{k=1}^{J} \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right) \frac{N_{ij,k}}{\pi_{ij,k}}, \;\; i = 1, \cdots, R, j = 1, \cdots, S,$$

And the hessian matrix is

$$H = \begin{pmatrix} \frac{\partial^2 \ell}{\partial \boldsymbol{\Psi} \partial \boldsymbol{\Psi}^T} & \frac{\partial^2 \ell}{\partial \boldsymbol{\Psi} \partial \boldsymbol{\lambda}^T} \\ \frac{\partial^2 \ell}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\Psi}^T} & \frac{\partial^2 \ell}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\lambda}^T} \end{pmatrix},$$

where

$$\frac{\partial^2 \ell}{\partial \psi_{k-1} \partial \psi_k} = \sum_{i=1}^{R} \sum_{j=1}^{S} \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \frac{N_{ij,k}}{\pi_{ij,k}^2}, k = 2, \cdots, J-1,$$

$$\frac{\partial^2 \ell}{\partial \psi_k^2} = \sum_{i=1}^{R} \sum_{j=1}^{S} \left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \left(\frac{N_{ij,k}}{\pi_{ij,k}} - \frac{N_{ij,k+1}}{\pi_{ij,k+1}} \right) - \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \right)^2 \left(\frac{N_{ij,k}}{\pi_{ij,k}^2} + \frac{N_{ij,k+1}}{\pi_{ij,k+1}^2} \right) \right], k = 1, \cdots, J-1,$$

$$\frac{\partial^2 \ell}{\partial \psi_k \partial \psi_m} = 0, \text{ for } |k-m| > 1,$$

$$\frac{\partial^2 \ell}{\partial \lambda_{ij}^2} = \sum_{k=1}^{J} \left[\left(\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} - \frac{\partial^2 \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}^2} \right) \pi_{ij,\mathrm{k}} - \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right)^2 \right] \frac{N_{ij,k}}{\pi_{ij,k}^2}, i = 1, \cdots, R, j = 1, \cdots, S,$$

$$\frac{\partial^2 \ell}{\partial \lambda_{ij} \partial \lambda_{uv}} = 0, \text{ for } i \neq u \text{ or } j \neq v,$$

$$\frac{\partial^2 \ell}{\partial \psi_k \partial \lambda_{ij}} = -\left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \pi_{ij,\mathrm{k}} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} - \frac{\partial \gamma_{ij,k-1}}{\partial \eta_{ij,k-1}} \right) \right] \frac{N_{ij,k}}{\pi_{ij,k}^2} + \left[\frac{\partial^2 \gamma_{ij,k}}{\partial \eta_{ij,k}^2} \pi_{ij,\mathrm{k}+1} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \left(\frac{\partial \gamma_{ij,k+1}}{\partial \eta_{ij,k+1}} - \frac{\partial \gamma_{ij,k}}{\partial \eta_{ij,k}} \right) \right] \frac{N_{ij,k+1}}{\pi_{ij,k+1}^2},$$

$$k = 1, \cdots, J-1, i = 1, \cdots, R, j = 1, \cdots, S,$$

7.7 Two-way-test pair search strategy

Suppose there are m_{fac} factors and m_{cov} covariates, thus there are $m (= m_{\text{fac}} + m_{\text{cov}})$ main effects. Suppose the number of parameters for them is p^m (including the intercept).

Input values (integers) for m_1 (threshold value to conduct interaction effect detection; the default is 100), m_2 (threshold value to select main effects (factors) for interaction effect detection; the default is 20) and p^{max} (maximum number of parameters the system can handle; the default is 5000), where $m_1 \geq m_2$.

When ($\mathrm{p}^{\max} \leq \mathrm{p}^{m} + m_{\mathrm{cov}}$), the strategy will not be conducted. That is to say, no any interactions of two factors and squared term of covariates are output.

When ($m_{\mathrm{fac}} < 2$) and ($\mathrm{p}^{\max} > \mathrm{p}^{\mathrm{m}} + m_{\mathrm{cov}}$), only squared term of covariates are output.

When($m_{\mathrm{fac}} > m_2$) and ($\mathrm{p}^{\max} > \mathrm{p}^{\mathrm{m}} + m_{\mathrm{cov}}$) then the strategy will be conducted with the following steps:

1. Build a generalized linear model using all main effects $\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_m$.
2. Select the significant main effects ($p < 0.05$) based on Type 3 analysis (using Wald statistics in Section 3.1.1). Assume there are m' significant effects and m'_{fac} significant effects of factors.
3. If ($m'_{\mathrm{fac}} < 2$) or ($m' > m_1$), then stop and no interaction detection is conducted. Otherwise, sort the main effects using p-value in ascending order.
4. Select the top $m''_{\mathrm{fac}}(= \min(m'_{\mathrm{fac}}, m_2))$ main effects to construct two-way interaction effects (of two factors) among these m''_{fac} main effects.
5. Test all candidate interaction effects using the methods given Sections 7.3 and 7.6.
6. Calculate the total number of parameters for all significant interaction effects, denoted by p^{inter}, if $p^{inter} < 0.5 \times (p^{\max} - p^{m} - m_{\mathrm{cov}})$, then stop and output all significant interaction effects and all squared term of covariates; otherwise go to step 7.
7. Calculate effect size for each significant two-way interaction effect

   $$EffectSize = \ell_{full} - \ell_{reduced} - df_{interaction}$$

   where $df_{interaction}$ denotes the difference of degrees of freedom of full and reduced models.
8. Sort all significant two-way interaction effects using their effect sizes in descending order and select and output top-k interaction effects and all squared term of covariates, where k is the maximum number satisfying the number of parameters for top-k interaction effects is less than or equal to $0.5 \times (p^{\max} - p^{m} - m_{\mathrm{cov}})$.

When ($m_{\mathrm{fac}} \leq m_2$) and ($p^{\max} > p^{\mathrm{m}} + m_{\mathrm{cov}}$), the strategy will be similar to the one given above, except that the step of constructing two-way interaction effects: here, two-way interaction effects are directly constructed among all m_{fac} main effects rather than based on m''_{fac} significant main effects. That is to say, it doesn’t need building a model of all main effects.

Note that the parameters settings for the model of all main effects may be different from the user’s setting for speeding up the process of the two-way interaction detection:

- For the ancillary parameter (k) in negative binomial distribution, it is set to 1.0 when k is estimated by MLE.
- For the scale parameter ϕ, it is fixed to 1.0 for the following two cases: (1) the ϕ is estimated by MLE; (2) the ϕ is estimated by Pearson chi-square or Deviance divided by degree of freedom.

8. Model selection

Model selection for generalized linear models involves 2 aspects:

(1) Distribution and/or link function specification: if both or one of them is unspecified, then we need to select them which would be based on measurement level and storage type of the target.

(2) Variable selection or regularization: the option of variable selection can be on or off. If it is on, then the available methods are forward stepwise, lasso (L1 regularization), elastic net (L1+L2 regularization) and ridge regression (L2 regularization).

Notes:

(a) We assume that the inputs list is given.

(b) Two-way interaction detection is a sub-option under variable selection. Only when the variable selection flag is on and the variable selection or regarulization method is selected as Forward-Stepwise/L1/L1+L2/L2, user can specify the two-way interaction flag (default is off). Interaction detection is disabled if the user specifies any higher-order effects (beyond main effects).

Hence there will be 4 scenarios from the combinations of the above 2 aspects and we will describe how each scenario would be processed:

(1) Distribution and link function are specified and variable selection flag is off:

The main task is parameter estimation and the estimation methods would be different depending on the inputs list size:

(1.1) If the inputs list falls into the small to median p situation, then use the Newton-Raphson in GLE phase 1 to estimate parameters (by whole data).

Note that the list with null or intercept-only will fall into this scenario.

(1.2) If the inputs list falls into the large *p* situation, then use the L-BFGS in ADMM to estimate parameters (by whole data).

(2) Distribution and link function are specified and variable selection flag is on.

The main tasks are variable selection and parameter estimation:

(2.1) If the inputs list falls into the small to median *p* situation and the variable selection method is forward stepwise, then apply Section 8.1 to select variables (by sample data) and use the Newton-Raphson to estimate parameters (by whole data).

(2.2) If the inputs list falls into the large *p* situation and the variable selection method is forward stepwise, switch to the lasso method in ADMM to select variable and estimate parameters (by whole data). A warning will be issued to let user know that variable selection method is changed.

(2.3) If the variable selection or regularization is L1, L2, or L1+L2, no matter whether the inputs list falls into the small to medium or large *p* situation, use ADMM with Newton Raphson or L-BFGS to select variable and estimate parameters (by whole data).

Note that for the list with null or intercept-only in (2.1) and (2.3), we use the Newton-Raphson to build a null or intercept only model directly (by whole data) and issue a warning message such as "Variable selection method is ignored because of no predictor. A null model or intercept-only model is built."

(3) Distribution and/or link function are unspecified and variable selection is off.

The main task is distribution/link function selection:

(3.1) If the inputs list falls into the small to medium *p* situation, then apply Section 8.2 to select distribution and/or link function (by sample data) and use the Newton-Raphson to estimate parameters (by whole data) for the selected distribution/link function.

Note that the list with null or intercept-only will fall into this scenario.

(3.2) If the inputs list falls into the large *p* situation, then apply Section 8.2 to select distribution and/or link function (by sample data) and use the L-BFGS in ADMM ADD to estimate parameters (by whole data) for the selected distribution/link function.

(4) Distribution and/or link function are unspecified and variable selection is on.

The main tasks are distribution/link function selection, variable selection and parameter estimation:

(4.1) If the inputs list falls into the small to medium *p* situation and the variable selection method is forward stepwise, then apply Section 8.3.1 to select distribution/link function and variables (by sample data) and use the Newton-Raphson to estimate parameters (by whole data) for the selected distribution/link function.

(4.2) If the inputs list falls into the large *p* situation and the variable selection method is forward stepwise, then apply Section 8.3.2, i.e., switch to the lasso method in ADMM to select distribution/link function (by sample data), and use the lasso method in ADMM and selected lambda to select variables and estimate parameters (by whole data) for the selected distribution/link function. A warning will be issued to let user know that variable selection method is changed.

(4.3) If the variable selection/regularization is L1, L2, or L1+L2, no matter whether the inputs list falls into the small to medium or large *p* situation, apply Section 8.3.2 (based on ADMM) to select distribution/link function (by sample data), and use with Newton Raphson or L-BFGS with selected lambda to select variables and estimate parameters (by whole data) for the selected distribution/link function.

Note that for the list with null or intercept-only in (4.1) or (4.3), we apply Section 8.2 to select distribution/link function based on the null or intercept only model (by sample data), and use the Newton-Raphson to estimate parameters (by whole data) for the selected distribution/link function. Issue a warning message such as "Variable selection method is ignored because of no predictor. A null model or intercept-only model is built to select distribution and link function."

Implementation notes:

Regarding the initial value when using ADMM for variable selection and model building:

- When the inputs list falls into the small to medium situation, compute the initial value according to section 3.1.3.1 in GLE phase 1.
- When the inputs list falls into the large *p* situation and the distribution is not binomial, ordinal or nominal, set 1.0e-6 as the initial value for all the regression parameters.
- When ordinal multinomial distribution, no matter whether the inputs list falls into the small to medium or large *p* situation, compute the initial value according to Appendix A (Ordinal Multinomial Distribution) in GLE phase 1; When nominal multinomial distribution, no matter

whether the inputs list falls into the small to medium or large p situation, compute the initial value according to Appendix B (Nominal Multinomial Distribution) in GLE phase 1.

8.1 Variable selection or regularization

For the small to median p situations ($p < p^{\max}$), four variable selection or regularization methods are supported: (1) forward stepwise; (2) the lasso (L_1 regularization); (3) elastic net (the $(L_1 + L_2)$ regularization); (4) ridge regression (L_2 regularization). For the large p situations $p \geq p^{\max}$), the lasso, elastic net and ridge regression would be supported, but not forward stepwise. We will utilize ADMM to do the lasso, elastic net and ridge regression and details are provided in GLE phase 3, so only forward stepwise is described in details here.

The basic idea of the forward stepwise method is to start off by choosing the best effect in addition to the intercept if exists and then tries to enter additional effect one at a time. After an effect has been added, all effects in the current model are checked to see if any of them should be removed. The process continues until a stopping criterion is met.

The five candidate statistics will be supported for the effect entry or removal: (1) Likelihood ratio (LR) statistic; (2) Score statistic (SCORE); (3) Wald statistic (WALD); (4) Finite sample corrected Akaike information criteria (AICC); and (5) Average square error (ASE) over the testing data. More specifically, six combinations (Table 8.2) of candidate statistics are available for the effect entry and removal. The default statistics for the effect entry and removal are SCORE and WALD, respectively.

Table 8.2. Six combinations of statistics for effect entry and removal

No	Statistics	
	Effect entry	Effect removal
1	SCORE	WALD
2	SCORE	LR
3	LR	WALD
4	LR	LR
5	AICC	AICC
6	ASE	ASE

It is noted that LR statistic, AICC and ASE might consume considerable computation time since a model is fitted for each effect. Score and Wald statistic use less computation time but may be less accurate in the significance test of the effect of interest.

The details of statistics calculations and the selection process are described below.

Implementation note:

- We only implement #1 so far and will implement other options later if time permits.

8.1.1 Candidate statistics

X_j	A continuous effect.
$\{X_{j_s}\}_{s=1}^{m}$	A categorical effect.
$\boldsymbol{B}_{cur}$	The parameters for the current model
$\boldsymbol{B}_j$	The parameters for the effect j which is continuous or categorical
m^*	The difference in the number of non-redundant parameters estimated of two successive models
ℓ_{cur}	The log likelihood for the current model.
ℓ_{cur+j}	The log likelihood for the resulting model after entering the effect j.
$\ell_{cur\setminus j}$	The log likelihood for the resulting model after removing the effect j.

(1) **LR statistic**

The LR statistic is defined as two times the log of the ratio of the likelihood functions of two models evaluated at their MLEs. The LR statistics for an effect j $\left(X_j \text{ or } \{X_{j_s}\}_{s=1}^{m}\right)$ entering and removing from the current model are calculated as follows:

$$S_{enter_j} = 2(\ell_{cur+j} - \ell_{cur})$$
$$S_{remove_j} = 2(\ell_{cur} - \ell_{cur\setminus j})$$

The asymptotic distribution of the LR statistic, under the hypothesis that the additional or removal parameters in the model are equal to 0, is a chi-square with m^* degrees of freedom, where m^* equal to the difference in the number of non-redundant parameters estimated in two successive models, i.e., $\chi^2_{m^*}$.

Then the *p*-values corresponding to the above LR statistic are

$$p_{enter_j} = 1 - P\left(\chi^2_{m^*} \leq S_{enter_j}\right)$$

$$p_{remove_j} = 1 - P\left(\chi^2_{m^*} \leq S_{remove_j}\right)$$

(2) **AICC**

The AICC values for the resulting model when an effect *j* enters to or is removed from the current model are calculated as follows:

$$AICC_{cur+j} = -2\ell_{cur+j} + \frac{2d_{cur+j} \cdot N}{N - d_{cur+j} - 1}$$

$$AICC_{cur\setminus j} = -2\ell_{cur\setminus j} + \frac{2d_{cur\setminus j} \cdot N}{N - d_{cur\setminus j} - 1}$$

where d (d_{cur+j} or $d_{cur\setminus j}$) denote the degrees of freedom of the resulting model. For all distributions except ordinal and nominal multinomial, $d = p_x$ if only $\boldsymbol{\beta}$ is included; $d = p_x + 1$ if $\boldsymbol{\beta}$ and ϕ for normal, inverse Gaussian, gamma and Tweedie distributions or $\boldsymbol{\beta}$ and k for negative binomial distribution are included. For ordinal and nominal multinomial, *d* is just the number of non-redundant parameters.

(3) **Score statistic**

The score statistic is calculated for each effect not in the model to determine whether the effect should enter the model.

Suppose the current model's maximum likelihood estimate is $\widehat{\boldsymbol{B}}_{cur}$. Using the block notations, the score function ($\boldsymbol{s}$, the gradient vector) and information matrix ($\boldsymbol{I} = -\boldsymbol{H}$, the negative Hessian matrix) of the resulting model (the current model with additional effect *j*) are calculated as

$$\boldsymbol{s}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) = \begin{pmatrix} \boldsymbol{s}_{cur}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) \\ \boldsymbol{s}_j(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) \end{pmatrix}$$

$$\boldsymbol{I}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) = \begin{pmatrix} \boldsymbol{I}_{cur,cur}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) & \boldsymbol{I}_{cur,j}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) \\ \boldsymbol{I}_{j,cur}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) & \boldsymbol{I}_{j,j}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) \end{pmatrix}$$

The inverse information matrix is

$$\boldsymbol{J}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) = \begin{pmatrix} \boldsymbol{J}_{cur,cur}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) & \boldsymbol{J}_{cur,j}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) \\ \boldsymbol{J}_{j,cur}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) & \boldsymbol{J}_{j,j}(\boldsymbol{B}_{cur}, \boldsymbol{B}_j) \end{pmatrix}$$

Then the score statistic for the null hypothesis $H_0\colon \boldsymbol{B}_j = \boldsymbol{0}$ is

$$S_{enter_j} = \boldsymbol{s}_j(\widehat{\boldsymbol{B}}_{cur}, \boldsymbol{0})^T \boldsymbol{J}_{j,j}(\widehat{\boldsymbol{B}}_{cur}, \boldsymbol{0}) \boldsymbol{s}_j(\widehat{\boldsymbol{B}}_{cur}, \boldsymbol{0})$$

Under the null hypothesis, the score statistic has a chi-square distribution with r_s degrees of freedom, where r_s equals to the rank of $\boldsymbol{J}_{j,j}(\widehat{\boldsymbol{B}}_{cur}, \boldsymbol{0})$. If r_s is zero, then the score statistic will be set to 0 and the *p*-value will be 1. Otherwise, the *p*-value is calculated as

$$p_{enter_j} = 1 - P\left(\chi^2_{r_s} \leq S_{enter_j}\right).$$

(4) **Wald statistic**

The Wald statistic is calculated for each effect in the model to determine whether the effect can be removed from the model.

The current model's parameter vector and its estimate can be partitioned into two parts as follows:

$$\boldsymbol{B}_{cur} = \begin{pmatrix} \boldsymbol{B}_{cur\backslash j} \\ \boldsymbol{B}_j \end{pmatrix} \text{ and } \widehat{\boldsymbol{B}}_{cur} = \begin{pmatrix} \widehat{\boldsymbol{B}}_{cur\backslash j} \\ \widehat{\boldsymbol{B}}_j \end{pmatrix}$$

Similarly, the information matrix and its inverse can be partitioned as follow,

$$\boldsymbol{I}(\boldsymbol{B}_{cur}) = \begin{pmatrix} \boldsymbol{I}_{cur\backslash j,cur\backslash j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) & \boldsymbol{I}_{cur\backslash j,j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) \\ \boldsymbol{I}_{j,cur\backslash j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) & \boldsymbol{I}_{j,j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) \end{pmatrix}$$

$$\boldsymbol{J}(\boldsymbol{B}_{cur}) = \begin{pmatrix} \boldsymbol{J}_{cur\backslash j,cur\backslash j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) & \boldsymbol{J}_{cur\backslash j,j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) \\ \boldsymbol{J}_{j,cur\backslash j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) & \boldsymbol{J}_{j,j}(\boldsymbol{B}_{cur\backslash j}, \boldsymbol{B}_j) \end{pmatrix}$$

Then the Wald statistic for the null hypothesis $H_0\colon \boldsymbol{B}_j = \boldsymbol{0}$ is

$$S_{remove_j} = (\widehat{\boldsymbol{B}}_j)^T \left[J_{j,j}(\widehat{\boldsymbol{B}}_{cur\backslash j}, \widehat{\boldsymbol{B}}_j)\right]^{-} \widehat{\boldsymbol{B}}_j$$

Under the null hypothesis, S has a chi-square distribution with r_s degrees of freedom, where r_s equals to the rank of $\boldsymbol{J}_{j,j}(\widehat{\boldsymbol{B}}_{cur\backslash j}, \widehat{\boldsymbol{B}}_j)$. If r_s is zero, then the score statistic will be set to 0 and the p-value will be 1. Otherwise, the p-value is calculated as

$$p_{remove_j} = 1 - P\left(\chi^2_{r_s} \le S_{remove_j}\right).$$

(5) **ASE**

For distributions except ordinal and nominal distributions, the ASE value over the testing data for the resulting model when an effect enters to or is removed from the current model is

$$ASE = \frac{1}{\sum_{i=1}^{n_T} f_i} \sum_{i=1}^{n_T} f_i (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = g^{-1}\left(\boldsymbol{x}_i^T \widehat{\boldsymbol{\beta}} + \boldsymbol{o}_i\right)$ is the predicted value of y_i, and n_T is the number of distinct cases in the testing data.

For ordinal and nominal distributions, the ASE value is calculated as

$$ASE = \frac{1}{\sum_{i=1}^{n_T} f_i} \sum_{i=1}^{n_T} f_i I\left(y_i \ne c(\boldsymbol{x}_i)\right)$$

$$c(\boldsymbol{x}_i) = \arg\max_j \hat{\pi}_{ij}$$

Where $I(\cdot)$ is indicator function, n_T is the number of distinct cases in the testing data, $c(\boldsymbol{x}_i)$ denotes the predicted category for $\boldsymbol{x}_i$, and $\hat{\pi}_{ij}$ denotes the probability for category *j* corresponding to $\boldsymbol{x}_i$. The detailed calculation of $\hat{\pi}_{ij}$ is given in Section 5.1 of “Algorithm: Generalized Linear Engine Phase I”.

Implementation notes:

- If ASE criterion is chosen, the data would be divided into two parts: training data and testing data. For the current phase, the partition would be set to 30% for testing data, and users have no control over this part. However, users can select the seed so the results can be reproduced.
- The ASE value for ordinal and nominal multinomial distribution represents the classification error actually.

8.1.2 The selection process

When distribution, link function and effects are all specified but the user requests the selection of effects, candidate effects are effects specified; when both distribution and link function are specified, but the effects have not been specified, the candidate effects includes all main effects, and some interaction effects obtained in Section 7.

The criteria could be grouped into two categories: (1) LR, SCORE and WALD; (2) AICC and ASE. The former is to select an effect for entry (removal) with minimum (maximum) p-value and continue doing it until the p-values of all candidates for entry (removal) are equal to or greater than (less than) a specified significance level. The latter is to compare the goodness of fit statistics (AICC or ASE) of the resulting model after entering (removing) an effect with that of the current model and selection would be stopped at a local optimal value.

Some definitions are needed for the selection process.

FLAG	A $p^e \times 1$ index vector which records the status of each effect. $FLAG_i = 1$ means the effect *i* is in the current model; $FLAG_i = 0$ means it is not. Note that $\lvert\{i \mid FLAG_i = 1\}\rvert$ denotes the number of effects with $FLAG_i = 1$.
MAXSTEP	The maximum number of iteration steps. The tentative default value is $3 \times p^e$.
MAXEFFECT	The maximum number of effects (excluding intercept if exists). The default value is p^e.
p_{in}	The significance level for effect entry when LR or SCORE is used. The default is 0.05.
p_{out}	The significant level for effect removed when LR or WALD is used. The default is 0.1.
$AICC_{cur}$	The AICC value for the current model.
ASE_{cur}	The ASE value for the current model.

(1) Set $\{FLAG_i\}_{i=1}^{p^e} = 0$ and $iter = 0$. The initial model is $\boldsymbol{\eta} = \boldsymbol{o}$.

If LR is used, compute the log-likelihood value;

If AICC (ASE) is used, compute AICC (ASE) for the initial model and denote it as $AICC_{cur}$ (ASE_{cur}).

(2) If $\{i \mid FLAG_i = 1\} \neq \emptyset$, $iter < MAXSTEP$ and $\lvert\{i \mid FLAG_i = 1\}\rvert < MAXEFFECT$, go to next step (3); otherwise stop and output the model.

(3) Based on the current model, for every effect *j* with $FLAG_i = 0$

If LR or SCORE is used, compute S_{enter_j} and p_{enter_j}.

If AICC (ASE) is used, compute $AICC_j$ (ASE_j).

(4) If LR or SCORE is used, choose the effect X_{j^*}, $j^* = \arg\min_j \left\{p_{enter_j}\right\}$, and enter X_{j^*} to the current model if $p_{enter_{j^*}} < p_{in}$.

If AICC (ASE) is used, choose the effect $X_{j^*}, j^* = \arg\min_j\{AICC_j\}$ (arg $\min_j\{ASE_j\}$), and enter X_{j^*} to the current model if $AICC_{j^*} < AICC_{cur}$ ($ASE_{j^*} < ASE_{cur}$).

Then go to (5); otherwise stop and output the current model.

(5) If the model with new effect is the same as any previous ones, stop and output the current model; otherwise update the current model: set $FLAG_{j^*} = 1$ and $iter = iter + 1$.

If AICC (ASE) is used, let $AICC_{cur} = AICC_{j^*}$ ($ASE_{cur} = ASE_{j^*}$).

(6) For every effect *k* in the current model (i.e., $FLAG_k = 1, \forall k$),

If LR or WALD is used, compute S_{remove_k} and p_{remove_k}.

If AICC (ASE) is used, compute $AICC_k$ (ASE_k).

(7) If LR or WALD is used, choose the effect X_{k^*}, $k^* = \arg\max_k\{p_{remove_k}\}$, and remove X_{k^*} from the current model if $p_{remove_{k^*}} > p_{out}$.

If AICC (ASE) is used, choose the effect X_{k^*}, $k^* = \arg\min_k\{AICC_k\}$ (arg $\min_k\{ASE_k\}$), and remove X_{k^*} from the current model if $AICC_{k^*} < AICC_{cur}$ ($ASE_{k^*} < ASE_{cur}$).

Then go to (8); otherwise go back to (2).

(8) If the model with the effect removed is the same as any previous one, stop and output the current model; otherwise update the current model: set $FLAG_{k^*} = 0$ and $iter = iter + 1$.

If AICC (ASE) is used, let $AICC_{cur} = AICC_{k^*}$ ($ASE_{cur} = ASE_{k^*}$).

Notes:

- The estimate method for ϕ or k should be kept consistent for the model sequence generated by entering or removing the effect. More specifically, when ϕ is estimated by ML method, or the deviance or Pearson chi-square divided by degrees of freedom, estimated $\hat{\phi}$ would be different for a pair of models. That is to say, score and Wald statistics will use $\hat{\phi}$ of the current model rather than of the final model (full model), because ϕ for the full model would be not obtained. For the same reason, $2\left(\ell_{1,\hat{\phi}_1} - \ell_{2,\hat{\phi}_2}\right)$ will be used in the LR statistics. Similar for k.
- Let $\boldsymbol{I} = \begin{pmatrix} \boldsymbol{I}_{11} & \boldsymbol{I}_{12} \\ \boldsymbol{I}_{21} & \boldsymbol{I}_{22} \end{pmatrix}$ denote the information matrix, then its inverse $\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J}_{11} & \boldsymbol{J}_{12} \\ \boldsymbol{J}_{21} & \boldsymbol{J}_{22} \end{pmatrix} = \boldsymbol{I}^{-}$ can be calculated as follows,

$$\begin{aligned} \boldsymbol{J}_{11} &= \boldsymbol{I}_{11}^{-1} + \boldsymbol{I}_{11}^{-1}\boldsymbol{I}_{12}\boldsymbol{J}_{22}\boldsymbol{I}_{21}\boldsymbol{I}_{11}^{-1} \\ \boldsymbol{J}_{12} &= -\boldsymbol{I}_{11}^{-1}\boldsymbol{I}_{12}\boldsymbol{J}_{22} \\ \boldsymbol{J}_{21} &= \boldsymbol{J}_{12}^{T} \\ \boldsymbol{J}_{22} &= [\boldsymbol{I}_{22} - \boldsymbol{I}_{21}\boldsymbol{I}_{11}^{-1}\boldsymbol{I}_{12}]^{-} \end{aligned}$$

- For LR, AICC and ASE, a model is fitted for each effect adding or being removed; For SCORE and WALD, only one model is fitted for the effect which is finally determined to add to or remove from the current model.
- The cold start for the initial model $\boldsymbol{\eta} = \mathbf{0}$ and power link family: $\boldsymbol{\mu} = \mathbf{0}$ when power link (including identity) is used. Thus, the score vector $\boldsymbol{s}$ is missing because $V(\mu_i)$ and $g'(\mu_i)$ are zero. Consequently, the score statistic could not be conducted. To overcome this issue, we first build a intercept-only model, then use score statistics to select the best effect for entering the model for the time based on the intercept-only model.
- For effect entry in step (4), when LR or SCORE is used, if there is a tie in determining the effect $j^*(\min_j\{p_{enter_j}\})$, then select the effect with the smallest degrees of freedom. If effects still have the same degrees of freedom, then select the one with ordering earlier in the effect list.

  For effect entry in step (4), when AICC or ASE is used, if there is a tie in determining optimal value, then select the one with the smallest degrees of freedom. If effects still have the same degrees of freedom, then select the one with ordering earlier in the effect list.
- Similarly, for effect removal in step (7), when LR or WALD is used, if there is a tie in determining optimal value $(\max_k\{p_{remove_k}\})$, then select the effect with the largest degrees of freedom. If effects still have the same degrees of freedom, then select the one with ordering later in the effect list.

  For effect removal in step (7), when AICC or ASE is used, if there is a tie in determining optimal value, then select the one with the largest degrees of freedom. If effects still have the same degrees of freedom, then select the one with ordering later in the effect list.
- Regarding rules for entering for removing effects when interaction effects are presented, please refer to Chu and Han (2011).

8.2 Distribution and link function selection

The distribution and link function selection is to select an appropriate distribution and/or link for the given data when distribution and/or link function are not specified by user. Table 8.3 gives candidate combinations of distribution and link function. Since there are too many link functions potentially, it will be too time consuming to check every combination of distribution and link function. Thus we will only consider the combinations listed on Table 8.3 and Table 8.4 according to the target's measurement level and the storage type.

Please note that if the distribution (link function) is specified, then the distribution (link function) will not be detected any more.

Table 8.3: List of combinations of distribution and link function

Candidate distribution	Candidate link functions
Normal	Identity, log, power(0.5)
Inverse Gaussian	Identity, log, power(-2)
Gamma	Identity, log, power(-1)
Tweedie(q)	Identity, log, power$(1-q)$
Negative binomial(k)	Identity, log, Negative binomial
Poisson	Identity, log, power(0.5)
Binomial	Logit, Probit, complementary log-log
Nominal multinomial	Generalized logit
Ordinal multinomial	Cumulative logit, cumulative probit, cumulative complimentary log-log

Table 8.4 List of candidate distributions based on measurement level and storage type of the target

Measurement level of the target		Storage type of the target	Candidate distribution
Continuous		Positive real	Normal
			Inverse Gaussian
			Gamma
			Tweedie(q)
		Positive real with zeros	Normal
			Tweedie(q)
		Real or integer with negative values	Normal
		Positive integer with zeros	Negative binomial(k)
			Poisson
			Normal
		Positive integer	Negative binomial(k)
			Poisson
			Normal
			Inverse Gaussian
			Gamma
Nominal	#categories = 2		Binomial
	#categories > 2		Nominal multinomial
Ordinal	#categories = 2		Binomial
	#categories > 2		Ordinal multinomial

Flag		Binomial

Notes:

- For the case that the distribution is given, the candidate link functions are determined using Table 8.3 after checking the given distribution is compatible with the measurement level and storage type of the target in Table 8.4.
- For the case that the link function is given, the following rules are used to determine the candidate distributions

  (1) If the link function is one of cumulative logit, cumulative probit, cumulative complimentary log-log, cumulative negative log-log and cumulative cauchit, then the candidate distribution is ordinal multinomial after checking the measurement level is ordinal. Otherwise, an error message should be issued.

  (2) If the link function is generalized logit, the candidate distribution is nominal multinomial after checking the measurement level is nominal. Otherwise, an error message should be issued.

  (3) If the link function is one of identity, log, and power, the candidate distributions are determined by Table 8.4.

  (4) If the link function is negative binomial, the candidate distribution is negative binomial.

  (5) If other link functions are used, the candidate distribution is binomial.

8.2.1 Candidate statistics

8.2.1.1 ASE

The definition is the same to that in Section 8.1.2.

8.2.2 The selection process

A model is run for each combination of distribution and link function, given a set of effects, depending the measurement level and the storage type of the target given in Table 8.3 and Table 8.4 on the training data, then the best model with minimum value of ASE on the testing data is selected. Thus, the corresponding distribution and link function are selected and output.

Implementation note:

- For the small to median p situation($p \ < p^{\max}$), there are two ways the model is built: (1) using the Newton-Raphson with MapReduce, see GLE phase 1 for details; (2) using ADMM with the Newton-Raphson method, see GLE phase 3 for details.
- For the large p situation ($p \ \geq p^{\max}$), the model is built using ADMM with L-BFGS method, see GLE phase 3 for details.

8.3 Automatic detection of distribution, link function and effects

When the effects in addition to at least one of distribution and link function have not been specified, we will apply different methods, depending on the variable selection methods, to detect distribution, link function and effects automatically.

8.3.1 The variable selection method is forward stepwise:

The two-stage model selection method is applied and its basic idea is to apply distribution and link function selection and variable selection alternately. It starts off by getting an initial estimate of the distribution $\hat{F}$ and the link function $\hat{g}$ based on a given effect set $\boldsymbol{X}$; then estimate the optimal effects $\widehat{\boldsymbol{X}}$ based on estimated $\hat{F}$ and $\hat{g}$. These two steps are performed alternatively until the convergence criterion is satisfied. In addition, an enhancement stage of adding two-way interaction effects is provided when the size of the optimal variable set is less than a predefined threshold.

$\mathbb{C} = \{C_i\}$	The set of potential combination of distribution and link function based on the type of the target, where $C_i = (F_i, g_i)$, and F_i and g_i denotes the distribution and the link function, respectively.
$\boldsymbol{X}^l$	The effects for the l-th iteration, including both main and interaction effects
e^l	The ASE value for the l-th iteration.
ε_m	The tolerance level for the convergence criteria. Its default value is set to 10^{-4}.
$l_{\max}$	The maximum number of iterations. Its default value is set to 2.
Λ_1, Λ_2	The significant and non-significant predictors for the optimal combination of distribution and link function, respectively.
E_1, E_2	The candidate set of effects for variable selection.
m_3	Threshold value to provide an enhancement stage; the default is 100.
m_4	Threshold value to select main effects for constructing interaction effects; the default is 20.

The detailed process is given below,

(1) Determine candidate combinations $\mathbb{C} = \{C_i\}, i = 1, \cdots, n_{fg}$ of distributions and link functions based on the target's measure level and storage type based on Table 8.3 and 8.4.

(2) Let $\boldsymbol{X}^0$ be the initial effect set, which only includes all main effects (predictors). Let $\left(\widehat{\boldsymbol{X}}^{\text{best}}, \hat{C}^{\text{best}}, \hat{e}^{\text{best}}\right)$ denote the optimal model.

(3) Select 3 best combinations, denoted by $\mathbb{C}'$, from candidate combination $\mathbb{C}$.

(a) Build one model for each combination in $\mathbb{C}$ based on $\boldsymbol{X}^0$ using the training data.

(b) Calculate ASE value for each model built using the testing data.

(c) Select top 3 models with minimum ASE.

(d) Denote the optimal combination by $\hat{C}^1$ and ASE value by $\hat{e}^{1/2}$. Let $\widehat{\boldsymbol{X}}^{\text{best}} = \boldsymbol{X}^0, \hat{C}^{\text{best}} = \hat{C}^1$, and $\hat{e}^{\text{best}} = \hat{e}^{1/2}$.

(4) Perform Type 3 analysis (using Wald statistics in Section 8.1.1) for the current optimal model. Based on p values of effects, divide effects (predictors) into two groups: one (Λ_1) is for significant predictors ($p < 0.05$), and the other (Λ_2) is for the non-significant predictors.

(5) Obtain the optimal effects $\widehat{\boldsymbol{X}}^1$ based on $\hat{C}^1$, from the candidate effects set of $E_1(= \Lambda_1)$ and $E_2(= \Lambda_2)$, where E_1 is considered as the initial model, and forward stepwise is used to select among E_2.

(6) Calculate the ASE value, $\hat{e}^1$, of the model $\left(\widehat{\boldsymbol{X}}^1, \hat{C}^1\right)$ on the testing data.

(7) If $\left(\hat{e}^1 < \hat{e}^{\text{best}}\right)$, then $\left(\widehat{\boldsymbol{X}}^{\text{best}} = \widehat{\boldsymbol{X}}^1, \hat{C}^{\text{best}} = \hat{C}^1, \hat{e}^{\text{best}} = \hat{e}^1\right)$.

(8) Let $l = 1$. If $l > (l_{\max} - 1)$, then stop and output the optimal model $\left(\widehat{\boldsymbol{X}}^{\text{best}}, \hat{C}^{\text{best}}, \hat{e}^{\text{best}}\right)$.

(9) Build one model for each combination in $\mathbb{C}'$ based on $\widehat{\boldsymbol{X}}^l$ using the training data, calculate ASE for each model using the testing data.

(10) Obtain the optimal combination (denoted by $\hat{C}^{l+1}$) with minimum ASE value (denoted by $\hat{e}^{l+1/2}$).

(11) If $\hat{e}^{l+1/2} < \hat{e}^{\text{best}}$, then $\widehat{\boldsymbol{X}}^{\text{best}} = \widehat{\boldsymbol{X}}^l, \hat{C}^{\text{best}} = \hat{C}^{l+1}$, and $\hat{e}^{\text{best}} = \hat{e}^{l+1/2}$.

(12) Obtain the optimal effects $\widehat{\boldsymbol{X}}^{l+1}$ based on $\hat{C}^{l+1}$, from the candidate effects set of $E_1 = \widehat{\boldsymbol{X}}^l$ and $E_2 = \boldsymbol{X}^0 - E_1$, where E_1 is considered as the initial model, and forward stepwise is used to select among E_2.

(13) Calculate the ASE value, $\hat{e}^{l+1}$, of the model $\left(\widehat{\boldsymbol{X}}^{l+1}, \hat{C}^{l+1}\right)$ on the testing data.

(14) If $\left(\hat{e}^{l+1} < \hat{e}^{\text{best}}\right)$, then $\left(\widehat{\boldsymbol{X}}^{\text{best}} = \widehat{\boldsymbol{X}}^{l+1}, \hat{C}^{\text{best}} = \hat{C}^{l+1}, \hat{e}^{\text{best}} = \hat{e}^{l+1}\right)$.

(15) If $\left[\left(\widehat{\boldsymbol{X}}^{l+1}, \hat{C}^{l+1}\right)=\left(\widehat{\boldsymbol{X}}^{l}, \hat{C}^{l}\right)\right]$ or $\left[l \geq\left(l_{\max }-1\right)\right]$ or $\left[\frac{\left|\hat{e}_{l+1}-\hat{e}_{l}\right|}{\hat{e}_{l}+10^{-6}}<\varepsilon_{m}\right]$, then stop and output the optimal model $\left(\widehat{\boldsymbol{X}}^{\text{best}}, \hat{C}^{\text{best}}, \hat{e}^{\text{best}}\right)$; otherwise, $l=l+1$ and go to step (9).

If $\left|\widehat{\boldsymbol{X}}^{\text{best}}\right|<m_{3}$, an enhancement stage of adding two-way interaction effects will be provided as follows:

(1) Perform Type 3 analysis (using Wald statistics in Section 8.1.1) for the current optimal model.

(2) Sorting the main effects using p-value in ascending order and select top $m'=\min\left(\left|\widehat{\boldsymbol{X}}^{\text{best}}\right|, m_{4}\right)$ main effects.

(3) Construct of two-way interaction effects (of any two different main effects, and squared term of covariates) among the m' main effects.

(4) Test all candidate interaction effects (using Score statistics in Section 8.1.1) based on the current optimal model.

(5) Select the significant interaction effects $(p<0.05)$, and sort them using their p-values in descending order and select and output top-k interaction effects (Denoted by $\widehat{\boldsymbol{X}}^{\text{inter}}$), where k is the maximum number satisfying the number of parameters for top-k interaction effects is less than or equal to $0.5 \times\left(p^{\max }-p^{m s}\right)$, where $p^{m s}$ denotes the number of parameters for $\widehat{\boldsymbol{X}}^{\text{best}}$ (including the intercept).

(6) Obtain the optimal effects $\widehat{\boldsymbol{X}}^{l+2}$ based on $\hat{C}^{\text{best}}$, $E_{1}=\widehat{\boldsymbol{X}}^{\text{best}}$ and $E_{2}=\widehat{\boldsymbol{X}}^{\text{inter}}$ using the method of variable selection given in Section 8.1, where E_{1} is considered as the initial model, and stepwise is used to select among E_{2}.

(7) Calculate the ASE value, $\hat{e}^{l+2}$, of the model $\left(\widehat{\boldsymbol{X}}^{l+2}, \hat{C}^{\text{best}}\right)$ on the testing data.

(8) If $\left(\hat{e}^{l+2}<\hat{e}^{\text{best}}\right)$, then $\left(\widehat{\boldsymbol{X}}^{\text{best}}=\widehat{\boldsymbol{X}}^{l+2}, \hat{C}^{\text{best}}=\hat{C}^{\text{best}}, \hat{e}^{\text{best}}=\hat{e}^{l+2}\right)$.

(9) Stop and output the optimal model $\left(\widehat{\boldsymbol{X}}^{\text{best}}, \hat{C}^{\text{best}}, \hat{e}^{\text{best}}\right)$.

Note that parameter settings of model selection are given below:

- For the scale parameter ϕ, it is estimated by MLE for normal, inverse Gaussian, gamma and Tweedie distribution; it is fixed at 1.0 for binomial, Poisson, negative binomial, and multinomial.

- If both distribution and link function are not specified, then,

  (1) For the ancillary parameter (k) in negative binomial distribution, it is estimated by MLE;

  (2) For the parameter q in the tweedie distribution, it is set to 1.5, namely, $q=1.5$;

  (3) The parameter in power function is given from Table 8.3.

- If only the distribution is given,

  (1) For k, it equals to the parameter specified by user.

  (2) For q, it equals to the parameter specified by user.

- If only the link function is given

  (1) For the parameter in power function, it equals to the parameter specified by user.

  (2) For k, it is estimated by MLE.

  (3) For q, it is set to 1.5.

- For the variable selection, the statistics for effect entry and removal are SCORE and WALD, respectively.

8.3.2 The variable selection method or regularization is the lasso, elastic net or ridge regression:

(1) Choose the candidate combinations of distributions and link functions based on the measurement level and storage type of target.

(2) Detect the interaction terms if interaction detection flag is on based on a combination chosen with the following rules, and then form the set of candidate effects.

When distribution and link function both are unknown, the rules are as follows:

(a) If normal is a possible distribution candidate, then normal + identify will be chosen.

(b) If binomial is a possible distribution candidate, then binomial + logit will be chosen.

(c) If measurement level is nominal, distribution is nominal multinomial, then nominal multinomial + generalized logit will be chosen.

(d) If measurement level is ordinal, distribution is ordinal multinomial (even nominal multinomial is a candidate), then ordinal multinomial + cumulative logit will be chosen.

When distribution is known, link function is unknown, the rules are listed in the follow table:

Distribution	Link function
Normal	Identity
Inverse Gaussian	Power(–2)
Gamma	Power(–1)
Tweedie (q)	Power(1 – q)

Negative binomial (*k*)	Log
Poisson	Log
Binomial	Logit
Nominal multinomial	Generalized logit
Ordinal multinomial	Cumulative logit

When link function is known, distribution is unknown, the rules are as below.

(a) If link function is identify, log or power, then normal will be chosen.

(b) If link function is negative binomial, then negative binomial will be chosen.

(c) If link function is generalized logit, then nominal multinomial will be chosen.

(d) If link function is one of 5 cumulative link functions (logit, probit, complimentary log-log, negative log-log, cauchit), then ordinal multinomial will be chosen.

(e) If link function is logit, probit, complementary log-log, log-complement, negative log-log or odds power, then binomial will be chosen.

(3) Select lambda from the grid search method described in GLE phase 3 with the chosen combination and the set of candidate effects.

(4) Run ADMM (the lasso, ridge or elastic net) to select effects for each combination based on the selected lambda and the training data, then compute ASE in the testing set.

(5) Choose the combination with the selected effects with the minimum ASE.

(6) If the chosen combination of distribution and link function is not the same as the one in step (2), then another round of lambda selection based on the chosen combination is conducted to update the effect selection.

Implementation notes:

- When running ADMMs in step (3) for each combination, we should use the results from step (2) as the initial values.
- The more complete process should be to select the lambda for each combination, instead of finding a lambda based on a particular combination and applying the same lambda for other combinations, in Step (2). Since it might be too time consuming, we propose the above process. However, the grid search method with the warm-start strategy might not take much longer than the one with a fixed lambda. Thus we should implement both processes and do some testing to compare the performance.

- If using user-specified lambda, step 3 and step 6 will be ignored.

8.4 Handle large volume of data

The model selection will be time-consuming when there is a large volume of data, because the model building might involve many data passes. To speed up the process of model selection, the sampling techniques are employed, which sample a small dataset from the whole data. In addition, from a practical viewpoint, it is not necessary to use all data for selecting an approximately best model.

Two sampling techniques are needed: (1) simple random sampling; (2) stratified random sampling. The former is used for all distributions except binomial distribution with 0/1 format and multinomial distribution (ordinal and nominal); the latter is used when the distribution is binomial distribution with 0/1 format and multinomial distribution.

8.4.1 Simple Random Sampling

A subset of records is chosen from the larger set. Each record is chosen randomly such that each record has the same probability of being chosen at any stage during the sampling process, and each subset of k records has the same probability of being chosen for the sample as any other subset of k records.

The sampling will be triggered when $N > N_T$ (=20,000 by default). The default sample size is $N_S = 10{,}000$. Please note that both exact and approximate simple random sampling can be used.

The details of simple random sampling methods see Dagli (2012).

8.4.2 Stratified Random Sampling

Stratified random sampling is a probability sampling technique wherein the entire population is divided into different subgroups or strata, then randomly selects the final samples proportionally from the different strata.

Some definitions are needed for the stratified random sampling.

$N_{\cdot j}$	The number of records for the j-th target category, $j = 1, \cdots, J$, in the whole data

N	The total number of records in the whole data, $N = \sum_{j=1}^{J} N_{\cdot j}$
$N_{S,j}$	The sample size for the j-th target category, $j = 1, \cdots, J$
N_S	The total sample size, $N_S = \sum_{j=1}^{J} N_{S,j}$.
$p_{S,j}$	The sampling rate for the j-th target category, $j = 1, \cdots, J$

The sample size for the j-th target category is determined by the following equation

$$N_{S,j} = \lceil p_{S,j} N_{\cdot j} \rceil$$

The sampling will be triggered when $N > N_T$ (=20,000 by default). The default sample size is $N_S = 20{,}000$, and default values for sampling rates are $p_{S,1} = \cdots p_{S,J} = \frac{N_S}{N}$.

Note that (1) the sampling rate $p_{S,j}$ should ensure $N_{S,j} \geq 1$; (2) the sampling rate $p_{S,j}$ may be different for handling the imbalanced data.

The details of stratified random sampling methods can be seen in Dagli (2013). From the viewpoint of generalized linear engine, our requirements for stratified random sampling method are: given the parameters $N_{S,j}$ $(\geq 1), j = 1, \cdots, J$, it returns a subset of data which contains $N'_{S,j}(\geq 1)$ records drew randomly for the j-th target category. Note that generally, $N'_{S,j} = N_{S,j}$ for small $N_{S,j}$, and $N'_{S,j} \approx N_{S,j}$ for large $N_{S,j}$, because it is easier to ensure the sample method will not return empty set of records for large $N_{S,j}$.

It is noted that if $N'_{S,j} = N_{S,j}$, then the sampling method is exact, otherwise, it is approximate.

9. Scoring

9.1 Prediction for binomial distribution with 0/1 binary response variable

9.1.1 Predicted category

Given the critical probability p_t ($p_t = 0.5$ by default), the predicted category $c(\boldsymbol{x}_i)$ is

$$c(\boldsymbol{x}_i) = \begin{cases} 1 \text{ (or success)} & \text{if } \mu_i \geq p_t \\ 0 \text{ (or failure)} & \text{otherwise} \end{cases}.$$

If there is a tie in determining $c(\boldsymbol{x}_i)$, then tie will be broken by choosing the category with

3) Higher $N_j = \sum_{i=1}^{n} f_i y_{i,j}$.
4) If it ties in 1), choose the one with lower category index number.

It should be noted that the classification table should be updated accordingly as well.

9.1.2 Critical probability selection

We select the optimal critical probability based on the following two measures: (1) G-mean and (2) F-measure, which are defined on sensitivity, specificity, and precision measures. Using the notation in Table 9.1, we give their definitions.

Table 9.1 Classification table

		Predicted class	
		success (positive)	failure (negative)
Actual class	success (positive)	$TP = \sum_{i=1}^{n} f_i I(y_i = 1, c(\boldsymbol{x}_i) = 1)$	$FN = \sum_{i=1}^{n} f_i I(y_i = 1, c(\boldsymbol{x}_i) = 0)$
	failure (negative)	$FP = \sum_{i=1}^{n} f_i I(y_i = 0, c(\boldsymbol{x}_i) = 1)$	$TN = \sum_{i=1}^{n} f_i I(y_i = 0, c(\boldsymbol{x}_i) = 0)$

where $I(\cdot)$ is indicator function.

Sensitivity and specificity denotes are two measures of the classification performance. Sensitivity (also called the recall) measures the proportion of actual positives which are correctly identified, which is defined as

$$\rho_1 = \frac{TP}{TP + FN}$$

Specificity measures the proportion of negatives which are correctly identified, which is defined as

$$\rho_2 = \frac{TN}{FP + TN}$$

Precision (also called positive predictive value) measures the ratio of true positives to combined true and false positives, which is defined as

$$\rho_3 = \frac{TP}{TP + FP}$$

The G-mean and F-measure are defined as

$$\begin{aligned} G\text{– mean} \quad &= \sqrt{\rho_1 \times \rho_2} \\ F\text{– measure} \quad &= \frac{\rho_1 \times \rho_3}{\rho_1 + \rho_3} \end{aligned}$$

The process to select a critical probability that maximize the G-mean or F-measure is given below

1) Using equal width method to define $n_c + 1$ ($n_c = 400$ by default) critical probabilities, $p_{t,0}, \cdots, p_{t,n_c}$, between the range of [0,1], where $p_{t,i} = i/n_c$.
2) For each critical probability $p_{t,i}$, calculate the corresponding TP, FN, FP and TN, and then calculate sensitivity and specificity measures if G-mean is used for each critical probability, thus, we obtain sensitivity and specificity vectors, respectively

$$\begin{aligned} \boldsymbol{\rho}_1 &= [\rho_{1,0}, \cdots, \rho_{1,n_c}] \\ \boldsymbol{\rho}_2 &= [\rho_{2,0}, \cdots, \rho_{2,n_c}] \end{aligned}$$

   If F-measure is used, sensitivity and precision measures are computed for each critical probability, thus, we obtain sensitivity and precision vectors, respectively

$$\begin{aligned} \boldsymbol{\rho}_1 &= [\rho_{1,0}, \cdots, \rho_{1,n_c}] \\ \boldsymbol{\rho}_3 &= [\rho_{3,0}, \cdots, \rho_{3,n_c}] \end{aligned}$$

3) Compute the G-mean or F-measure vectors for each critical probability, and find the critical probability with maximum G-mean or F-measure

$$p_t^* = \arg\max_{p_{t,i}} \left\{ \sqrt{\rho_{1,i} \times \rho_{2,i}} \right\}$$

   or

$$p_t^* = \arg\max_{p_{t,i}} \left\{ \frac{\rho_{1,i} \times \rho_{3,i}}{\rho_{1,i} + \rho_{3,i}} \right\}$$

Implementation notes

- Only one data pass is need for calculating $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2$ or $\boldsymbol{\rho}_1, \boldsymbol{\rho}_3$ if we store a tetrad (TP, FN, FP, TN) for each critical probability.

9.2 ROC curve for binomial distribution with 0/1 binary response variable

A ROC curve is a graphical plot which illustrates the performance of a binary classifier as its critical probability is varied. It is created by plotting true positive rate (sensitivity, ρ_1) by false positive rate (1-specificty, $\varphi = 1 - \rho_2$) at various critical probability settings.

The process to obtain the information for ROC curve is given below

1) Using equal width method to define $n_c + 1$ ($n_c = 400$ by default) critical probabilities, $p_{t,0}, \cdots, p_{t,n_c}$, between the range of [0,1], where $p_{t,i} = i/n_c$.
2) For each critical probability $p_{t,i}$, calculate the corresponding TP, FN, FP and TN, then compute true positive rate and false positive rate for each critical probability, thus, we obtain a vector of triads, namely
$$\left[\begin{pmatrix} \rho_{1,0} \\ \varphi_0 \\ p_{t,0} \end{pmatrix}, \cdots, \begin{pmatrix} \rho_{1,n_c} \\ \varphi_{n_c} \\ p_{t,n_c} \end{pmatrix} \right]$$
where $\varphi_i = 1 - \rho_{2,i}$.
3) Remove the redundancy by deleting $\left(\rho_{1,i}, \varphi_i, p_{t,i}\right)^T$ if $\rho_{1,i} = \rho_{1,i-1}$ and $\varphi_i = \varphi_{i-1}$, $i = 1, \cdots, n_c - 1$.
4) Save $\left(\rho_{1,i}, \varphi_i, p_{t,i}\right)^T$,where $\left(\rho_{1,i}, \varphi_i\right)$ is used to plot ROC curve and $p_{t,i}$ is the corresponding critical probability that might be shown in the plot, i.e., using a tooltip.

Implementation notes

- Only one data pass is need for calculating $\boldsymbol{\rho}_1, \boldsymbol{\varphi}$ if we store a tetrad (TP, FN, FP, TN) for each critical probability.

10.Model diagnostics

10.1 Influential outlier

We will identity a record to be an influential outlier based on the following two statistics for all distribution except multinomial:

(1) Cook's distance is larger than $4/(N-d)$, where $d=p_x$ if only $\boldsymbol{\beta}$ is included; $d=p_x+1$ if $\boldsymbol{\beta}$ and ϕ for normal, inverse Gaussian, gamma and Tweedie distributions or $\boldsymbol{\beta}$ and k for negative binomial distribution are included.

(2) The absolute of standardized deviance residual is larger than 2 (or 2.5).

The definitions of Cook's distance and standardized deviance residual are given in Section 5.2 of Generalized Linear Engine Phase I (Chu and Zhong, 2012)

10.2 Diagnostic plots

A scatter plot is provided for all distributions except ordinal and nominal multinomial distributions, which is used to check whether the fitted regression model adequately represents the data.

10.2.1 Scatter plot of standardized deviance residual by predicted linear predictor

The expected pattern of this plot is that a distribution of standardized deviance residuals for varying the linear predictors with mean 0 and constant range.

Let $\hat{r}_k^{SD}$and $\hat{\eta}_k$ be the standardized deviance residual and the predicted linear predictor of the k-th record, respectively, where $k=1,\cdots,n$. Note that because i has been used below, here we use k as a subscript. Then the information needed for a binned scatter plot of standardized deviance residual by the predicted linear predictor is created as follows:

1) Using equal width method to compute n_c (=19 by default) cut points $c_1^{(1)},\cdots,c_{n_c}^{(1)}$ between the range $[L,U]$ for the x-axis, where $L=g\left(\min_k(y_k)\right)$ and $U=g\left(\max_k(y_k)\right)$, i.e., $c_i^{(1)}=L+i\times(U-L)/(n_c+1)$. Then we have (n_c+1) intervals by letting $c_0^{(1)}=-\infty$ and $c_{n_c+1}^{(1)}=\infty$,

$$\left(c_0^{(1)},c_1^{(1)}\right],\left(c_1^{(1)},c_2^{(1)}\right],\cdots,\left(c_{n_c}^{(1)},c_{n_c+1}^{(1)}\right].$$

2) Similarly, compute n_c cut points $c_1^{(2)}, \cdots, c_{n_c}^{(2)}$ between the range $[-8,8]$ for the y-axis: $c_i^{(2)} = -8 + i \times 16/(n_c+1)$. Then we have another (n_c+1) intervals by letting $c_0^{(2)} = -\infty$ and $c_{n_c+1}^{(2)} = \infty$,

$$\left(c_0^{(2)}, c_1^{(2)}\right], \left(c_1^{(2)}, c_2^{(2)}\right], \cdots, \left(c_{n_c}^{(2)}, c_{n_c+1}^{(2)}\right].$$

3) For each two-dimension interval $\left(c_i^{(1)}, c_{i+1}^{(1)}\right] \times \left(c_j^{(2)}, c_{j+1}^{(2)}\right], i,j = 0, \cdots, n_c$, obtain the number of records that fall into this interval incorporating the frequency weight:

$$n_{ij} = \sum_{k=1}^{n} f_k I_{ij}(\hat{\eta}_k, \hat{r}_k^{SD})$$

and the corresponding mean $\left(\bar{\hat{\eta}}_{ij}, \bar{\hat{r}}_{ij}^{SD}\right)$ incorporating the frequency weight:

$$\bar{\hat{\eta}}_{ij} = \frac{1}{n_{ij}} \sum_{k=1}^{n} f_k I_{ij}(\hat{\eta}_k, \hat{r}_k^{SD})\, \hat{\eta}_k$$

$$\bar{\hat{r}}_{ij}^{SD} = \frac{1}{n_{ij}} \sum_{k=1}^{n} f_k I_{ij}(\hat{\eta}_k, \hat{r}_k^{SD})\, \hat{r}_k^{SD}$$

where

$$I_{ij}(\hat{\eta}_k, \hat{r}_k^{SD}) = \begin{cases} 1, & \text{if } \hat{\eta}_k \in \left(c_i^{(1)}, c_{i+1}^{(1)}\right] \text{ and } \hat{r}_k^{SD} \in \left(c_j^{(2)}, c_{j+1}^{(2)}\right] \\ 0, & \text{otherwise} \end{cases}$$

4) Save the mean, $\left(\bar{\hat{\eta}}_{ij}, \bar{\hat{r}}_{ij}^{SD}\right)$ and the corresponding number of records, n_{ij} $(i,j = 0, \cdots, n_c)$ for the scatter plot of standardized deviance residual by predicted linear predictor. Note that if $n_{ij} = 0$, there is no need to save it and the corresponding $\left(\bar{\hat{\eta}}_{ij}, \bar{\hat{r}}_{ij}^{SD}\right)$.

Implementation notes

- If $n \le n_{\text{plot}}(= 3(n_c+1) = 60$ by deafult), then the data will not be binned. The data point $(\hat{\eta}_k, \hat{r}_k^{SD})$ and the corresponding number of records, n_{ij}, will be used for scatter plot directly.
- In addition, we consider a special case: All effects contain only factors and the number of combinations of all factors (n_{cf}) in the model is less than n_{cf}^{max}(=100 by default), namely, $n_{cf} < n_{cf}^{max}$. Let $m = n_{cf}$.
  (a) Compute the linear predictors $\hat{\eta}_k, k = 1, \cdots, m$.
  (b) Sorting $\hat{\eta}_k$ by an ascending order.
  (c) Divide $\hat{\eta}_k$ into m intervals as follows

  $$\left(-\infty, \frac{\hat{\eta}_1 + \hat{\eta}_2}{2}\right], \left(\frac{\hat{\eta}_1 + \hat{\eta}_2}{2}, \frac{\hat{\eta}_2 + \hat{\eta}_3}{2}\right], \cdots, \left(\frac{\hat{\eta}_{m-1} + \hat{\eta}_m}{2}, +\infty\right)$$

  (d) The intervals for $\hat{r}_k^{SD}$ are the same to those given above, in addition, we use the same method to compute $\left(\bar{\hat{\eta}}_{ij}, \bar{\hat{r}}_{ij}^{SD}\right)$. Note that $\bar{\hat{\eta}}_{ij} = \hat{\eta}_i$ for any j.

- n_{ij}, $\bar{\hat{\eta}}_{ij}$ and $\bar{\hat{r}}_{ij}^{SD}$ can be computed in parallel in the map-reduce environment.
- For binomial distribution with r/m format, the standardized deviance residuals can be the one based on proportion or the one based on the number of events, because they are the same.

10.3 Trend analysis from diagnostics plots

The plot in Section 10.2.1 provides the informal checks on whether a fitted regression model adequately represents the data. It still needs the experienced analyst to make such a decision. Here we provide a trend analysis to give a formal check which can automatically determine whether a fitted model is adequate.

By analyzing the trend of the plot, the expected pattern can be a horizontal line through 0.

The process of trend analysis contains three steps: (1) calculate the data points representing the trend; (2) remove outliers in the trend data; (3) fit a simple linear model on given trend data points; (4) test whether the simple linear model adequately represents the trend data.

Calculate the trend data

We consider the following three cases:

Denote $(x_{MED,i}, y_{MED,i}, n_{MED,i}), i = 0, \cdots, m$ by the trend data.

- when $n > n_{\text{plot}}$, we denote $(x_{ij}, y_{ij}, n_{ij}), i, j \in \{0, \cdots, n_c\}$ by the binned data obtained from Section 5.2.1, where $x_{ij} = \bar{\hat{\eta}}_{ij}$, and $y_{ij} = \bar{\hat{r}}_{ij}^{SD}$. Then, we have

$$\begin{cases} x_{MED,i} = \text{median}(\underbrace{x_{i0}, \cdots, x_{i0}}_{n_{i0}}, \cdots, \underbrace{x_{in_c}, \cdots, x_{in_c}}_{n_{in_c}}) \\ y_{MED,i} = \text{median}(\underbrace{y_{i0}, \cdots, y_{i0}}_{n_{i0}}, \cdots, \underbrace{y_{in_c}, \cdots, y_{in_c}}_{n_{in_c}}) \\ n_{MED,i} = \sum_{j=0}^{n_c} n_{ij} \end{cases}$$

  and $m = n_c$.
- when $n \le n_{\text{plot}}$, we denote $(x_i, y_i, n_i), i = 0, \cdots, n-1$ by the non-aggregated data obtained from Section 5.2.1, we have

$$\begin{cases} x_{MED,i} = x_i \\ y_{MED,i} = y_i \\ n_{MED,i} = n_i \end{cases}$$

  and $m = n$.
- For the special case: all effects contain only factors and the number of combinations of all factors (n_{cf}) in the model is less than n_{cf}^{max}, namely, $n_{cf} < n_{cf}^{max}$. We denote $(x_{ij}, y_{ij}, n_{ij}), i =$

$1, \cdots, m, j = 0, \cdots, n_c$ by the aggregated data obtained from Section 5.2.1, where $x_{ij} = \bar{\hat{\eta}}_{ij}$, and $y_{ij} = \bar{\hat{r}}_{ij}^{SD}$.

$$\begin{cases} x_{MED,i} = x_{i1} \\ y_{MED,i} = \text{median}(\underbrace{y_{i0}, \cdots, y_{i0}}_{n_{i0}}, \cdots, \underbrace{y_{in_c}, \cdots, y_{in_c}}_{n_{in_c}}) \\ n_{MED,i} = \sum_{j=0}^{n_c} n_{ij} \end{cases}$$

Notes:

- For binomial distribution with 0/1 binary response, weighted mean function is used to replace median function for calculating $y_{MED,i}$ and (or) $x_{MED,i}$.

Remove the outliers

Without loss of generality, we assume that the trend data records are $(x_i, y_i, n_i), i = 0, \cdots, m$. Here, we use modified z score to remove the outliers.

(1) Calculate the median (MED) and the median absolute deviation (MAD) for $y_{MED,i}, i = 0, \cdots, m$

$$MED = \text{median}(\underbrace{y_0, \cdots, y_0}_{n_0}, \cdots, \underbrace{y_m, \cdots, y_m}_{n_m})$$

$$MAD = \text{median}(|y_0 - MED|, \cdots, |y_0 - MED|, \cdots, |y_m - MED|, \cdots, |y_m - MED|)$$

(2) Compute the modified z-score for $y_i, i \in \{minIndex,\ maxIndex\}$, where $minIndex$ and $maxIndex$ are the index of minimum and maximum value of $\{y_0, \cdots, y_m\}$, respectively.

$$z_i = \begin{cases} \dfrac{y_i - MED}{1.4826 \times MAD} & \text{if } MAD \neq 0 \\ \dfrac{y_i - MED}{1.2533 \times MeanAD} & \text{if } MAD = 0 \end{cases}$$

where $MeanAD = \frac{1}{\sum_{i=0}^{m} n_{i\cdot}} \sum_{i=1}^{m} n_i |y_i - MED|$.

(3) (x_i, y_i, n_i) is removed from the trend data if $|z_i| > 3$ for $i \in \{minIndex,\ maxIndex\}$.

Fit a simple linear model

For the trend data $(x_i, y_i, n_i), i = 0, \cdots, m$, we fit a simple linear model ($y = b_0 + b_1 x$) incorporating the frequency weight.

Let

$$\bar{x} = \frac{1}{\sum_{i=0}^{m} n_i} \sum_{i=0}^{m} n_i x_i$$
$$\bar{y} = \frac{1}{\sum_{i=0}^{m} n_i} \sum_{i=0}^{m} n_i y_i$$
$$S_{xx} = \sum_{i=0}^{m} n_i (x_i - \bar{x})^2$$
$$S_{xy} = \sum_{i=0}^{m} n_i (x_i - \bar{x})(y_i - \bar{y})$$
$$S_{yy} = \sum_{i=0}^{m} n_i (y_i - \bar{y})^2$$

Then the estimates $\hat{b}_0$ and $\hat{b}_1$ are given below

$$\hat{b}_1 = \frac{S_{xy}}{S_{xx}}$$
$$\hat{b}_0 = \bar{y} - \hat{b}_1 \bar{x}$$

The variance $\hat{\sigma}^2$ can be computed as

$$\hat{\sigma}^2 = \frac{\sum_{i=0}^{m} n_i (y_i - \hat{y}_i)^2}{\sum_{i=0}^{m} n_i - 2}$$

where $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i$.

Tests for the trend

The statistic for the hypothesis $H_0 \colon b_1 = 0$ is

$$t = \frac{\hat{b}_1 \sqrt{S_{xx}}}{\hat{\sigma}}$$

which has an asymptotic t-distribution with df $(= \sum_{i=0}^{m} n_i - 2)$ degrees of freedom. Then calculate the corresponding p value. If p value is less than 0.05, then the hypothesis is rejected.

The partial correlation of x and y adjusted for z is calculated as follows

$$r_{AB|C} = \frac{r_{AB} - r_{AC} r_{BC}}{\sqrt{(1 - r_{AC}^2)(1 - r_{BC}^2)}}$$

where r_{AB} denotes the correlation between A and B, $r_{AB} = \frac{s_{AB}}{\sqrt{s_{AA}}\sqrt{s_{BB}}}$. It should be noted that if $r_{AC} = 0$ or $r_{BC} = 0$, then $r_{AB|C}$ is set to 0.

For the plot, we provide the following insights

- If p value is less than 0.05, the current model does not represent the data
- If p value is greater than or equal to 0.05, we calculate the partial correlation $r_{yx^2|x}$
    - If $|r_{yx^2|x}| < 0.775$, the current model may represent the data
    - If $|r_{yx^2|x}| \geq 0.775$, the current model does not represent the data

Appendix A: Grouping analysis and unusual category detection

For a significant factor or factor interaction, we can infer that some categories or category combinations should have a statistically significant impact on the target. Here, we provide analyses to indentify which factor's (factor interaction's) categories have large impacts on the target. For the sake of brevity, the description is for a significant factor, but it also works for a significant factor interaction.

For all distributions except multinomial distribution and binomial distribution with 0/1 binary response, we propose two analyses which follow the analyses in the reference Shyr et al. (2011).

(1) Grouping analysis: Partitions all factor's categories into a high group and a low group (with a possible medium group) by conducting tests on whether the EMMEAN in each category is different from that in the category with the largest or smallest EMMEAN.

(2) Unusual category detection analysis: detects possible unusual categories in the high and low groups.

For multinomial distribution (including ordinal and multinomial) and binomial distribution with 0/1 binary response, we propose two analyses which are based on tests described in the reference Agresti (2002).

(1) Grouping analysis: partitions all categories into a significant group and an insignificant group by conducting tests on whether the target's categorical distribution in each category is different from that the overall distribution (population distribution).

(2) Influential target category analysis: identifies influential target categories for each significant category.

A.1. All distributions except multinomial and binomial distribution with 0/1 binary response

Let the m categories of a significant factor A be $A_1, \cdots, A_m$, and their corresponding EMMEANS are $M_1, \cdots, M_m$, respectively. Let the number of records in $A_1, \cdots, A_m$ be $n_1, \cdots, n_m$, respectively.

A.1.1 Grouping analysis

The following process is used to find the high and low groups and the possible medium group among all categories of a significant factor with more than 3 categories based on the EMMEANS for the target rather than the linear predictor.

1) For a significant factor A with m categories, compute the EMMEANS, $\boldsymbol{M} = \{M_1, \cdots, M_m\}$, and the corresponding variance matrix $\boldsymbol{V}$. See Chu and Zhong (2005, 2012), and Zheng (2009) for details of calculations of $\boldsymbol{M}$ and $\boldsymbol{V}$.
2) Sort the EMMEAN M_i $(i = 1, \cdots, m)$ by a descending order. Without loss of generality, assume that they are $M_1, M_2, \cdots, M_m$, namely, M_1 has the largest EMMEAN and M_m has the smallest one.
3) The category with the largest EMMEAN is firstly formed as the high group. Then test whether there is a significant difference between the second largest EMMEAN and the largest one. The test statistic is Wald chi-square statistics,

   $$s = \frac{(M_1 - M_2)^2}{\sigma^2}$$

   with 1 degree of freedom, where $\sigma^2 = \boldsymbol{V}_{11} + \boldsymbol{V}_{22} - 2\boldsymbol{V}_{12}$. The corresponding p-value is calculated accordingly.

   If the null hypothesis $M_1 - M_2 = 0$ is not rejected, i.e., the p-value is greater than α (significance level specified by the user, default is 0.05), then the category with M_2 will be added to the high group.

   It is noted that (a) if $M_1 - M_2 = 0$, then it does not need to compute σ^2 and assign the p-value = 1.0, i.e., the category with the second largest target mean will be added to the high group. (b) If $M_1 - M_2 \neq 0$ and $\sigma^2 = 0$, then the p-value = 0.0 and stops.
4) Repeat the same process for the next EMMEAN in line, i.e., compare M_3 with M_1, until there is no category can be added to the high group.
5) Similarly, form the low group from the smallest EMMEAN for those categories not assigned to the high group.
6) If there still exist some categories after forming the high and low groups, they are grouped into the medium group.

The method used above is an extension of that in Chu and Han (2011) to the case of generalized linear models. It should be noted that a Chi-square test is used rather than t-test.

Implementation notes

- When $M_i = M_j$ and $n_i \neq n_j$, if $n_i > n_j$, then M_i will be first to be compared to M_1 or M_m; if $n_i < n_j$, then M_j will be first to be compared to M_1 or M_m.

A.1.2 Unusual category detection analysis

The process to detect unusual categories for a significant factor is described as follows:

1) Calculate the median of m EMMEANS incorporating the number of records in each category. Denote MED by the median,

$$MED = \text{median}(\underbrace{M_1, \cdots, M_1}_{n_1}, \cdots, \underbrace{M_m, \cdots, M_m}_{n_m})$$

2) Calculate the median absolute deviation (MAD) of m target means, again incorporating with the number of records in each cell

$$MAD = \text{median}(|M_1 - MED|, \cdots, |M_1 - MED|, \cdots, |M_m - MED|, \cdots, |M_m - MED|)$$

3) Compute the modified z-score for the category $A_i, i = 1, \cdots, m$

$$z_i = \begin{cases} \dfrac{M_i - MED}{1.4826 \times MAD} & \text{if } MAD \neq 0 \\ \dfrac{M_i - MED}{1.2533 \times MeanAD} & \text{if } MAD = 0 \end{cases}$$

where $MeanAD = \frac{1}{N}\sum_{i=1}^{n} n_i |M_i - MED|$.

4) Detect unusual categories

If $z_i > 3$, the category A_i has an unusually high EMMEAN in the high group.

If $z_i < -3$, the category A_i has an unusually low EMMEAN in the low group.

A.2. Multinomial distribution and binomial distribution with 0/1 binary response

Notations

$A_1, \cdots, A_m$	The m categories of a significant factor A.
n_i	The number of records in A_i.
$\hat{p}_{ij}$	The EMMEAN value for j-th target category of the category A_i, where $\sum_{j=1}^{J} \hat{p}_{ij} = 1$.
$p_{\cdot j}$	The overall probability of the j-th target category, $j = 1, \cdots, J$ from the whole dataset.

A.2.1 Grouping analysis

Under the assumption that the overall target distribution is known and fixed, it will partition all categories into two groups: a significant group and an insignificant group by the following steps:

1) Compute the EMMEANS values for the category A_i, $\left(\hat{p}_{i1}, \cdots, \hat{p}_{iJ}\right)$, $i = 1, \cdots, m$:

$$\begin{cases} \hat{p}_{ij} = g^{-1}\left(\boldsymbol{L}_i\boldsymbol{\beta}_j\right) & \text{for } j = 1, \cdots, J-1 \\ \hat{p}_{iJ} = 1 - \sum_{j=1}^{J-1} \hat{p}_{ij} & \text{for } j = J \end{cases}$$

   where $\boldsymbol{L}_i$ is L matrix for the category A_i.

2) Compute the Pearson's one sample chi-square statistics and the corresponding p-value for each category A_i, $i = 1, \cdots, m$

$$\chi_i^2 = n_i \sum_{j=1}^{J} \frac{\left(\hat{p}_{ij} - p_{\cdot j}\right)^2}{p_{\cdot j}}$$

$$p_i = 1 - \Pr\left(\chi_{(J-1)}^2 \leq \chi_i^2\right)$$

   where $\chi_{(J-1)}^2$ is a random variable which following a chi-square distribution with $\mathrm{df} = (J-1)$ degrees of freedom.

3) Compute the effect size for each category

$$w_i = \sqrt{\frac{\chi_i^2}{n_i(J-1)}}$$

4) Sort the category $A_1, \cdots, A_m$ using $w_1, \cdots, w_m$ by a descending order. Without loss of the generality, assume that the order is $A_1, \cdots, A_m$.

   If $p_i < \alpha$, where α is a significant level (the default is 0.05), then the category A_i has a significantly different distribution from the overall distribution and will be added into the significant group.

   If $p_i \geq \alpha$, then the category A_i and $A_{i+1}, \cdots, A_m$ will be assigned to the insignificant group

5) The results are a list of the categories in the significant group with relevant test statistics, e.g. χ_i^2, df, p_i, and w_i.

Implementation notes

- When $w_i = w_j$ and $n_i \neq n_j$, if $n_i > n_j$, then the cell A_j will be first to be compared to the root node; if $n_i < n_j$, then the cell with A_i will be first to be compared to the root node.

A.2.2 Influential target category analysis

It identifies target categories, which have significantly large frequency differences from that of the root node, based on another chi-square test in the significant group (suppose it is Λ) by the following steps:

1) Compute the chi-squared statistics, and the corresponding p-value for the category $i \in \Lambda$ and for the j-th target category

$$\chi_{ij}^2 = \frac{n_i\left(\hat{p}_{ij} - p_{\cdot j}\right)^2}{p_{\cdot j} \times (1 - p_{\cdot j})}$$

$$p = 1 - \Pr\left(\chi_1^2 \leq \chi_{ij}^2\right)$$

where χ_1^2 is a random variable which following a chi-square distribution with $\mathrm{df} = 1$ degree of freedom.

If $p < \alpha$, where α is a significant level (the default is $0.05/J$ based on the Bonferroni adjustment method), then the j-th target category is an influential target category for the category $i \in \Lambda$.

2) The results are a list of influential target categories for each significant category with relevant test statistics.

Influential target category detection analysis is only performed on the non-binary target. If the target is binary, the chi-square statistic $\chi_{i,j}^2$ $(j = 1,2)$ is equal to χ_i^2, namely, $\chi_{i,1}^2 = \chi_{i,2}^2 = \chi_i^2$. It could be expected that both two categories are influential for each significant cell i.

References – Phase II

[20]. Agresti, A. (2002), *Categorical Data Analysis*, Second Edition, Hoboken, NJ: John Wiley & Sons, Inc.

[21]. Chu, J. and Han, S. (2011), “Algorithm: Linear Engine”, *IBM SPSS Internal Document*..

[22]. Chu, J. and Zhong, W. (2005), “Algorithm: Generalized linear models and generalized estimating equation,” *SPSS Internal Document.*

[23]. Chu, J. and Zhong, W. (2012), “Algorithm: Generalized linear engine phase I”, *IBM SPSS Internal Document*.

[24]. Dagli, A. (2012), “Simple random sampling in Map-Reduce”, *IBM SPSS Internal Document*

[25]. Dunn, P. K. and Smyth, G. K. (2008). "Evaluation of Tweedie exponential dispersion model densities by Fourier inversion". *Statistics and Computing*, **18**, 73-86.

[26]. Mittlbock, M. and Heinzl, H. "Pseudo R-squared measures for generalized linear models," The 1st European Workshop on the Assessment of Diagnostic Performance, 2004, 71-80.

[27]. Shyr, J., Chu, J. and Han, S. (2011), "Category profiling and unusual category detection based on Estimated Marginal Means (EMMEANS)", In *JSM Proceedings, Social Statistics Section*, Alexandria, VA: American Statistical Association. 4289-4300.

[28]. Zheng, P. (2009), "Algorithm: EMMEANS and custom tests," *SPSS Internal Document*.

11. Introduction – Phase III

Generalized Linear Engine Phase III (GLE Phase III) adds two main features on top of GLE Phase I (Chu and Zhong, 2012) and Phase II (Zhong and Han, 2013):

- Estimation of generalized linear models for the large p situations (the number of parameters (p) is greater than or equal to a threshold (p_c, the default = 5000), i.e., $p \geq p_c$.

  Besides the optimization issue for parameter estimation, the other difficult issue is the post-estimation statistics.

- Estimation of regularized generalized linear models: L_1 (the lasso), L_2 (ridge regression) and mixtures of two penalties (the elastic net). This feature is applicable for both the large p situations and the small and medium p situations.

Both features will be solved by using the new optimization engine ADMM (Zhong, 2014) which is distributed optimization framework and can solve the problems with large numbers of parameters and records.

Besides the optimization issue of parameter estimation, the other difficult issue introduced by the large p situations is the post-estimation statistics. The reason is that a large number of parameters, even after the variable selection, would cause difficulty to calculate the parameter estimate covariance matrix (minus inverse of Hessian matrix) and thus many statistics, such as Wald test, etc., based on it. Here, we will firstly transform the original problem of calculating the statistics into a linear system, and then use ADMM to solve it.

The organization of this document is Section 12 describes parameter estimation for the large p situation. Section 13 describes parameter estimation with regularizations. Finally, how to compute post-estimation statistics without the parameter estimate covariance matrix when p is large is given in Section 14.

Notations

N	The number of data blocks (parts)
p	The number of parameters. Note that it doesn't include the scale parameter for continuous distributions or the auxiliary parameter for the negative binomial distribution.
$\boldsymbol{\beta}_i$	$\boldsymbol{\beta}_i \in \mathbf{R}^p$ denotes parameters for the i-th data block
$\boldsymbol{z}$	The common global parameters, where $\boldsymbol{z} \in \mathbf{R}^p$
$\boldsymbol{u}_i$	$\boldsymbol{u}_i \in \mathbf{R}^p$ denotes the Lagrange multipliers of the i-th term in the objective

$f_i(\cdot)$	The i-th term in the objective for the i-th data block
$g(\cdot)$	The regularization (penalty function)
ρ	The augmented Lagrange parameter
$\boldsymbol{y}$	The target variable
$\boldsymbol{X}$	$\boldsymbol{X} = [\boldsymbol{X}_1, \cdots, \boldsymbol{X}_p]$ denotes the design matrix, where $\boldsymbol{X}_j$ denotes the j-th column
$\lVert\boldsymbol{z}\rVert_1$	The L_1 norm of the vector $\boldsymbol{z}$, which is defined as $\lVert\boldsymbol{z}\rVert_1 = \lvert z_1\rvert + \cdots + \lvert z_p\rvert$
$\lVert\boldsymbol{z}\rVert_2$	The L_2 norm of the vector $\boldsymbol{z}$, which is defined as $\lVert\boldsymbol{z}\rVert_2 = \left(z_1^2 + \cdots + z_p^2\right)^{1/2}$
p_c	The threshold denoting whether there is a large number of parameters (large p). If $p \geq p_c$, it is called large p situation, otherwise, it is called small to medium p situation.
$\boldsymbol{s}$	The gradient vector (function)
$\mathbf{H}$	The Hessian matrix (function)

12. Parameter estimation for the large p situations

For the generalized linear models, the parameter estimation is based on the maximum likelihood method as $\max_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta})$. Since ADMM usually solves the optimization problem in the form of minimization, the maximum likelihood method can be written as

$$\min_{\boldsymbol{\beta}} -\ell(\boldsymbol{\beta})$$

The $\ell(\boldsymbol{\beta})$ is separable with respect to the partition of the records, $\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \ell_i(\boldsymbol{\beta})$. If we optimize $\ell_i(\boldsymbol{\beta})$ for each data block i, then we obtain the following form for ADMM

$$\begin{array}{ll} \min_{\boldsymbol{\beta}_i, \boldsymbol{z}} & -\sum_{i=1}^{N} \ell_i(\boldsymbol{\beta}_i) \\ s.t. & \boldsymbol{\beta}_i - \boldsymbol{z} = \boldsymbol{0}, i = 1, \cdots, N. \end{array} \tag{12.1}$$

For the large p situations ($p \geq p_c$), the L-BFGS method in ADMM will be used to solve Equation (12.1). In order to call ADMM, GLE needs to prepare three pieces of information: the fitting function, the gradient function, and initial values.

The fitting function $f_i(\boldsymbol{\beta}_i)$ is

$$f_i(\boldsymbol{\beta}_i) = -\ell_i(\boldsymbol{\beta}_i) \tag{12.2}$$

The gradient of $f_i(\boldsymbol{\beta}_i)$ is

$$\mathbf{s}_i = -\nabla \ell_i(\boldsymbol{\beta}_i) \tag{12.3}$$

The initial values can be computed as

$$\boldsymbol{\beta}^0 = \mathbf{0} \tag{12.4}$$

Implementation notes:

- The forms of ℓ_i and $\mathbf{s}_i$ for different distributions can be found in GLE Phase I.
- If the scale parameter for continuous distributions or the auxiliary parameter for the negative binomial distribution is estimated with regression parameters, then $\mathbf{s}_i$ should include it and the initial value is also set to 0.

13. Parameter estimation with regularizations

For the regularized generalized linear models, a penalty function will be added into Equation (12.1) then we obtain the following form for ADMM

$$\begin{array}{ll} \min_{\boldsymbol{\beta}_i,\boldsymbol{z}} & -\sum_{i=1}^{N} \ell_i(\boldsymbol{\beta}_i) + g(\boldsymbol{z}) \\ s.t. & \boldsymbol{\beta}_i - \boldsymbol{z} = \mathbf{0}, i = 1, \cdots, N. \end{array} \tag{13.1}$$

GLE will support the following penalty functions:

(1) The L_1 regularization (the lasso): $g(\boldsymbol{z}) = \lambda\|\boldsymbol{z}\|_1$.

(2) The L_2 regularization (ridge regression): $g(\boldsymbol{z}) = \lambda\|\boldsymbol{z}\|_2^2$.

(3) The $(L_1 + L_2)$ regularization (elastic net): $g(\boldsymbol{z}) = \lambda_1\|\boldsymbol{z}\|_1 + \lambda_2\|\boldsymbol{z}\|_2^2$.

Note that λ, λ_1 and λ_2 are penalty parameters to regulate the strength of penalty. For lasso or ridge regression, $\lambda = 0$ implies unconstrained solution and $\lambda = \infty$ implies totally constrained solution ($\boldsymbol{z} = \mathbf{0}$). For elastic net, $\lambda_1 = \lambda_2 = 0$ implies unconstrained solution and one of λ_1 or $\lambda_2 = \infty$ implies totally constrained solution ($\boldsymbol{z} = \mathbf{0}$). Both L_1 and L_2 regulations prevent overfitting by shrinking (imposing a penalty) on the parameters. The L_1 regulation can shrink some parameters to zero, performing variable selection, while the L_2 regulation shrinks all the parameters by the same proportions but eliminates none. In terms of model fits, the L_2 regulation usually performs better than the L_1 regulation in practice. Even the $(L_1 + L_2)$ regularization might perform better than the L_1 regularization.

There are two ways to choose these parameters:

(1) They are set to fixed values in the range of $[0,\ \infty]$.

(2) They are chosen by a grid search method as follows:

a) Partition the data into two parts: training and testing sets. By default, the ratio of training to testing is $0.7{:}\,0.3$.

b) Specify the maximum value, $\lambda_{\max}$:

(i) For the lasso, use the method in Park and Hastie (2007): if the target does not follow nominal multinomial distribution or ordinal distribution,

$$\lambda_{\max} = \max_{j\in\{1,\cdots,p\}}\left|\boldsymbol{X}_j^{\mathrm{T}}\boldsymbol{W}(\mathbf{y}-\bar{y}\mathbf{1})\mathrm{g}'(\bar{y})\right| \tag{13.2}$$

where $\boldsymbol{W}$ is $n\times n$ diagonal matrix with the i^{th} diagonal element

$$w_{ii} = \frac{f_i\omega_i}{V(\bar{y})(\mathrm{g}'(\bar{y}))^2}$$

where $V(\cdot)$ is variance function and $\mathrm{g}(\cdot)$ is the link function.

If target follows nominal multinomial distribution

$$\lambda_{\max} = \max_{j\in\{1,\cdots,p\}}\left\{\max_{k\in\{1,\cdots,J-1\}}\left|\boldsymbol{X}_j^{\mathrm{T}}\boldsymbol{W}\left(\mathbf{y}^{(\mathbf{k})}-\bar{y}^{(\mathrm{k})}\mathbf{1}\right)\right|\right\} \tag{13.3}$$

where J is the number of categories of target, $\boldsymbol{W}$ is $n\times n$ diagonal matrix with the i^{th} diagonal element $f_i\omega_i$, $\mathbf{y}^{(\mathrm{k})} = (y_{1k}, y_{2k}, \cdots, y_{nk})^{\mathrm{T}}$and $\bar{y}^{(\mathrm{k})} = \frac{\sum_{i=1}^{n} f_i y_{ik}}{\sum_{i=1}^{n} f_i}$, Here, $y_{ik} = 1$ if the target value of the i^{th} record takes the k^{th} category, otherwise $y_{ik} = 0$.

If target follows ordinal distribution

$$\lambda_{\max} = \max_{\mathrm{j}\in\{1,\cdots,\mathrm{p}\}}\left|\boldsymbol{X}_{\boldsymbol{j}}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{d}\right| \tag{13.4}$$

where $\boldsymbol{W}$ is $n\times n$ diagonal matrix with the i^{th} diagonal element $f_i\omega_i$, $\boldsymbol{d}$ is a $n\times 1$ vector with the i^{th} element

$$d_i = \sum_{k=1}^{J}\left(\frac{\partial\gamma_{i,k}}{\partial\eta_{i,k}} - \frac{\partial\gamma_{i,k-1}}{\partial\eta_{i,k-1}}\right)\frac{y_{i,k}}{\pi_{i,k}}$$

where J is the number of categories of target; $y_{i,k} = 1$ if the target value of the i^{th} record takes the k^{th} category, otherwise $y_{i,k} = 0$; $\pi_{i,k} = \frac{n_{k,f}}{n_f}$ for $k = 1, \cdots, J$ with $n_{k,f}$ being the number of records for the kth category of target incorporating the frequency weight and n_f being the total number of records incorporating frequency weight. The $\frac{\partial \gamma_{i,k}}{\partial \eta_{i,k}}$ is defined in the GLE phase I. Since the maximum value of $\frac{\partial \gamma_{i,k}}{\partial \eta_{i,k}} - \frac{\partial \gamma_{i,k-1}}{\partial \eta_{i,k-1}}$ is less than 0.5, we will use $0.5 * \sum_{k=1}^{J} \frac{y_{i,k}}{\pi_{i,k}}$ as approximate d_i, i.e. $d_i \approx 0.5 * \sum_{k=1}^{J} \frac{y_{i,k}}{\pi_{i,k}}$.

(ii) For ridge regression, there is no limitation of the maximum value. User could specify it. By default we use follow value.

$$\lambda_{\max} = e^{20}$$

(iii) For elastic net, which includes lasso and ridge regression, we will set two regularization parameters for L1 and L2 respectively when invoking ADMM. But they can be specified with the relationship as follows.

$$\lambda_1 = \alpha\lambda,\ \ \lambda_2 = (1-\alpha)\lambda$$

Where $0 < \boldsymbol{\alpha} < 1$, λ_1 is for L1 regularization and λ_2 is for L2 regularization. The value of λ_2 can be got easily with given value of λ_1. Therefore, only the maximum value for $\lambda_1$needs to be determined. We will specify the maximum value of λ_1 using the same method as that we specify the maximum value of λ for lasso in (i).

c) Set the minimal value $\lambda_{\min} = e^{-10}$.

d) Select the number of search points, n_λ (the default is 100), and determine those points:

$$\lambda_{\max\,,}\, \lambda_{\min}\, e^{(\mathrm{n}_\lambda - 2)\Delta}, \lambda_{\min}\, e^{(\mathrm{n}_\lambda - 3)\Delta}, \ldots, \lambda_{\min}\, e^{\Delta}, \lambda_{\min}$$

where $\Delta = \frac{\log \lambda_{\max} - \log \lambda_{\min}}{n_\lambda - 1}$.

Note: For elastic net, the search points of λ_1 are set as above. Then for each search point, the corresponding value of λ_2 will be determined with a fixed value of α by following formula:

$$\lambda_2 = \frac{(1-\alpha)}{\alpha}\lambda_1$$

We will do a grid search with different combinations of λ_1 and λ_2, which are generated by the combinations of search points of λ_1 and search points of α.

Regarding to the sequence of the search points of α, we will use {0.1, 0.2, ... 0.9} by default. User could specify this sequence, with each element in the sequence in the range of (0.0, 1.0). Note that the values 0.0 and 1.0 are not allowed.

e) We will build a model for each λ (for lasso), or build a model for each λ (for ridge regression), or build a model for each combination of λ_1 and λ_2 (for elastic net) on the training set, and calculate the ASE value on the testing set.

f) Output the model with λ or the combination of λ_1 and λ_2 and the corresponding the minimal ASE value.

Implementation notes:

- The warm-start strategy will be used to speed up the grid search process. It means that we will build the models from $\lambda_{\max}$ to$\lambda_{\min}$ sequentially and the next model will use the solution obtained from the current model as the initial values.
- $j \in \{2, \cdots, p\}$ in Equation (13.2) and (13.3) if there is an intercept. For ordinal distribution, the index j is always from 1 to p because the design matrix $\boldsymbol{X} = [\boldsymbol{X}_1, \cdots, \boldsymbol{X}_p]$ does not contain intercept in the GLE phase I.
- The definition of ASE is given in GLE Phase II.

For the large p situations, the same three pieces of information shown above need to be prepared. For the small and medium p situations, one extra piece of information is needed: the Hessian matrix of $f_i(\boldsymbol{\beta}_i)$:

$$\mathbf{H}_i = -\nabla^2 \ell_i(\boldsymbol{\beta}_i) \tag{13.5}$$

Implementation notes:

- If the scale parameter for continuous distributions or the auxiliary parameter for the negative binomial distribution is estimated with regression parameters, then (a) partial penalty in hybrid penalty functions (Section 3.2.2 in ADMM ADD) should be used because no penalty is applied on the scale parameter or auxiliary parameter.
- If some predictors are categorical, i.e., factors, then (b) group penalty in hybrid penalty functions should be used.
- If the above conditions exist at the same time, then (c) partial group penalty in hybrid penalty functions should be used.

- We will not include two-way interaction effects in the regularized generalized linear models.
- The threshold value for judging any regression parameter $\beta = 0$ is 1.0e-12, i.e., if $\beta < 1.0\mathrm{e} - 12$, then the corresponding predictor is not entered into the model.
- If a sample is used for model selection in the regularized generalized linear models, then it is possible that $N = 1$ (there is only one data block). In this case, the mean of local solutions would be just from one local solution, $\overline{\boldsymbol{\beta}}^{k+1} = \boldsymbol{\beta}_1^{k+1}$ and $\overline{\boldsymbol{u}}^{k} = \boldsymbol{u}_1^{k}$, then they would be used to update the global parameter $\boldsymbol{z}^{\boldsymbol{k+1}}$.

14. Post-estimation statistics

Many post-estimation statistics will be based on the parameter estimate covariance matrix, $\boldsymbol{\Sigma} = -\mathbf{H}^{-1}$. For instance, confidence interval and chi-square statistics for parameters, Lagrange multiplier test, model effect test, custom test, EMMEANS, standard errors of predicted values, and leverage values. However, it is impossible to directly calculate the inverse of Hessian matrix **H** in the large p situations because its computational cost scales as $O(p^3)$.

After analysis, we found that the post-estimation statistics have two ways to use the parameter estimate covariance matrix

(1) Involving $\mathbf{L\Sigma L}^{\mathrm{T}}$.

(2) Involving the diagonal values of $\boldsymbol{\Sigma}$.

We would solve these two problems by transferring them into linear system problems.

14.1. Solving $\mathbf{L\Sigma L}^{\mathrm{T}}$

Let $\mathbf{V} = \boldsymbol{\Sigma}\mathbf{L}^{\mathrm{T}} \in \mathbf{R}^{p\times r}$, then we have $\mathbf{L\Sigma L}^{\mathrm{T}} = \mathbf{LV}$. Usually, r is quite small comparing with p, so it is easier to calculate $\mathbf{LV}$ with $\mathbf{V}$ than to calculate $\mathbf{L\Sigma L}^{\mathrm{T}}$ with $\boldsymbol{\Sigma}$. The key problem is to compute $\mathbf{V}$.

To compute $\mathbf{V}$, we do the transformation as follows:

$$\mathbf{V} = \boldsymbol{\Sigma}\mathbf{L}^{\mathrm{T}} \Rightarrow \boldsymbol{\Sigma}^{-}\mathbf{V} = \mathbf{L}^{\mathrm{T}} \Rightarrow (-\mathbf{H})\mathbf{V} = \mathbf{L}^{\mathrm{T}} \tag{14.1}$$

which is a linear system problem. We could use the ADMM to solve it.

Note that when $r > 1$, $\mathbf{L}$ is a matrix, thus, $\mathbf{V}$ is a matrix as well. We will use vec operator (Lam, 1995) to convert $\mathbf{L}$ to a vector form, $\mathrm{vec}(\mathbf{L})$. Thus, Equation (4.1) becomes

$$[\mathbf{1}_r \otimes (-\mathbf{H})]\text{vec}(\mathbf{V}) = \text{vec}(\mathbf{L}^{\mathrm{T}}) \tag{14.2}$$

14.1.1 ADMM for a linear system problem

Considering a general linear system $\mathbf{Av} = \mathbf{b}$, the three pieces of information used to call ADMM are

(1) the fitting function: $f(\mathbf{v}) = \frac{1}{2}(\mathbf{v}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Av} - \mathbf{2b}^{\mathrm{T}}\mathbf{Av} + \mathbf{b}^{\mathrm{T}}\mathbf{b})$;

(2) the gradient of $f(\mathbf{v})$: $\mathbf{s} = \mathbf{A}^{\mathrm{T}}\mathbf{Av} - \mathbf{A}^{\mathrm{T}}\mathbf{b}$;

(3) the initial value: $\mathbf{v}^0 = \mathbf{0}$.

14.2. Calculating diagonal values of Σ

Similar to the previous section, we convert the problem of estimating the diagonal values of $\mathbf{\Sigma}$ to a linear system problem.

To obtain the i^{th} diagonal value, we could first generate a vector, $\mathbf{w}_i = [w_1, \cdots, w_i, \cdots, w_p]^T$, where $w_i = 1$, and $w_j = 0$ for $j \neq i$. Then we have $\mathbf{v}_i = \mathbf{\Sigma w}_i$ and the i^{th} element in $\mathbf{v}_i$ is the i^{th} diagonal value of $\mathbf{\Sigma}$.

Further, we can see that to obtain $\mathbf{v}_i$ is equivalent to solve the following linear system problem

$$(-\mathbf{H})\mathbf{v}_i = \mathbf{w}_i \tag{14.3}$$

which we could use ADMM method to get the solution of $\mathbf{v}_i$.

Implementation notes:

- Even there is a method to calculate $\mathbf{L\Sigma L}^{\mathrm{T}}$ and diagonal values of $\mathbf{\Sigma}$, we still need to compute the Hessian matrix **H** after the parameter estimation process. For the large p situations, all elements in **H** might not be saved in memory, not to mention the computation of $\mathbf{L\Sigma L}^{\mathrm{T}}$ for all effects and diagonal values of $\mathbf{\Sigma}$ is extremely time consuming, so we will try to keep the final # of parameters is less than p_c. For example, conduct feature selection (in the SDP (Smart Data Preprocessing) or DE (Descriptive Engine)) before running GLE and/or select the lasso penalty to perform variable selection within GLE.

References

[29]. Chu, J. and Zhong, W. (2012), “Algorithm: Generalized linear engine phase I”, *IBM SPSS Internal Document*.

[30]. Lam, M.L (1995), “A general overview of the multivariate β-model” , *SPSS Internal Document*.

[31]. Park, M. Y. and Hastie, T. (2007), “L_1-regularization path algorithm for generalized linear models”, J. R. Statist. Soc. B, 69 (4): 659-677.

[32]. Zhong, W. and Han, S. (2013), “Algorithm: Generalized linear engine phase II”, *IBM SPSS Internal Document*.

[33]. Zhong, W. (2014), “Algorithm: ADMM”, *IBM SPSS Internal Document*.

Linear-AS Modeling Algorithms

1. Linear AS (Phase I)

A linear regression model usually analyzes the relationship between one target variable (also called responses) and a list of feature variables (also called predictors). Linear-AS, also known as the "Linear Engine", builds linear regression models for large and distributed data and runs within Analytic Engine (AE).

2. Notations

The following notation is used throughout the document unless otherwise stated:

n	Number of distinct records in the dataset. It is an integer and $n \geq 1$.
p	Number of parameters (including parameters for dummy variables but excluding intercept) in the model. It is an integer and $p \geq 0$.
p^*	Number of non-redundant parameters (excluding intercept if exists) currently in the model. It is an integer and $0 \leq p^* \leq p$.
p^c	Number of non-redundant parameters currently in the model, so $p^c = \begin{cases} p^* + 1 & \text{if there is an intercept} \\ p^* & \text{if there is no intercept} \end{cases}$.
p^e	Number of effects excluding intercept. it is an integer and $0 \leq p^e \leq p$
$\mathbf{y}$	$n \times 1$ vector of single target variable consists of y_i.
f	$n \times 1$ vector of frequency count variable. If an element is not an integer, it is computed by rounding the value to the nearest integer. If it is less than 0.5 or if it is missing, the corresponding case is not used.
g	$n \times 1$ vector of regression weight. If there is no regression weight specified, $g = 1$. If regression weight g_i for case i is zero, negative or missing. The corresponding case is not used.
N	Effective sample size. it is a integer number, $N = \sum_{i=1}^{n} f_i$ .If frequency count variable f is not use, N=n.
$\boldsymbol{X}$	$n \times (p+1)$ design matrix. The rows represent the cases and the columns represent the

	parameters. The i^{th} row is $\boldsymbol{x}_i = (x_{i0},..., x_{ip})$, $i = 1,2,..., n$, with $x_{i0} = 1$, The *j*th column is $\boldsymbol{X}_j = (x_{1j},..., x_{nj})^{\mathrm{T}}$, , $j = 0,1,..., p$, with $\boldsymbol{X}_0 = (1,...,1)^{\mathrm{T}}$. . If there is no intercept, $\boldsymbol{X} = \{\boldsymbol{X}_j\}_{j=1}^{p}$ is a $n \times p$ matrix.
ε	$n \times 1$ vector of unobserved errors .
β	$(p+1)\times 1$ vector of unknown parameters. $\beta = (\beta_0, \beta_1, \cdots \beta_p)$. β_0 is the intercept, if exist. If there is no intercept, $\beta = (\beta_1, \cdots \beta_p)^T$ is a $p \times 1$ vector.
$\hat{\beta}$	$(p+1)\times 1$ vector of estimated β . $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \cdots \hat{\beta}_p)$. If there is an intercept, $\hat{\beta}_0$ is its estimate, else $\hat{\beta} = (\hat{\beta}_1, \cdots \hat{\beta}_p)^T$ is a $p \times 1$ vector.
b	$(p+1)\times 1$vector of standardized estimate of β .It is the result from sweep operation on matrix ***R***. $b = (b_0, b_1, \cdots b_p)$. If there is an intercept, b_0 is its standardized estimate, else $b = (b_1, \cdots b_p)^T$ is a $p \times 1$ vector.
$\hat{\mathbf{y}}$	Predicted value of $\mathbf{y}$**,** consists of $\hat{y}_i$
$\overline{X}_j$	Weighted sample mean for X_j , $j = 1,2, \cdots p$
$\bar{y}$	Weighted sample mean for the dependent variable *y*.
S_{ij}	Weighted sample covariance between X_i and X_j . $i, j = 1,2, \cdots p$
S_{iy}	Weighted sample covariance between X_i and *y*.
S_{yy}	Weighted sample variance for *y*.
R	$(p+1)\times(p+1)$ weighted sample correlation matrix for ***X*** (exclude intercept, if exist) and ***y*****.** It is also used to represent the current sweeping matrix before each sweep operation step.
$\tilde{\boldsymbol{R}}$	The result matrix after sweep operation whose elements are $\tilde{r}_{ij}$.

3. Model

Linear regression of single target variable ***y*** and design matrix ***X*** has the form

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1}$$

where $\boldsymbol{\varepsilon}$ follows a normal distribution with mean $\mathbf{0}$ and variance $\sigma^2\boldsymbol{D}^{-1}$, i.e., $\boldsymbol{\varepsilon} \sim N_n\left(\mathbf{0},\ \sigma^2\boldsymbol{D}^{-1}\right)$ with $\boldsymbol{D}^{-1} = \mathrm{diag}\left(1/g_1, \ldots, 1/g_n\right)$. Note that the k^{th} case will be ignored if $g_k \le 0$. Then the target variable $\boldsymbol{y}$ also follows a normal distribution with mean $\boldsymbol{X\beta}$ and variance $\sigma^2\boldsymbol{D}^{-1}$, $\boldsymbol{y} \sim N_n\left(\boldsymbol{X\beta},\ \sigma^2\boldsymbol{D}^{-1}\right)$.

Notes:

1. The elements of $\boldsymbol{\varepsilon}$ are independent with each other, so are those of $\boldsymbol{y}$.
2. $\boldsymbol{X}$ can be any combination of continuous and categorical effects and interaction effects (up to two-way). The parameterization of design matrix $\boldsymbol{X}$ is the same as that in GLM procedure. See Lam (1995a) for further details on the model parameterization. Please note that we might expand interaction effects to include more than two-way and nested effects used in old SPSS procedures.

3.1. Missing values

List-wise deletion is used.

If missing value handling feature has been conducted in “Bivariate Data Preparation” component, then only the target variable still has missing values and those records would be excluded.

4. Least Squares Coefficient Estimation

The coefficients would be estimated by least squares (LS) method with the following closed form solution

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{WX}\right)^{-}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Wy}, \tag{2}$$

where $\boldsymbol{W} = \mathrm{diag}\left(w_1,\ldots,w_n\right) = \mathrm{diag}\left(g_1 f_1,\ldots,g_n f_n\right)$.

The actual computation of $\hat{\boldsymbol{\beta}}$ is done by applying sweep operations instead of applying equation (2). In addition, sweep operations would be applied to transformed scale of $\boldsymbol{X}$ and $\boldsymbol{y}$ to achieve numerical stability. Specifically, we construct the weighted sample correlation matrix $\boldsymbol{R}$ then apply sweep operations to it. The construction the $\boldsymbol{R}$ matrix is described as follows.

First, compute weighted sample means, variances and covariances among $\boldsymbol{X}_i, \boldsymbol{X}_j,\ i, j = 1,\ldots, p,$ and $\boldsymbol{y}$:

Weights sample means of $\boldsymbol{X}_i$ and $\boldsymbol{y}$ are $\bar{X}_i = \frac{1}{\sum_{k=1}^{n} w_k} \sum_{k=1}^{n} w_k x_{ki}$ (3)

and $\bar{y} = \frac{1}{\sum_{k=1}^{n} w_k} \sum_{k=1}^{n} w_k y_k$; (4)

Weighted sample covariance for $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ is $S_{ij} = \frac{1}{N-1} \sum_{k=1}^{n} w_k (x_{ki} - \bar{X}_i)(x_{kj} - \bar{X}_j)$; (5)

Weighted sample covariance for $\boldsymbol{X}_i$ and $\boldsymbol{y}$ is $S_{iy} = \frac{1}{N-1} \sum_{k=1}^{n} w_k (x_{ki} - \bar{X}_i)(y_k - \bar{y})$; (6)

Weighted sample variance for $\boldsymbol{y}$ is $S_{yy} = \frac{1}{N-1} \sum_{k=1}^{n} w_k (y_k - \bar{y})^2$. (7)

If there is no intercept, $S_{ij} = \frac{1}{N-1} \sum_{k=1}^{n} w_k x_{ki} x_{kj}$, (8)

$$S_{iy} = \frac{1}{N-1} \sum_{k=1}^{n} w_k x_{ki} y_k \ , \tag{9}$$

$$S_{yy} = \frac{1}{N-1} \sum_{k=1}^{n} w_k y_k^2 \ . \tag{10}$$

Second, compute weighted sample correlations $r_{ij} = \frac{S_{ij}}{\sqrt{S_{ii} S_{jj}}}$, $i, j = 1, \ldots, p$ & y . (11)

Implementation notes: All statistics are computed in map/reduce environment, see Section A.2 in Appendix A of this section for details.

Then the matrix $\boldsymbol{R}$ is

$$\boldsymbol{R} = \left[\begin{array}{cccc|c} r_{11} & r_{12} & \cdots & r_{1p} & r_{1y} \\ r_{21} & r_{22} & \cdots & r_{2p} & r_{2y} \\ \vdots & \vdots & \ddots & \vdots & \\ r_{p1} & r_{2p} & \cdots & r_{pp} & r_{py} \\ \hline r_{y1} & r_{y2} & \cdots & r_{yp} & r_{yy} \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ \hline \boldsymbol{R}_{12}^{\mathrm{T}} & R_{22} \end{array}\right]. \tag{12}$$

If the sweep operations are repeatedly applied to each row of $\boldsymbol{R}_{11}$ (see Appendix B in this section), where $\mathbf{R}_{11}$ contains predictors in the model at the current step, the result is

$$\tilde{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R}_{11}^{-1} & \boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12} \\ -\boldsymbol{R}_{12}^{\mathrm{T}}\boldsymbol{R}_{11}^{-1} & R_{22} - \boldsymbol{R}_{12}^{\mathrm{T}}\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12} \end{bmatrix}. \tag{13}$$

The last column $\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12}$ contains the standardized coefficient estimates, i.e., $\boldsymbol{b} = \boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12}$. Then the coefficient estimates $\hat{\boldsymbol{\beta}}$, except the intercept estimate if there is an intercept in the model could be obtained as follows:

$$\hat{\beta}_j = b_j\sqrt{\frac{S_{yy}}{S_{jj}}}, \quad j = 1,..., p. \tag{14}$$

5. Automatic Interaction Effect Detection

We'd like to catch two-way interaction among main effects in $\boldsymbol{X}$ in the model selection phase, but including all possible pairs would make model selection difficult and inefficient. Thus we will limit to the following steps.

1. For all covariates (continuous variables), a squared term of each covariate will be created and included in the design matrix $\boldsymbol{X}$, but not the cross product terms, i.e. suppose there are two continuous variables, say $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, then $\boldsymbol{X}_1^2$ and $\boldsymbol{X}_2^2$ will be created, but not $\boldsymbol{X}_1 \times \boldsymbol{X}_2$.
2. For each pair of two factors (categorical variables), say $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, the ANOVA method will be used to test whether the interaction effect $\boldsymbol{X}_1 \times \boldsymbol{X}_2$ should be included. See Section 5.1 for details.
3. For each pair of one covariate and one factor, the ANOVA method is used as well. See Section 5.2 for details.

However, even with this original limitation, it might not be possible to check all candidate pairs in Steps 2 and 3 or include all eligible pairs from all 3 steps for the model selection methods in Section 6. The reason is, if there are large number of main effects in $\boldsymbol{X}$, the whole process might require too much memory (so user might receive "run out of memory" message and no output at all) or too much computational cost (so user might wait for a long time to receive output). Hence, we provide a two-way-test pair search strategy to restrict number of the pairs in those which are more likely to be selected to the final model in the model selection method. See Section 5.3 for details.

5.1. Interaction of two factors

Suppose the pair of two factors is X_1 with known R levels $(1,\ldots,R)$ and X_2 with known S levels $(1,\ldots,S)$. Instead of fitting a model for each pair, we will compute some statistics to implement the ANOVA method by the following steps:

1. Create a $R \times S$ contingency table based on $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ with the following statistics for each combination of $X_1 = i,\ i = 1,\ldots,R,$ and $X_2 = j,\ j = 1,\ldots,S$:

   n_{ij} : the number of distinct records;

   f_{ijk} : the frequency weight for the k^{th} distinct record, $k = 1,\ldots,n_{ij}$;

   y_{ijk} : the target value for the k^{th} distinct record, $k = 1,\ldots,n_{ij}$;

   N_{ij} : effect sample size (including frequency weights), i.e., $N_{ij} = \sum_{i=1}^{n_{ij}} f_{ijk}$;

   $\bar{y}_{ij}$: the target mean; $\bar{y}_{ij} = \frac{1}{N_{ij}} \sum_{i=1}^{n_{ij}} f_{ijk} \times y_{ijk}$;

   $C_{yy,ij}$: the sum of squared terms of target, i.e. $C_{yy,ij} = \sum_{k=1}^{n_{ij}} f_{ijk} \left(y_{ijk} - \bar{y}_{ij} \right)^2$.

X_2 X_1	1	2	$\cdots$	S
1	$N_{11}, \bar{y}_{11}, C_{yy,11}$	$N_{12}, \bar{y}_{12}, C_{yy,12}$	$\cdots$	$N_{1S}, \bar{y}_{1S}, C_{yy,1S}$
2	$N_{21}, \bar{y}_{21}, C_{yy,21}$	$N_{22}, \bar{y}_{22}, C_{yy,22}$	$\cdots$	$N_{2S}, \bar{y}_{2S}, C_{yy,2S}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
R	$N_{R1}, \bar{y}_{R1}, C_{yy,R1}$	$N_{R2}, \bar{y}_{R2}, C_{yy,R2}$	$\cdots$	$N_{RS}, \bar{y}_{RS}, C_{yy,RS}$

2. Compute residual sum of squares for the full model which contains two main effects $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ and the interaction effect $\boldsymbol{X}_1 \times \boldsymbol{X}_2$:

$$SS_{e,\,full} = \sum_{i=1}^{R} \sum_{j=1}^{S} \sum_{k=1}^{n_{ij}} f_{ijk} \left(y_{ijk} - \bar{y}_{ij} \right)^2 \tag{15}$$

3. Denote $SS_{e,interaction}$ to be the difference of residual sum of squares between the full model and the main effects only model and we will approximate it by the following iterative process:

(a) Input values for M (maximum number of iterations and the default is 10) and ε **(**tolerance level of stopping criterion and the default is 1.0e-6).

(b) Compute the initial value $SS_{e,interaction}^{(0)} = \sum_{i=1}^{R} \sum_{j=1}^{S} \left(\bar{y}_{ij}^{(0)} \right)^2 \times N_{ij}$, where $\bar{y}_{ij}^{(0)}$ is the means from the above table.

(c) Set the iteration number m = 1.

(d) Update $\bar{y}_{ij}^{(m)}, i = 1, \ldots, R,\ j = 1, \ldots, S,$ as follows:

$$\bar{y}_{ij}^{*} = \bar{y}_{ij}^{(m-1)} - \frac{\sum_{j=1}^{S} \bar{y}_{ij}^{(m-1)} \times N_{ij}}{\sum_{j=1}^{S} N_{ij}}; \tag{16}$$

$$\bar{y}_{ij}^{(m)} = \bar{y}_{ij}^{*} - \frac{\sum_{i=1}^{R} \bar{y}_{ij}^{*} \times N_{ij}}{\sum_{i=1}^{R} N_{ij}}. \tag{17}$$

(e) Compute $SS^{(m)}_{e,interaction} = \sum_{i=1}^{R}\sum_{j=1}^{S}\left(\bar{y}^{(m)}_{ij}\right)^2 \times N_{ij}$. (18)

(f) If $\left|SS^{(m)}_{e,interaction} - SS^{(m-1)}_{e,interaction}\right| < \varepsilon$ or $m \ge M$, then stop and output $SS^{(m)}_{e,interaction}$. Otherwise, set $m = m + 1$ and go back to step (d).

4. Compute the F statistic

$$F = \frac{SS_{e,interaction} \big/ df_{interaction}}{SS_{e,\,full} \big/ df_{full}}, \tag{19}$$

where $df_{interaction}$ and df_{full} are the degrees of freedom corresponding to $SS_{e,interaction}$ and $SS_{e,full}$, respectively. And $df_{interaction} = (R-1)(S-1)-\ell$, where ℓ is the number of category combinations where there are no valid record and $df_{full} = N - R - S - df_{interaction} + 1 = N - RS + \ell$. Then the F statistic follows an asymptotic F distribution with degrees of freedom $df_{interaction}$ and df_{full}.

5. Compute the p-value

$$p = 1 - \Pr\left(F_{df_{interaction},df_{full}} \le F\right). \tag{20}$$

If $p \le 0.05$, then the interaction effect $\boldsymbol{X}_1 \times \boldsymbol{X}_2$ would be included in the design matrix $\boldsymbol{X}$.

Please see Han (2010) for details.

5.2. Interaction of a covariate and a factor

Suppose a covariate is X_1 and a factor is X_2 with known S levels $(1,\ldots,S)$. similar to the method used in Section 5.1, the F statistic is computed by the following steps:

1. Create a $1 \times S$ table based on $X_2 = j,\ j = 1,\ldots,S,$ with the following statistics:

n_j : the number of distinct records;

f_{jk} : the frequency weight for the kth distinct record, $k = 1,\ldots,n_j$;

y_{jk} : the target value for the kth distinct record, $k = 1,\ldots,n_j$;

N_j : effective sample size (including frequency weights), i.e., $N_j = \sum_{i=1}^{n_j} f_{jk}$;

$\bar{X}_{1,j}$, $\bar{y}_j$: the means of X_1 and y; i.e., $\bar{X}_{1,j} = \frac{1}{N_j}\sum_{i=1}^{n_j} f_{jk} \times X_{1,jk}$ and $\bar{y}_j = \frac{1}{N_j}\sum_{i=1}^{n_j} f_{jk} \times y_{jk}$;

$C_{x_1x_1,j}$, $C_{yy,j}$: the sum of squared terms of X_1 and Y, i.e., $C_{x_1x_1,j} = \sum_{k=1}^{n_j} f_{jk}\left(X_{1,jk} - \bar{X}_{1,j}\right)^2$,

$C_{yy,j} = \sum_{k=1}^{n_j} f_{jk}\left(y_{jk} - \bar{y}_j\right)^2$;

$C_{x_1y,j}$: the sum of cross product terms of X_1 and Y, i.e., $C_{x_1y,j} = \sum_{k=1}^{n_j} f_{jk}\left(X_{1,jk} - \bar{X}_{1,j}\right)\left(y_{jk} - \bar{y}_j\right)$.

2. Compute residual sum of squares for the full model which contains two main effects $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ and the interaction effect $\boldsymbol{X}_1 \times \boldsymbol{X}_2$:

$$SS_{e,\,full} = \sum_{j=1}^{S} C_{yy,j} - \sum_{j=1}^{S} \frac{\left(C_{x_1y,j}\right)^2}{C_{x_1x_1,j}}. \tag{21}$$

3. Compute residual sum of squares for the main effects model which contains two main effects only:

$$SS_{e,main} = \sum_{j=1}^{S} C_{yy,j} - \frac{\left(\sum_{j=1}^{S} C_{x_1y,j}\right)^2}{\sum_{j=1}^{S} C_{x_1x_1,j}}. \tag{22}$$

4. Compute the F statistic

$$F = \frac{\left(SS_{e,main} - SS_{e,\,full}\right)\big/\left(S-1\right)}{SS_{e,\,full}\big/\left(N-2S\right)}, \tag{23}$$

and it follows an asymptotic F distribution with degrees of freedom $S-1$ and $N-2S$.

5. Compute the p-value

$$p = 1 - \Pr\left(F_{S-1,N-2S} \le F\right). \tag{24}$$

If $p \le 0.05,$ then the interaction effect $\boldsymbol{X}_1 \times \boldsymbol{X}_2$ would be included in the design matrix $\boldsymbol{X}$.

Please see Zheng (2010) for details.

Implementation notes:

- All statistics are computed in map/reduce environment, see Section A.4 in Appendix A of this chapter for details.
- If there is no valid record in any category then adjust *S* value accordingly.
- Regression weights will not be used even it is specified.
- If $SS_{e,full} = 0$, then *F* statistics is assigned as sysmis and the *p* value for *F* statistic is as follows:

$$p\text{-value} = \begin{cases} 0 & \text{if } \left(SS_{e,\text{main}} - SS_{e,full}\right) \neq 0 \\ \text{sysmis} & \text{if } \left(SS_{e,\text{main}} - SS_{e,full}\right) = 0, \end{cases}$$

  If $p\text{-value} = \text{sysmis}$, then the interaction effect $\boldsymbol{X}_1 \times \boldsymbol{X}_2$ would NOT be included in the design matrix $\boldsymbol{X}$.

  Please note that we will treat $SS_{e,full} = 0$ or $\left(SS_{e,\text{main}} - SS_{e,full}\right) = 0$ when two criteria are met: $SS_{e,full} \le \varepsilon^* \times SS_t \times p^*$ and $SS_{e,full} \le 1.0e-8$ or $\left(SS_{e,\text{main}} - SS_{e,full}\right) \le \varepsilon^* \times SS_t \times p^*$ and $\left(SS_{e,\text{main}} - SS_{e,full}\right) \le 1.0e-8$, respectively, where ε^* is machine epsilon (about 2.2*e*-16), p^* is the number of non-redundant estimated parameters for the full model and $p^* = RS - \ell - 1$ here, SS_t is total sum of squares for the target and it can be computed by $SS_t = \sum_{i=1}^{n} f_i (y_i - \overline{y})^2$ and please see Section A.2 in Appendix A on how to compute it in map/reduce environment.

- Regarding Eq. (5), if $C_{x_1x_1,j} = 0$, then the item $\left(C_{x_1y,j}\right)^2 / C_{x_1x_1,j}$ is set to 0.
- Regarding Eq. (6), if $\sum_{j=1}^{S} C_{x_1x_1,j} = 0$, then $SS_{e,main}$ is set to missing. That is to say, the interaction of X_1 and X_2 is not significant.

5.3. Two-way-test pair search strategy

Suppose there are m main effects (factors and covariates), the number of parameters for them is p^m (excluding the intercept) and the number parameters for covariates is p^{cm}.

Input values (integers) for m_1 (threshold value to conduct interaction effect detection; the default is 100), m_2 (threshold value to select main effects for interaction effect detection; the default is 50) and $p^{\max}$ (maximum number of parameters the system can handle; the default is 5000), where $m_1 \geq m_2$.

When ($p^{\max} > p^m + p^{cm}$) and $m > m_2$, then the strategy will be conducted with the following steps:

1. Build a linear model using all main effects $\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_m$ using the sweep operation method described in Section 4.

2. Select the significant main effects ($p < 0.05$) based on tests of model effects based on Wald test. Assume there are m' significant effects.

3. If ($m' < 2$) or ($m' > m_1$), then stop and no interaction detection is conducted. Otherwise, sort the main effects using *p*-value in ascending order.

4. Select the top $m''(= \min(m', m_2))$ main effects to construct two-way interaction effects (of two factors, and one covariate and one factor) among these m'' main effects.

5. Test all candidate interaction effects using the methods given Sections 5.1 and 5.2.

6. Calculate the total number of parameters for all significant interaction effects, denoted by p^{inter}, if $p^{inter} < (p^{\max} - p^m - p^{cm})$, then output all significant interaction effects and stop; otherwise go to step 7.

7. Calculate effect size for each significant two-way interaction effect

$$\eta^2 = \frac{SS_{e,interaction}}{SS_t} \quad (25)$$

   where $SS_{e,interaction}$ denotes to be difference of weighted residual sum of squares between the full model and the main effects only model, please see Sections 5.1 and 5.2 for formulae and SS_t denotes the weighted sum of squares for the target, please see Section 5.1 for formula.

8. Sort all significant two-way interaction effects using their effect sizes in descending order and select and output top-k interaction effects, where k is the maximum number satisfying the number of parameters for top-k interaction effects is less than or equal to ($p^{\max} - p^m - p^{cm}$).

When $(p^{\max} > p^{m} + \ p^{\,cm})$ and $m \leq m_2$, the strategy will be similar to the one given above, except that the step of constructing two-way interaction effects: here, two-way interaction effects are directly constructed among all m main effects rather than based on m'' significant main effects.

After the strategy is conducted, the total number of parameters of candidate effects for model selection methods is smaller than a threshold value and the candidate effects will include (1) all main effects; (2) square terms of all covariates; (3) some (or all) significant interaction effects.

6. Model Selection

For the small to median p situations $(p^{\max} \geq p)$ which means the $\boldsymbol{R}$ matrix can be fit into memory or, we will support three model selection methods: (1) none; (2) forward stepwise; and (3) best subsets.

For the large p situations $(p^{\max} < p)$, we will utilize ADMM to conduct model selection which is similar to LASSO and details would be provided later.

6.1. None

No selection method is used.

6.2. Forward stepwise

The basic idea of the forward stepwise method is to start off by choosing the best effect in addition to the intercept if exists and then tries to enter additional effect one at a time as long as these additions are worthy. After an effect has been added, all effects in the current model are checked to see if any of them should be removed. Then the process continues until a stop criterion is met.

The traditional criterion for effect entry and removal is based on the *F*-statistics, which their corresponding p-values are used to compare with some specified entry and removal significance levels, but they do not follow an *F* distribution so the results might be questionable. Hence three additional criteria for effect entry and removal are offered: (1) maximum adj. R^2; (2) minimum corrected Akaike information criterion (AICC); and (3) minimum average square error (ASE) over the overfit prevention data.

How to calculate these statistics and the selection process are described in details below.

6.2.1. Candidate statistics

Some clearer notations are needed to calculate the following 4 statistics for a continuous effect X_j or categorical effect $\{X_{j_s}\}_{s=1}^{\ell}$ entering to and removing from the current model as follows in each step.

ℓ^*	The number of non-redundant parameters of the eligible effect X_j or $\{X_{j_s}\}_{s=1}^{\ell}$.
p^c	The number of non-redundant parameters in the current model (including an intercept if exists).
p^r	The number of non-redundant parameters in the resulting model (including the intercept if exists). Note that $p^r = \begin{cases} p^c + \ell^* & \text{for entering an effect} \\ p^c - \ell^* & \text{for removing an effect} \end{cases}$
SSe_p	The weighted residual sum of squares for the current model.
$SSe_{p+\ell}$	The weighted residual sum of squares for the resulting model after entering the effect.
$SSe_{p-\ell}$	The weighted residual sum of squares for the resulting model after removing the effect.
r_{yy}	The last diagonal element in the current $\boldsymbol{R}$ matrix.
$\tilde{r}_{yy}$	The last diagonal element in the resulting $\tilde{\boldsymbol{R}}$ matrix.

(1) ***F*-statistics**:

The *F*-statistics are different for an effect X_j or $\{X_{j_s}\}_{s=1}^{\ell}$ entering to and removing from the current model as follows:

$$F_{enter_j} = \frac{(SSe_p - SSe_{p+\ell})/\ell^*}{SSe_{p+\ell}/(N-p^r)} = \frac{(r_{yy} - \tilde{r}_{yy})(N-p^r)}{\tilde{r}_{yy} \times \ell^*} \text{ and} \tag{26}$$

$$F_{remove_j} = \frac{\left(SSe_{p-\ell} - SSe_p\right)/\ell^*}{SSe_p/\left(N-p^c\right)} = \frac{\left(\tilde{r}_{yy} - r_{yy}\right)\left(N-p^c\right)}{r_{yy} \times \ell^*}, \text{ respectively.} \tag{27}$$

Then the p-values corresponding to the above *F*-statistics are

$$p_{enter_j} = P\left(F_{\ell^*,N-p^r} \geq F_{enter_j}\right) = 1 - P\left(F_{\ell^*,N-p^r} \leq F_{enter_j}\right) \text{ and} \tag{28}$$

$$p_{remove_j} = P\left(F_{\ell^*,N-p^c} \geq F_{remove_j}\right) = 1 - P\left(F_{\ell^*,N-p^c} \leq F_{remove_j}\right), \text{ respectively.} \qquad (29)$$

(2) **adj. R^2**:

The adj. R^2 value for the resulting model when an effect entering to or removing from the current model is as follows:

$$\text{adj. } R^2 = \begin{cases} 1 - \dfrac{\left(N-1\right)\tilde{r}_{yy}}{N-p^r} & \text{if there is an intercept} \\ 1 - \dfrac{N \times \tilde{r}_{yy}}{N-p^r} & \text{if there is no intercept} \end{cases} \qquad (30)$$

(3) **AICC**:

The AICC value for the resulting model when an effect entering to or removing from the current model is as follows:

$$AICC = N \ln\left(\frac{(N-1)S_{yy} \times \tilde{r}_{yy}}{N}\right) + \frac{2p^r N}{N-p^r-1} \qquad (31)$$

(4) **ASE**:

The ASE value over the overfit prevention data for the resulting model when an effect entering to or removing from the current model is as follows:

$$ASE = \frac{1}{\sum_{t=1}^{T} f_t} \sum_{t=1}^{T} w_t \left(y_t - \hat{y}_t\right)^2 \qquad (32)$$

where $\hat{y}_t = \boldsymbol{x}_t\hat{\boldsymbol{\beta}}$ is the predicted values of y_t and T is the number of distinct testing cases in the testing (overfit prevention) data.

6.2.2. The selection process

The nature of *F*-statistics criterion is different from the other three criteria. The *F*-statistics criterion is to select an effect for entry (removal) with the minimum (maximum) p-value and continue doing it until the p-values of all candidates for entry (removal) are equal to or greater than (less than) a specified significance level. The other three criteria are to compare the statistic (adj. R^2, AICC or ASE) of the resulting model after entering (removing) an effect with that of the current model and selection would be stopped at a local optimal value (a maximum for the adj. R^2 criterion but a minimum for the AICC criterion and ASE criterion). Hence the following selection process is described in terms of the *F*-statistics criterion (denoted as FC) and AICC criterion (denoted as AC), then it should be easy to change from AICC criterion to the other two criteria).

Some definitions are needed for the selection process:

FLAG	A $p^e \times 1$ index vector which records the status of each effect. $FLAG_i$ = 1 means the effect *i* is in the current model, $FLAG_i$ = 0 means it is not. Note that $\left|\{i \mid FLAG_i = 1\}\right|$ denotes the number of effects with $FLAG_i$ = 1.
MAXSTEP	The maximum number of iteration steps. The tentative default value is $3 \times p^e$.
MAXEFFECT	The maximum number of effects (excluding intercept if exists). The default value is p^e.
P_{in}	The significance level for effect entry when *F*-statistics criterion is used. The default is 0.05.
P_{out}	The significance level for effect removal when *F*-statistics criterion is used. The default is 0.1.
ΔF	The *F*-statistic change. It is F_{enter_j} or F_{remove_j} for entering or removing an effect X_j (here X_j could represent continuous or categorical for simpler notation).
$p_{\Delta F}$	The corresponding p-value for ΔF .
$AICC_{current}$	The AICC value for the current model.

(1) Set $\{FLAG_i\}_{i=1}^{p^e} = 0$ and *iter* = 0. If there is an intercept, the initial model is $\hat{y} = \bar{y}$, otherwise it is $\hat{y} = 0$. If AC (AICC criterion, similarly for other two criteria) is used, compute AICC for the initial model and denote it as $AICC_{current}$.

(2) If $\{i \mid FLAG_i = 0\} \neq \phi$, *iter* < *MAXSTEP* and $\left|\{i \mid FLAG_i = 1\}\right| < MAXEFFECT$, go to next step (3); otherwise stop and output the model .

(3) Based on the current model, for every effect *j* eligible for entry (see Condition below) ,

if FC (F-statistics criterion) is used, compute F_{enter_j} and p_{enter_j} ;

if AC is used, compute $AICC_j$.

(4) If FC is used, choose the effect $X_{j^*}, j^* = \arg\min_j \left\{ p_{enter_j} \right\}$ and if $p_{enter_{j^*}} < P_{in}$, enter X_{j^*} to the current model.

IF AC is used, choose the effect $X_{j^*}, j^* = \arg\min_j \left\{ AICC_j \right\}$ and if $AICC_{j^*} < AICC_{current}$, enter X_{j^*} to the current model.

(Note that for adj. R^2 criterion, $j^* = \arg\max_j \left\{ \text{adj. } R_j^2 \right\}$ and $\text{adj. } R_{j^*}^2 >$ $\text{adj. } R_{current}^2$.)

Then go to (5); otherwise stop and output the current model.

(5) If the model with new effect is the same as any previous ones, stop and output the current model; otherwise update the current model by doing sweep operation on corresponding row(s) and column(s) associated with X_{j^*} in the current ***R*** matrix. Set $FLAG_{j^*} = 1$ and $iter = iter + 1$.

If FC is used, let $\Delta F = F_{enter_{j^*}}$ and $p_{\Delta F} = p_{enter_{j^*}}$;

if AC is used, let $AICC_{current} = AICC_{j^*}$.

(6) For every effect k in the current model (i.e., $FLAG_k = 1, \ \forall k$),

if FC is used, compute F_{remove_k} and p_{remove_k} ;

if AC is used, compute $AICC_k$.

(7) If FC is used, choose the effect X_{k^*}, $k^* = \arg\max_k \left\{ p_{remove_k} \right\}$ and if $p_{remove_{k^*}} > P_{out}$, remove X_{k^*} from the current model.

If AC is used, choose the effect X_{k^*}, $k^* = \arg\min_k \left\{ AICC_k \right\}$ and if $AICC_{k^*} < AICC_{current}$, remove X_{k^*} from the current model.

(Note that for adj. R^2 criterion, $k^* = \arg\max_k \left\{\text{adj. } R_k^2\right\}$ and adj. $R_{k^*}^2$ > adj. $R_{current}^2$.)

Then go to (8); otherwise go back to (2).

(8) If the model with the effect removed is the same as any previous one, stop and output the current model; otherwise update the current model by doing sweep operation on corresponding row(s) and column(s) associated with X_{k^*} in the current ***R*** matrix. Set $FLAG_{k^*} = 0$ and *iter* = *iter* + 1.

If FC is used, let $\Delta F = F_{remove_{k^*}}$ and $p_{\Delta F} = p_{remove_{k^*}}$;

if AC is used, let $AICC_{current} = AICC_{k^*}$. Then go back to (6).

Condition

Eligible for entry conditions for the effect *j*: (i.e., $FLAG_j = 0, \ \forall j$)

a) For continuous effect X_j, $r_{jj} \geq t$ (singularity tolerance *t* with a default of 1e-4);
For categorical effect $\{X_{j_s}\}_{s=1}^{\ell}$, $\max\{r_{j_1 j_1}, r_{j_2 j_2}, \ldots, r_{j_\ell j_\ell}\} \geq t$.

Note that here r_{jj} and $r_{j_s j_s}$, $s = 1, \ldots, \ell$, are diagonal elements in the current ***R*** matrix (before entering).

b) For each continuous effect X_k that is currently in the model, $\tilde{r}_{kk} t \leq 1$.
For each categorical effect $\{X_{k_s}\}_{s=1}^{\ell'}$ with ℓ' levels that is currently in the model, $\max\{\tilde{r}_{k_1 k_1}, \tilde{r}_{k_2 k_2}, \ldots, \tilde{r}_{k_{\ell'} k_{\ell'}}\} t \leq 1$.

Note that here $\tilde{r}_{kk}$ and $\tilde{r}_{k_s k_s}$, $s = 1, \ldots, \ell'$, are diagonal elements in the resulting ***R*** matrix, i.e., the results after doing sweep operation on corresponding row(s) and column(s) associated with X_k or $\{X_{k_s}\}_{s=1}^{\ell'}$ in the current ***R*** matrix.

The above condition is imposed so that entry of the effect does not reduce the tolerance of other effects already in the model to unacceptable levels.

Rules for entering or removing effects when interaction effects are present:

1. **NONE**

No requirement need be satisfied for any effects in the model.

2. **SINGLE** (default)

Hierarchy requirement is satisfied for all effects in the model. It stipulates that, for any effect to be in a model, all lower-order effects that are part of the former effect must also be in the model. For example, given A, X, and A*X, then for A*X to be in a model, the effects A and X must also be in the model.

3. **SFACTOR**

Hierarchy requirement is satisfied for all factor-only effects in the model.

4. **CONTAINMENT**

Containment requirement is satisfied for all effects in the model. It stipulates that, for any effect to be in the model, all effects contained in the former effect must also be in the model.

The meaning of containment is that, for any two effects *F* and *F*', *F* is contained in *F'*, if:

- Both effects *F* and *F*' involve the same covariate effect, if any. (Note that effects *A**X* and *A**X**X* are not considered to involve the same covariate effect because the first involves covariate effect X and the second involves covariate effect *X***2.)†
- *F'* consists of more factors than *F*.
- All factors in *F* also appear in *F'*.

The following table illustrates how the hierarchy and containment requirements mean for effect entering and removing. The cells contain the order in which effects must occur in the model.

Effects	**SINGLE**	**SFACTOR**	**CONTAINMENT**
*A,B,A*B*	1. *A, B* 2. *A*B*	1. *A, B* 2. *A*B*	1. *A, B* 2. *A*B*
X, *X***2, *X***3	1. *X* 2. *X***2 3. *X***3	Effects can occur in the model in any order.	Effects can occur in the model in any order.
A, *X*, *X*(*A*)	1. *A*, *X* 2. *X*(*A*)	Effects can occur in the model in any order.	1. *X* 2. *X*(*A*) Effect A can occur in the model in any order.
A, *X*, *X**2(A)*	1. *A*, *X* 2. *X**2(A)*	Effects can occur in the model in any order.	Effects can occur in the model in any order.

† *A* *B* are factors and *X* is covariate effect.

‡ The intercept effect is contained in all the pure factor effect. However it is not contained in any effect involving a covariate. No effect is contained in the intercept effect.

£ This is an important definition, since all type II, type III and Type IV estimable functions rely on this definition.

§ In the implementing, it is useful to store "hierarchy" and "contained" information for each effect in order to define the order of sweeping and calculation of the Type III sums of square.

6.3. Best subsets (will update the relevant default values later)

Stepwise method based on adding or removing effect one at a time with respect to some criterion is a method to do model selection. However, these methods search fewer combinations of sub-models and rarely select the best one, so to select the best one according to some criterion is to check all possible models. The available criteria are (1) maximum adj. R^2; (2) minimum AICC; and (3) minimum ASE over the overfit prevention data.

Since there are p^e free effects no matter whether there is an intercept in the model or not, we do exhaustive search over 2^{p^e} models, which include intercept-only model ($\hat{\boldsymbol{y}} = \overline{\boldsymbol{y}}$) if there is an intercept, or the null model ($\hat{\boldsymbol{y}} = \mathbf{0}$) otherwise. Because the number of calculations increases exponentially with p^e, it is important to have an efficient algorithm for carrying out the necessary computations. However, if p^e is too large, it may not be practical to check all of the possible models.

We divide the problem into 2 tiers in terms of the number of effects: (1) when $p^e \le 20$, we do all-possible-subset search; (2) when $p^e > 20$, we apply a hybrid method which combines forward stepwise method and all possible subset method.

6.3.1. All possible subset method for the first tier problem

If $p^e \le 20$, we do exhaustive search of all the possible sub-models.

An efficient method, which was proposed by Schatzoff (1968) and would have the minimum number of sweep operations on the ***R*** matrix, is applied to traverse all the models and outlined as follows:

Each sweep step(s) on one effect results a model. So 2^{p^e} models can be obtained through a sequence of exactly 2^{p^e} sweeps on effects. Assume that the all possible models on $p^e - 1$ effects can be obtained in a sequence S_{p^e-1} of exactly 2^{p^e-1} sweeps on the first $p^e - 1$ pivotal effects. And sweeping on the last effect will produce a new model which adds the last effect to the model produced by the sequence S_{p^e-1}. Then repetition of the sequence S_{p^e-1} will produce another 2^{p^e-1} distinct models (including the last effect). It is a recursive algorithm for constructing the sequence S_{p^e}, i.e.,

$$S_{p^e} = \left(\underline{S_{p^e-1}}, k, \underline{S_{p^e-1}}\right) = \left(\underline{S_{p^e-2}, k-1, S_{p^e-2}}, k, \underline{S_{p^e-2}, k-1, S_{p^e-2}}\right) = \ldots, \text{etc.}$$

The sequence of model produced is demonstrated in the following table:

k	S_k^*	Sequence of models produced
0	0	Only intercept
1	1	(1)**
2	121	(1),(12),(2)
3	1213121	(1),(12),(2),(23),(123),(13),(3)
4	121312141213121	(1),(12),(2),(23),(123),(13),(3),(34),(134),(1234),(234),(24),(124),(14),(4)
…	…	…
p^e	S_{p^e-1}, p^e, S_{p^e-1}	All 2^{p^e} models including the intercept model.

* The indexes of effects which are pivoted on.

** Each parenthesis in the third column represents a regression model. The numbers in the parentheses indicate the effects which are included in that model.

6.3.2. Hybrid method for the second tier problem

If $p^e > 20$, we apply a hybrid method by combining the forward stepwise method with possibly all possible subset method as follows:

(1) Select the effects by the stepwise method (note that the same criterion used to select the best model is also used in the forward stepwise and see Section 6.2.2 for details). Assume p^s is the number of these effects.

(2) Apply different approaches, depending on the value of p^s, as follows:

(a) If $p^s \le 20$, do exhaustive search of all possible subsets on these selected effects via the method in Section 6.3.1.

(b) If $20 < p^s \le 40$, select $p^s - 20$ effects based on the p-values of type III sum of squares tests from all p^s effects (see Section 7.2) and enter them to the model as a constant part, then do exhaustive search of all remaining 20 effects via the method in Section 6.3.1, i.e., do exhaustive search on remaining 20 effects with $p^s - 20$ effects always in the model.

(c) If $40 < p^s$, do nothing and assume the best model is the one with these p^s effects (with a warning message that the selected model is based on the forward stepwise method).

7. Model and Predictor Summary

7.1. Coefficients and statistical inference

After the model selection process, we can get the coefficients and related statistics in or not in from the swept correlation matrix. The following statistics are computed for each effect in the model or not in.

All the effects with $FLAG_{\boldsymbol{j}}$ =1 are currently in the model, as well as intercept (if exists). We calculate these below base on the $\tilde{\boldsymbol{R}}$ matrix.

- **Unstandardized coefficient estimates**

$$\hat{\beta}_j = b_j \sqrt{\frac{S_{yy}}{S_{jj}}} = \tilde{r}_{jy} \sqrt{\frac{S_{yy}}{S_{jj}}} \qquad j = 1, \cdots, p^*, \tag{33}$$

The redundant coefficient estimates are set to zero.

- **Standard errors of regression coefficients**

The standard error of $\hat{\beta}_j$ is

$$\hat{\sigma}_{\hat{\beta}_j} = \sqrt{\operatorname{var}(\hat{\beta}_j)} = \sqrt{\frac{\tilde{r}_{jj}\tilde{r}_{yy}S_{yy}}{S_{jj}df_e}} = \sqrt{\frac{\tilde{r}_{jj} \times SS_e}{S_{jj} \times \left(N-1\right) df_e}} \qquad (34)$$

- **Intercept estimation**

If the model includes an intercept, the intercept is estimated by all other parameters in the model as

$$\hat{\beta}_0 = \bar{y} - \sum_{j=1}^{p} \hat{\beta}_j \bar{X}_j \qquad (35)$$

The standard error of $\hat{\beta}_0$ is estimated by

$$\hat{\sigma}_{\hat{\beta}_0} = \sqrt{\hat{\sigma}^2_{\hat{\beta}_0}} \quad \text{with} \quad \begin{aligned} \hat{\sigma}^2_{\hat{\beta}_0} &= \frac{(N-1)\tilde{r}_{yy}S_{yy}}{N(N-p^*-1)} + \sum_{j=1}^{p} \bar{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \bar{X}_k \bar{X}_j \operatorname{cov}\left(\hat{\beta}_k, \hat{\beta}_j\right) \\ &= \frac{SS_e}{N \times df_e} + \sum_{j=1}^{p} \bar{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \bar{X}_k \bar{X}_j \frac{\tilde{r}_{kj} \times SS_e}{\sqrt{S_{kk}S_{jj}} \times \left(N-1\right) df_e}. \end{aligned} \qquad (36)$$

- **t-statistics for regression coefficients**

t - statistic for $\hat{\beta}_j$ is

$$t = \frac{\hat{\beta}_j}{\hat{\sigma}_{\hat{\beta}_j}} = \tilde{r}_{jy}\sqrt{\frac{df_e}{\tilde{r}_{yy}\tilde{r}_{jj}}} \quad , \ j = 0,1,\ldots, p, \qquad (37)$$

and it follows an asymptotic *t* distribution with the degree of freedom df_e. Then the *p*-value is computed as

$$p = 2 \times \left(1 - prob\left(t_{df_e} \le \left|t\right|\right)\right). \qquad (38)$$

- **100($1-\alpha$)% confidence internals**

$$\hat{\beta}_j \pm \hat{\sigma}_{\hat{\beta}_j} \times t_{\alpha/2, df_e}. \qquad (39)$$

7.2. ANOVA (Tests of model effects)

- **Weighted total sum of squares (*SS_t*)**

$$SS_t = \begin{cases} \sum_{i=1}^{n} w_i (y_i - \overline{y})^2 = (N-1)S_{yy} \text{ with d.f.} = df_t = N-1 \text{ if there is an intercept} \\ \sum_{i=1}^{n} w_i {y_i}^2 = (N-1)S_{yy} \quad \text{with d.f.} = df_t = N \quad \text{if there is no intercept} \end{cases}, \quad (40)$$

where d.f. means degrees of freedom. It is called "SS (sum of squares) for Corrected Total" if there is an intercept or "SS for Total" if there is no intercept.

- **Weighted residual sum of squares (*SS_e*)**

$$SS_e = \sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2 = \tilde{r}_{yy}(N-1)S_{yy} \quad \text{with d.f.} = df_e = N - p^c. \qquad (41)$$

It is also called "SS for Error".

- **Weighted regression sum of squares (*SS_r*)**

$$SS_r = \begin{cases} \sum_{i=1}^{n} w_i (\hat{y}_i - \overline{y})^2 = (1-\tilde{r}_{yy})(N-1)S_{yy} = SS_t - SS_e \text{ if there is an intercept} \\ \sum_{i=1}^{n} w_i {\hat{y}_i}^2 = (1-\tilde{r}_{yy})(N-1)S_{yy} = SS_t - SS_e \quad \text{if there is no intercept} \end{cases}, \quad (42)$$

with d.f. = $df_r = p^*$. It is called "SS for Corrected Model" if there is an intercept, or "SS for Model" if there is no intercept.

- **Regression mean square error**

$$SS_r / df_r . \qquad (43)$$

- **Residual mean square error**

$$SS_e / df_e . \qquad (44)$$

- **F statistic for corrected model**

$$F = \frac{SS_r / df_r}{SS_e / df_e} = \frac{SS_r \cdot df_e}{SS_e \cdot df_r}, \tag{45}$$

which follows an F distribution with degrees of freedom df_r and df_e , and the corresponding p-value can be calculated accordingly.

- **Type III sum of squares for each effect**

To compute type III SS for the effect j, $j = 1, \dots, p^e$, type III test matrix $\boldsymbol{L}_i$ needs to be constructed first. Construction of matrix $\boldsymbol{L}_i$ is based on the generating matrix $\boldsymbol{H}_\omega = \left(\boldsymbol{X}^\mathrm{T}\boldsymbol{D}\boldsymbol{X}\right)^- \boldsymbol{X}^\mathrm{T}\boldsymbol{D}\boldsymbol{X}$, where $\boldsymbol{D} = \mathrm{diag}(g_1, \dots, g_n)$, such that $\boldsymbol{L}_i\boldsymbol{\beta}$ is estimable. It involves parameters only for the given effect and the effects containing the given effect. For type III analysis, $\boldsymbol{L}_i$ doesn't depend on the order of effects specified in the model (but it does for type I analysis). If such a matrix cannot be constructed, the effect is not testable. See Chiu (1995a, b) for computational details on construction of type III test matrices.

For each effect j, type III SS is calculated as follows

$$\boldsymbol{S}_j = \hat{\boldsymbol{\beta}}^\mathrm{T}\boldsymbol{L}_j^\mathrm{T}(\boldsymbol{L}_j\boldsymbol{G}\boldsymbol{L}_j^\mathrm{T})^{-1}\boldsymbol{L}_j\hat{\boldsymbol{\beta}} \tag{46}$$

where $\boldsymbol{G} = (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^-$.

Implementation notes:

- The $\boldsymbol{X}$ matrix in $\boldsymbol{G}$ only includes the effects selected into the final model, so does $\tilde{\boldsymbol{R}}$. Obtain $\boldsymbol{G}$ from $\tilde{\boldsymbol{R}}_{11}$ (the upper left-hand $p \times p$ block matrix of $\tilde{\boldsymbol{R}}$) as follows:

$$\boldsymbol{G} = \begin{cases} \begin{bmatrix} 1 & \boldsymbol{A}_{01} \\ \boldsymbol{0} & \boldsymbol{A}_{11} \end{bmatrix} \begin{bmatrix} 1/W_n & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{R}}_{11} \end{bmatrix} \begin{bmatrix} 1 & \boldsymbol{A}_{01} \\ \boldsymbol{0} & \boldsymbol{A}_{11} \end{bmatrix}^\mathrm{T} & \text{if there is an intercept} \\ \boldsymbol{A}_{11}\tilde{\boldsymbol{R}}_{11}\boldsymbol{A}_{11} & \text{if there is no intercept} \end{cases} \tag{47}$$

where $W_n = \sum_{k=1}^{n} w_k$, $\boldsymbol{A}_{11} = \mathrm{diag}\left(1\big/\sqrt{(N-1)S_{11}}, \dots, 1\big/\sqrt{(N-1)S_{pp}}\right)$, and

$\boldsymbol{A}_{01} = -\bar{\boldsymbol{X}}^\mathrm{T}\boldsymbol{A}_{11} = -\left[\bar{X}_1, \dots, \bar{X}_p\right]\boldsymbol{A}_{11}$. Note that $\bar{X}_i$ and S_{ii}, $i = 1, \dots, p$, are weighted sample mean and variance for $\boldsymbol{X}_i$, respectively; and p denotes the number of parameters (excluding intercept) in the final model.

After some algebra, G can be expressed as follows if there is an intercept

$$\boldsymbol{G} = \begin{bmatrix} 1/W_n + \bar{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{A}_{11} \tilde{\boldsymbol{R}}_{11} \boldsymbol{A}_{11} \bar{\boldsymbol{X}} & -\bar{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{A}_{11} \tilde{\boldsymbol{R}}_{11} \boldsymbol{A}_{11} \\ -\boldsymbol{A}_{11} \tilde{\boldsymbol{R}}_{11} \boldsymbol{A}_{11} \bar{\boldsymbol{X}} & \boldsymbol{A}_{11} \tilde{\boldsymbol{R}}_{11} \boldsymbol{A}_{11} \end{bmatrix}. \tag{48}$$

- Obtain $\boldsymbol{S}_j$ by sweeping the following matrix

$$\begin{bmatrix} -\boldsymbol{L}_j \boldsymbol{G} \boldsymbol{L}_j^{\mathrm{T}} & \boldsymbol{L}_j \hat{\boldsymbol{\beta}} \\ (\boldsymbol{L}_j \hat{\boldsymbol{\beta}})^{\mathrm{T}} & 0 \end{bmatrix}, \tag{49}$$

then the last diagonal element of the resulting matrix corresponds to $\boldsymbol{S}_j$.

- **F statistic for each effect**

The above SS for the effect j is also used to compute the F statistic for hypothesis test H_0: $\boldsymbol{L}_j\boldsymbol{\beta} = \mathbf{0}$ as follows:

$$F_j = \frac{\boldsymbol{S}_j / r_j}{SS_e / df_e} \tag{50}$$

where r_j is the full row rank of $\boldsymbol{L}_i$. It follows an F distribution with degrees of freedom r_i and df_e, then the p-values can be calculated accordingly.

Note that the parameter estimate covariance matrix is used in the above F statistic implicitly as it is $SS_e / df_e \times \boldsymbol{G}$.

7.3. Model quality measures

The squared multiple correlation coefficient (R square) or coefficient of determination is to measure of how much of the variation in the data is explained by the model. It denoted by R^2, is expressed as

$$R^2 = \frac{SS_r}{SS_t} = 1 - \frac{SS_e}{SS_t} = 1 - \tilde{r}_{yy}.$$

- **Adjusted R square**

$$\text{adj. } R^2 = 1 - \frac{SS_e / df_e}{SS_t / df_t} = R^2 - \frac{(1-R^2)p^*}{df_e} = 1 - \frac{df_t \times \tilde{r}_{yy}}{df_e}. \tag{51}$$

- **Corrected Akaike information criterion (AICC)**

$$AICC = N \ln\left(\frac{SS_e}{N}\right) + \frac{2p^c N}{N - p^c - 1}. \tag{52}$$

7.4. Predictor importance (PI)

The predictor importance computation for all modelling engines would be based on “variance-based sensitivity analysis” which is a model free method and has been used many models in Modeler, such as Regression (older version of ALM), Tree, Logistic regression, genlin, etc. (see Zhong (2008) and Xu (2011) for details).

7.5. EMMEANS

The EMMEANS for significant effects would be computed and compared based on some contrasts. Please see Zheng (2009) for details.

However, the contrast types and adjustment methods would be determined later.

7.6. Grouping and unusual category detection

For a significant factor or factor interaction from the ANOVA table, some categories or category combinations must have statistically significant impact on the target and we can partition them into high and low groups. The following process is used to find the high and low groups and the possible middle group among all categories of a significant factor with at least 3 categories. Note that grouping and unusual category detection analyses would not be conducted for any insignificant factors or factor interactions, i.e., their *p*-values are larger than the significance level (including sysmis); and the description is for a significant factor, but it should be applied to a factor interaction similarly.

1) For a significant factor with m categories, $C_1,\cdots,C_m$, compute the EMMENS, $EM_1,\cdots,EM_m$ and their corresponding standard errors, $\hat{\sigma}_{EM_1},\ldots,\hat{\sigma}_{EM_m}$.

2) Sort the EMMEANS by a descending order. Without loss of generality, assume they are $EM_1,\cdots,EM_m$ so EM_1 has the largest EMMEAN and EM_m has the smallest EMMEAN.

3) At first, the category with largest EMMEAN is formed the high group. Then test if there is a difference between the second largest EMMEAS and the largest EMMEAN. The test statistic is similar to the pairwise contrast statistics described in Zheng (2009),

$$t_1 = \frac{EM_1 - EM_2}{\hat{\sigma}_{(EM_1-EM_2)}} \tag{53}$$

where $\hat{\sigma}_{(EM_1-EM_2)}$ is the corresponding standard error for $EM_1 - EM_2$. It has an asymptotic t distribution with df_e $(= N - p^c)$ degrees of freedom. The corresponding p-value could be computed as follows:

$$p = 1 - prob(t_{df_e} \le |t_1|) \tag{54}$$

If the null hypothesis is not rejected, i.e., the p-value > α (significance level specified by user, default is 0.05), then the category with the second largest EMMEAN will be added to the high group.

Implementation notes:

(1) If $EM_1 - EM_2 = 0$, then there is no need to compute $\hat{\sigma}_{(EM_1-EM_2)}$ and assign $p = 1$, i.e., the category with the second largest EMMEAN will be added to the high group.

(2) If $EM_1 - EM_2 \neq 0$ and $\hat{\sigma}_{(EM_1-EM_2)} = 0$, then $p = 0$ and stops. Please note that

$\hat{\sigma}_{(EM_1-EM_2)} = 0$ should only happen when there is a perfect fit, i.e., $SS_e < \varepsilon^* \times SS_t \times p^*$ and $SS_e < 1.0e-8$, where ε^*, SS_t and p^* are defined in Section 7.2.

4) Repeat the same process for the next EMMEANS in line, i.e., compare EM_3 with EM_1, compare EM_4 with EM_1, etc. until there is no category can be added into the high group.

5) Similarly, form the low group from the smallest EMMEAN for those categories not assigned to the high group.

6) If there still exist some categories after forming the high and low groups, they are grouped into the middle group.

Furthermore, there might exist few categories or category combinations with extremely high or low EMMEANS in the high or low group. We call such categories or category combinations "unusual categories". The process to detect those "unusual categories" for a significant factor is described as follows:

First, suppose there are m categories, $C_1, \cdots, C_m$, for a significant and corresponding EMMEANS are $EM_1, \cdots EM_m$ respectively. The number of records in $C_1, \cdots, C_m$ are $n_1, \cdots, n_m$, respectively. Then the unusual category detection process is described as follows:

(1) Find the median of m EMMEANS, incorporating the number of records in each category (suppose The number of records in $C_1, \cdots, C_m$ are $n_1, \cdots, n_m$, respectively). Denote the median as M, then $M = median(EM_{1,n_1}, \cdots, EM_{m,n_m})$, where EM_{i,n_i} is a set which contains only EM_i value with n_i of them.

(2) Compute the median absolute deviation (MAD) of m EMMEANS, again incorporating with the number of records in each category

$$MAD = median(|EM_1 - M|_{n_1}, \cdots, |EM_m - M|_{n_m}), \qquad (55)$$

where $| EM_i - M |_{n_i}$ is a set which contains only $| EM_i - M |$ value with but the n_i of them.

(3) Compute the modified z-score for each C_i

$$z_i = \begin{cases} \dfrac{EM_i - M}{1.4826 \times MAD} & \text{if } MAD \neq 0 \\ \dfrac{EM_i - M}{1.253314 \times MeanAD} & \text{if } MAD = 0, \end{cases} \tag{56}$$

where $MeanAD = \dfrac{1}{\sum_{i=1}^{m} n_i} \sum_{i=1}^{m} n_i \, | EM_i - M |$.

(4) Detect unusual category as follows:

If $z_i >$ 3, a category C_i has an unusually high EMMEAN in the high group.

If $z_i < -3$, a category C_i has an unusually low EMMEAN in the low group.

Repeat the processes for grouping and unusual category detection analyses for all significant factors and factor interactions.

8. Scoring

8.1. Predictive and residual values

After the model has been fit, predicted and residual values are usually calculated and output.

Notice that the predicted values can be computed for the case not used in the model-building phase. That is the response variable can be missing and the predicted values are still computed as long all the predictor variables have non-missing values in the given model. An additional requirement is that given predictor variable values could be properly parameterized by using only the existing model parameters. See Woods (2004), “Guidelines for Scoring under Various Data and Model Conditions,” for details.

- **Predicted values**

$$\hat{y}_k = \sum_{i=0}^{p} x_{ki}\hat{\beta}_i,\ k = 1,\ldots,n. \tag{57}$$

- **Residuals**

$$e_k = y_k - \hat{y}_k \tag{58}$$

- **Studentized residuals**

This is the ratio of the residual to its standard error.

$$SRES_k = \frac{e_k}{s\sqrt{{(1-h_k)}/{g_k}}}\ , \tag{59}$$

where s is the square root of the mean square error, i.e., $s = \sqrt{SS_e/df_e}$ and h_k is the leverage value for the case k (see section 8.2 below).

- **Deleted residuals**

The deleted residual for case k is defined as the residual for the k^{th} case that results from dropping the kth case from the parameter estimates.

$$DRESID_k = e_k/(1-h_k)\ . \tag{60}$$

- **Studentized deleted residuals**

$$SDRESID_k = \frac{e_k}{s_{(k)}\sqrt{(1-h_k)/g_k}}\ , \tag{61}$$

where $s_{(k)} = \sqrt{s_{(k)}^2}$ with $s_{(k)}^2 = \dfrac{1}{(df_e - 1)}\left(df_e \cdot s^2 - \dfrac{g_k \cdot e_k^2}{1-h_k}\right).$

8.2. Influence statistics

These statistics can be calculated for each case to measure the influence of each case on the estimates.

- **Leverage values**

The leverage value h_k is defined as the k^{th} diagonal element of the hat matrix $\boldsymbol{H}$ with

$$\boldsymbol{H} = \boldsymbol{W}^{1/2}\boldsymbol{X}\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}\right)^{-}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}^{1/2} = \boldsymbol{W}^{1/2}\boldsymbol{X}\boldsymbol{G}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}^{1/2} \qquad (62)$$

so $h_k = g_k \boldsymbol{x}_k \boldsymbol{G} \boldsymbol{x}_k^{\mathrm{T}}$, $k = 1, 2, \cdots, n$.

Implementation note:

We can compute h_k in two ways and it is up to software engineer to decide which one is easier and faster:

(a) Plug $\boldsymbol{G} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X}\right)^{-}$, which how to compute is described in Section 6.1, into $h_k = g_k \boldsymbol{x}_k \boldsymbol{G} \boldsymbol{x}_k^{\mathrm{T}}$.

(b) Compute h_k directly as follows:

$$h_k = \begin{cases} \dfrac{g_k}{N-1}\displaystyle\sum_{i=1}^{p*}\sum_{j=1}^{p*}\frac{\left(x_{ki}-\overline{X}_i\right)\left(x_{kj}-\overline{X}_j\right)}{\sqrt{S_{ii}S_{jj}}}\tilde{r}_{ij} + \frac{g_k}{\displaystyle\sum_{t=1}^{n} w_t} & \text{if there is an intercept} \\[2ex] \dfrac{g_k}{N-1}\displaystyle\sum_{i=1}^{p*}\sum_{j=1}^{p*}\frac{x_{ki}x_{kj}}{\sqrt{S_{ii}S_{jj}}}\tilde{r}_{ij} & \text{if there is no intercept} \end{cases} \qquad (63)$$

Computing h_k is just related to the effects in the model, that is, we exclude the indices *i* and *j* corresponding to the effects out of the model in the sum from 1 to p. but since we have constructed the $\boldsymbol{G}$ matrix in computing type III SS, we could turn back to formula in (a) to get h_k.

- **Cook's distance**

$$COOK_k = \frac{e_k^2 h_k g_k}{s^2\left(1-h_k\right)^2 p^c}. \qquad (64)$$

8.3. Influential outliers

We will identity a record to be an influential outlier based on the following two statistics:

(1) Cook's distance is larger than $\frac{4}{N-p^c}$ (Fox, 1997).

(2) The absolute of studentized delete residual is larger than 2 (or 2.5).

The definition of Cook's distance is in Section 8.2 and and the definition of studentized delete residual is in Section 8.1.

9. Model diagnostics

For all assumptions in linear regression, we will only test homoskedasticity formally. If the test is rejected, the robust (or heteroskedasticity consistent or sandwich estimator) for covariance matrix of coefficient estimates would be computed then relevant statistics/tests would be updated accordingly. There are several assumptions entered the inferences for the estimators of the model. If all these assumptions are held, we can be confident about the estimated coefficients and their statistics are unbiased, efficient and consistent. The model diagnostics is to check whether these assumptions are held, how serious the consequence if one or more assumptions were found being violated indeed and what should be done in this situation. Currently, we focus on testing the assumptions of normality and homoscedasticity.

9.1. Homoskedasticity

The homoskedasticity assumption is about variance of the error (σ^2) is constant across records. When the assumption is violated, the OLS coefficient estimates are still consistent, but not efficient. So for valid inference, according to Huber (1967) or White (1980), a heteroskedastic consistent (HC) or robust estimator of covariance matrix of the estimated coefficient should be used. To investigate the homoskedasticity assumption properly and automatically, there are 3 steps:

(1) A test to determine if the homoskedasticity assumption is violated: a modified Breusch Pagan (BP) test would be used.

However, keep in mind that Long and Ervin (2000) recommend that "*a test for heteroskedasticity should not be used to determine whether* [*an HC estimator*] *should be used*." So the test is only used in automatic modeling process.

(2) If the test is rejected, compute a robust estimator to replace the model-based estimator: 4 variations would be provided.

(3) All statistics related to inference, such as t-statistics, p-values, confidence intervals in coefficient estimates, etc., should be computed based on robust estimators.

Three subsections describe each step in details.

9.1.1 The modified Breusch Pagan test

The original test is proposed by Breusch and Pagan (1979) based on Normality assumption on the error, then Koenker (1981) and Koenker and Bassett (1982) release Normality assumption so it is called the modified BP test and the test statistic is defined as follows:

$$S_{\text{MBP}} = N \times \frac{\boldsymbol{u}'\boldsymbol{Z}\left(\boldsymbol{Z}'\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}'\boldsymbol{u} - N\bar{u}^2}{\boldsymbol{u}'\boldsymbol{u} - N\bar{u}^2}, \tag{65}$$

where N is total sample size, $N = \sum_{i=1}^{n} f_i$, $\boldsymbol{u}$ be a $n \times 1$ vector of squared weighted residuals, i.e., $\boldsymbol{u} = \left(g_1 e_1^2, g_2 e_2^2, \ldots, g_n e_n^2\right)^{\mathrm{T}}$, $\bar{u} = \frac{1}{N}\sum_{i=1}^{n} f_i u_i$, and $\boldsymbol{Z}$ is a set of regressors which are related to $\boldsymbol{u}$. Note that typical $\boldsymbol{Z}$ would include all predictors in the design matrix $\boldsymbol{X}$ and their squares and cross products terms, but here we will assume $\boldsymbol{Z}$ = $\boldsymbol{X}$, then S_{MBP} will follow an asymptotic chi-square distribution with p^c degrees of freedom and the p-value can be computed accordingly.

Implementation note:

- In addition to $\boldsymbol{Z}$ = $\boldsymbol{X}$, we also assume $\left(\boldsymbol{Z}'\boldsymbol{Z}\right)^{-1} = \boldsymbol{G} = (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-}$ to simplify the computational process such that

$$S_{\text{MBP}} = N \times \frac{\left(\sum_{i=1}^{n} f_i u_i \cdot \boldsymbol{x}_i\right)\boldsymbol{G}\left(\sum_{i=1}^{n} f_i u_i \cdot \boldsymbol{x}_i\right)^{\mathrm{T}} - N\bar{u}^2}{\left(\sum_{i=1}^{n} f_i u_i^2\right) - N\bar{u}^2}, \tag{66}$$

where $\boldsymbol{x}_i = (x_{i0}, \ldots, x_{ip})$ is the i^{th} row of $\boldsymbol{X}$. Three summation terms should be straightforward to compute in map/reduce environment.

9.1.2 Robust estimator of coefficient estimate covariance

When the *p*-value < a significance level (default = 0.05), reject the null hypothesis of homoskedasticity and compute a robust estimator as follows:

$$\hat{\boldsymbol{\Psi}} = \boldsymbol{GX}'\boldsymbol{W}^{1/2}\hat{\boldsymbol{\Omega}}\boldsymbol{W}^{1/2}\boldsymbol{XG}, \tag{67}$$

where $\hat{\boldsymbol{\Omega}}$ is a diagonal matrix of variance estimates of weighted residuals, $\hat{\boldsymbol{\Omega}} = diag(\omega_1, \dots, \omega_n)$,

and there are 4 estimators differ in their choice of the ω_i:

HC0: $$\omega_i = u_i = g_i e_i^2 \tag{68}$$

HC1: $$\omega_i = \frac{N}{N - p^c} u_i \tag{69}$$

HC2: $$\omega_i = \frac{1}{1 - h_i} u_i \tag{70}$$

HC3: $$\omega_i = \frac{1}{(1 - h_i)^2} u_i \tag{71}$$

Notes:

- The estimator HC0 is introduced by White (1980), is justified by asymptotic arguments.
- The estimator HC1 – HC3 are suggested by MacKinnon and White (1985) to improve the performance in small samples and Long and Ervin (2000) conclude that HC3 provide the best performance in sample samples based on Monte Carlo simulation.
- Under homoskedasticity assumption, $\omega_i = SS_e / df_e = s^2$ (variance estimate of weighted residuals is constant), $\hat{\boldsymbol{\Omega}} = s^2 \boldsymbol{I_n}$ and $\hat{\boldsymbol{\Psi}} = s^2 \boldsymbol{G}$.

9.1.3 Affected statistics

Many statistics computed previously would be affected by replacing the original or model-based covariance matrix $\hat{\boldsymbol{\Psi}} = s^2 \boldsymbol{G}$ with the robust estimator $\hat{\boldsymbol{\Psi}} = \boldsymbol{G}X'W^{1/2}\hat{\boldsymbol{\Omega}}W^{1/2}X\boldsymbol{G}$ (assume the (*i*, *j*) element in $\hat{\boldsymbol{\Psi}}$ is $\psi_{i,j}$) and they are listed according to areas:

- **Statistics related to coefficient estimates** (in Section 7.1):

  $\hat{\sigma}_{\hat{\beta}_j} = \psi_{j+1,j+1}$, $j = 0,\dots,p$ (note that $\hat{\boldsymbol{\Psi}}$ includes intercept term if there is one); then t-statistics, *p*-values and confidence intervals should be updated as well.

- **Statistics related to tests of individual effects** (in Section 7.2):

  When the robust estimator is used, the F-statistics listed in ANOVA table cannot be computed based on sum of squares anymore, included (corrected) model. For each effect *j*, the F-statistic should be computed as

  $$F_j = \frac{\hat{\boldsymbol{\beta}}^{\mathrm{T}} \boldsymbol{L}_j^{\mathrm{T}} \left(\boldsymbol{L}_j \hat{\boldsymbol{\Psi}} \boldsymbol{L}_j^{\mathrm{T}} \right)^{-1} \boldsymbol{L}_j \hat{\boldsymbol{\beta}}}{r_j} \tag{72}$$

  and the F-statistics for corrected model (with intercept) and model (without intercept) can be computed similarly except the ***L*** matrix is from GEF (general estimable function). If there is no intercept, the ***L*** matrix is the whole GEF. If there is an intercept, the ***L*** matrix is GEF without the first row which corresponds to the intercept. (Please see the GLMM document for details).

- **Statistics related to EMMEANS** (in Section 7.5):

  When standard errors and comparison statistics are computed related to EMMEANS, the covariance matrix of coefficient estimates should be replaced by $\hat{\boldsymbol{\Psi}} = \boldsymbol{G}X'W^{1/2}\hat{\boldsymbol{\Omega}}W^{1/2}X\boldsymbol{G}$ (note that the notation used is $V(\hat{\beta})$ in Zheng (2009)).

9.2. Plots (in Model Viewer)

In this section, we will show what information should be saved for the StatXML file to create a scatter plot of observed by predicted target values, a scatter plot of predicted target values by residuals, histogram and PP plot of residuals in model viewer from binned data of the whole training set.

9.2.1. Scatter plot of predicted by observed target values

Let y_k and $\hat{y}_k$ be the target observed and predicted value of the k^{th} record, respectively, $k = 1,\cdots,n$. Then the information needed for a binned scatter plot of predicted by observed target values is created as follows:

Step 1. Using equal width method to compute 19 cut points $cut_1,\cdots,cut_{19}$ between the range $[a,b]$, where $a = \min\{y_k\}$ and $b = \max\{y_k\}$, i.e., $cut_i = a + i \times (b-a)/20$. Then we have 20 intervals $(cut_0, cut_1] = (-\infty, cut_1],\cdots,(cut_{19}, cut_{20}] = (cut_{19}, +\infty]$.

Step 2. For each two-dimension interval $(cut_i, cut_{i+1}] \times (cut_j, cut_{j+1}],\ i, j = 0,\cdots,19$, using map/reduce algorithm in Appendix A, we can get the number of cases that fall into this interval incorporating the frequency weight:

$$n_{ij,f} = \sum_{k=1}^{n} f_k I(cut_i < y_k \le cut_{i+1}\ and\ cut_j < \hat{y}_k \le cut_{j+1})$$

and the corresponding mean of $(y, \hat{y})$ incorporating the frequency weight (note that regression weight is not included):

$$Mean(i,j) = \frac{1}{n_{ij,f}} \left(\sum_{k=1}^{n} f_k I_{ij}(y_k, \hat{y}_k) y_k, \sum_{k=1}^{n} f_k I_{ij}(y_k, \hat{y}_k) \hat{y}_k \right)$$

where $I(.)$ is an indictor function defined as follows:

$$I_{ij}(y_k, \hat{y}_k) = \begin{cases} 1, & if\ cut_i < y_k \le cut_{i+1}\ and\ cut_j < \hat{y}_k \le cut_{j+1}; \\ 0, & otherwise \end{cases}$$

Step 3. Save the mean, $Mean(i, j)$, and the corresponding number of cases, $n_{ij,f}$, $i, j = 0,\cdots,19$, for the StatXML file. Note that if $n_{ij,f} = 0$, there is no need to save it and corresponding $Mean(i, j)$ which is (0, 0) as well.

9.2.2. Scatter plot of residuals by predicted target values

The construction of the scatter plot of predicted target values by residuals is very similar to that in Section 9.2.1, nonetheless it is described in details as follows:

Let $\hat{y}_k$ and e_k be the predicted value and the residual of the kth case, respectively, $k = 1, \cdots, n$.

Then the information needed for a binned scatter plot of predicted values by residuals is created as follows:

Step 1. Using equal width method to compute 19 cut points $cut_1^{(1)}, \cdots, cut_{19}^{(1)}$ between the range $[a, b]$ for the x-axis, where $a = \min\{y_k\}$ and $b = \max\{y_k\}$, , i.e., $cut_i^{(1)} = a + i \times (b - a) / 20$. Then we have 20 intervals :

$$(cut_0^{(1)}, cut_1^{(1)}] = (-\infty, cut_1^{(1)}], \cdots, (cut_{19}^{(1)}, cut_{20}^{(1)}] = (cut_{20}^{(1)}, +\infty].$$

Step 2. Similarly, compute 19 cut points $cut_1^{(2)}, \cdots, cut_{19}^{(2)}$ between the range $[-8s, 8s]$ for the y-axis: $cut_i^{(1)} = -8s + i \times 16s / 20$, where s is the square root of the mean square error, i.e. $s = \sqrt{SS_e / df_e}$. Then we have another 20 intervals :

$$(cut_0^{(2)}, cut_1^{(2)}] = (-\infty, cut_1^{(2)}], \cdots, (cut_{19}^{(2)}, cut_{20}^{(2)}] = (cut_{20}^{(2)}, +\infty].$$

Step 3. For each two-dimension interval $(cut_i^{(1)}, cut_{i+1}^{(1)}] \times (cut_j^{(2)}, cut_{j+1}^{(2)}], i, j = 0, \cdots, 19$, using map/reduce algorithm in Appendix A, we can get the number of cases that fall into this interval incorporating the frequency weight:

$$n_{ij,f} = \sum_{k=1}^{n} f_k I_{ij}(\hat{y}_k, e_k)$$

and the corresponding mean of $(\hat{y}, e)$ incorporating the frequency weight (note that regression weight is not included):

$$Mean(i,j) = \frac{1}{n_{ij,f}} \left(\sum_{k=1}^{n} f_k I_{ij}(\hat{y}_k, e_k)\hat{y}_k, \sum_{k=1}^{n} f_k I_{ij}(\hat{y}_k, e_k) e_k \right)$$

where

$$I_{ij}(\hat{y}_k, e_k) = \begin{cases} 1, & if\ cut_i^{(1)} < \hat{y}_k \leq cut_{i+1}^{(1)}\ and\ cut_j^{(2)} < e_k \leq cut_{j+1}^{(2)}; \\ 0, & otherwise \end{cases}$$

Step 4. Save the mean, $Mean(i, j)$, and the corresponding number of cases, $n_{ij,f}$, $i, j = 0, \cdots, 19$, for scatter plot of predicted values by residuals. Note that if $n_{ij,f} = 0$, there is no need to save it and corresponding $Mean(i, j)$ which is (0, 0) as well.

9.2.3. Histogram and PP plot

The information needed for binned histogram and PP plot of residuals is created as follows:

Step 1. Find out 400 cut points, $cut_1, \cdots, cut_{400}$, between $[-8s, 8s]$, such that

$cut_i = -8 \times s + i \times (16/400) \times s,$ where s is the square root of the mean square error,

i.e., $s = \sqrt{SS_e / df_e}$.

Step 2. For each bin $(cut_{i-1}, cut_i]$, using map/reduce algorithm in Appendix A, we can get the number of cases of e_k that fall into this bin incorporating the frequency weight:

$$n_{i,f} = \sum_{k=1}^{n} f_k I(cut_{i-1} < e_k \leq cut_i), i = 1, \cdots, 400$$

and the corresponding mean incorporating the frequency weight(note that regression weight is not included):

$$Mean_i = \frac{1}{n_{i,f}} \sum_{k=1}^{n} f_k I(cut_{i-1} < e_k \le cut_i) e_k, i = 1, \cdots, 400$$

where $I(.)$ is an indictor function and $cut_0 = -8s$.

For those e_k that are outside the range $(-8s, 8s]$, we also need to record each distinct value of e_k and the number of cases that equal to this distinct value, incorporating the frequency weight.

Step 3. After **step 2**, suppose we have m_1 distinct values, $e_{(1)} < \cdots < e_{(m_1)}$, that are less than or equal to $-8s$, and m_2 distinct values , $e_{(n-m_2+1)} < \cdots < e_{(n)}$, that are greater than $8s$. And the numbers of cases that e_k equal to these distinct values are $c_{1,f}, c_{2,f}, \cdots, c_{m_1,f}$ and $c^*_{n-m_2+1,f}, \cdots, c^*_{n,f}$.

Then we can get mean vector

$$Mean = [e_{(1)}, \cdots, e_{(m_1)}, Mean_1, \cdots, Mean_{400}, e_{(n-m_2+1)}, \cdots, e_{(n)}],$$

and quantile vector of residuals

$$Quan = [e_{(1)}, \cdots, e_{(m_1)}, cut_1, \cdots, cut_{400}, e_{(n-m_2+1)}, \cdots, e_{(n)}].$$

Frequency in bins

$$FreInBin = [c_{1,f}, c_{2,f}, \cdots, c_{m_1,f}, n_{1,f}, \cdots, n_{400,f}, c^*_{n-m_2+1,f}, \cdots, c^*_{n,f}],$$

and cumulative percentage of residuals:

$$CumPer = \frac{1}{N}[cc_{1,f}, cc_{2,f}, \cdots, cc_{400+m_1+m_2,f}].$$

where $cc_{k,f} = \sum_{i=1}^{k} FreInBin_i$, and $FreInBin_i$ is the *i*th element of $FreInBin$.

Step 4. For histogram, save the $Mean$ vector, $FreInBin$vectors, mean and standard deviation of residuals. Here the mean and standard deviation of residual is 0 and s , respectively. Again if the *i*th element of $FreInBin$ is 0, there is no need to save it and the corresponding element of $Mean$.

Step 5. For a PP plot, compute the cumulative probabilities vector of standard normal distribution from $Quan$ as follows:

$$Cum\Pr ob = [p_1, p_2, \cdots, p_{400+m_1+m_2}]$$

where $p_i = \Phi(Quan_i)$, and $Quan_i$ is the *i*th element of vector $Quan$.

Then save the vectors $CumPer$ and $Cum\Pr ob$ for the StatXML file as a PP plot is a plot of $CumPer$ by $Cum\Pr ob$. Again if the *i*th element of $FreInBin$ is 0, there is no need to save the corresponding element of $CumPer$ and $Cum\Pr ob$.

Implementation note:

- If $n \le 400$, then the data will not be binned. The histogram and PP plot of residual are created as follows:

  1. The residual e_k, the corresponding number of case f_k, mean and standard deviation of residuals are used for histogram of residual directly.

  2. For PP plot, the residual e_k are needed to sort first. Suppose after sort, the residuals are $e_{(1)} \le e_{(2)} \le \cdots \le e_{(n)}$, and the corresponding number of case are $f_1^*, f_2^*, \cdots, f_n^*$, then the vector of cumulative percentage of residuals is

$$CumPer = \frac{1}{N}[cc_{1,f}, cc_{2,f}, \cdots, cc_{n,f}]$$

where $cc_{k,f} = \sum_{i=1}^{k} f_i^*$.

And the cumulative probabilities vector of standard normal distribution is

$$Cum\,\mathrm{Pr}\,ob = [p_1, p_2, \cdots, p_n]$$

where $p_i = \Phi(e_{(i)})$.

References – Phase I

[1]. Belsley, D. A., Kuh, E. and Welsch, R. E. (1980), *Regression Diagnostics*, New York: John Wiley & Sons, Inc.

[2]. Chiu, Y. M. (1995a), “The four types of sum of squares for univariate β-Model,” *SPSS Internal Document*.

[3]. Chiu, Y. M. (1995b), “Calculation of the four types of sums of squares,” *SPSS Internal Document*.

[4]. Dempster, A. P. (1969), *Elements of Continuous Multivariate Analysis*, Reading, MA: Addison-Wesley.

[5]. Fox, J. (1997), *Applied Regression Analysis, Linear Models, and Related Methods*, Thousand Oaks, CA: SAGE Publications, Inc.

[6]. Han, S. (2010), “Interaction Detection for Two Factors,” *SPSS Internal Document*.

[7]. Han, S. and Zheng, P. (2010), “Linear Model Output List for NextGen,” *SPSS Internal Document*.

[8]. Lam, M. L. (1995a), “Constructing the Design Matrix for the β-Model,” *SPSS Internal Document*.

[9]. Lam, M. L. (1995b), “Algorithm: the symmetric sweep operator,” *SPSS Internal Document*.

[10]. SAS Institute Inc. (2004), “Chapter 61 the REG Procedure,” *SAS/STAR 9.1 User’s Guide*, Cary, NC, USA.

[11]. SPSS Inc. “REGRESSION Algorithm,” *SPSS Internal Document*.

[12]. Schatzoff, M., Tsao, R. and Fienberg, S. (1968), “Efficient computing of all possible regressions,” *Technometrics*, 10,769–779.

[13]. Smirnov, N.V. (1948), “Table for estimating the goodness of fit of empirical distribution,” *Annals of the Mathematical Statistics*, 19,279-281.

[14]. Velleman, P. F. and Welsch, R. E. (1981), “Efficient computing of regression diagnostics,” *American Statistician*, 35, 234–242.

[15]. Woods, M. (2004), “Guideline for Scoring under Various Data and Model Conditions,” *SPSS Internal Document*.

[16]. Xu, J. (2011), “Evaluation of Algorithms for Predictor Importance,” *SPSS Internal Document*.

[17]. Zheng, P. (2009), “Algorithm: EMMEANS and Custom Tests,” *SPSS Internal Document*.

[18]. Zheng, P. (2010), “Detecting Factor-Covariate interaction,” *SPSS Internal Document.*

[19]. Zhong, R. (2008), “Algorithm: Variable Importance,” *SPSS Internal Document.*

Appendix A: Map Reduce Algorithm for Some Statistics

A.1. Notation

The following notation is used throughout the appendix unless otherwise stated:

n	Number of distinct records in the whole dataset. It is an integer and $n \geq 1$.
f_i	Frequency count for record *i*.
g_i	Regression weight for record *i*.
w_i	Combined weight for record *i*, $w_i = f_i \times g_i$.
N	Effective sample size. it is a integer number, $N = \sum_{i=1}^{n} f_i$ .If frequency count variable *f* is not use, *N*=*n*.
W	Total combined weight, $W = \sum_{i=1}^{n} w_i$.
$\bar{X}$	Weighted mean of a continuous variable X with y_i is the value for record *i*.
$\bar{Y}$	Weighted mean of a continuous variable Y with y_i is the value for record *i*.
C_{xy}	$C_{xy} = \begin{cases} \sum_{i=1}^{n} w_i \left(x_i - \bar{X}\right)\left(y_i - \bar{Y}\right) & \text{if centered or a model with the intercept;} \\ \sum_{i=1}^{n} w_i x_i y_i & \text{if non-centered or a model without the intercept.} \end{cases}$
S_{xy}	Weighted covariance between *X* and *Y*, so $S_{xy} = \frac{C_{xy}}{N-1}$; and S_{xx} and S_{yy} would be weighted variance of *X* and *Y*, respectively.
r_{xy}	Weighted correlation between *X* and *Y*, so $r_{xy} = \frac{S_{xy}}{\sqrt{S_{xx} S_{y}}}$.

A.2. Computing Correlation

For constructing the correlation matrix $\boldsymbol{R}$ which is a $(p+1)\times(p+1)$ matrix, where p is the number of parameters, there are $p(p-1)/2$ pairs of correlation to compute. Without loss generality, suppose a pair of variables is *X* and *Y*. Then also suppose there are *M* mappers and more than one reducer, then the correlation can be computed in map/reduce environment as follows:

(1) Provisional means algorithm in each mapper:

Denote N_j is the cumulative frequency weight up to record *j*, $N_j = \sum_{i=1}^{j} f_i$.

W_j is the cumulative combined weight up to record *j*, $W_j = \sum_{i=1}^{j} w_i$.

$\bar{X}_j$ is the estimate of $\bar{X}$ up to record *j*; $\bar{Y}_j$ is the estimate of $\bar{Y}$ up to record *j*.

$C_{xy,j}$ is the estimate of C_{xy} up to record *j*.

Start with $N_0 = W_0 = \bar{X}_0 = \bar{Y}_0 = C_{xy,0} = 0$, then compute the following statistics recursively for all records in the mapper:

$$N_j = N_{j-1} + f_j,$$

$$W_j = W_{j-1} + w_j,$$

$$\bar{X}_j = \bar{X}_{j-1} + \frac{w_j}{W_j}\left(x_j - \bar{X}_{j-1}\right),$$

$$\bar{Y}_j = \bar{Y}_{j-1} + \frac{w_j}{W_j}\left(y_j - \bar{Y}_{j-1}\right),$$

$$C_{xy,j} = \begin{cases} C_{xy,j-1} + \left(w_j - \frac{w_j^2}{W_j}\right)\left(x_j - \bar{X}_{j-1}\right)\left(y_j - \bar{Y}_{j-1}\right) & \text{if centered or a model with intercept;} \\ C_{xy,j-1} + w_j x_j y_j & \text{if non-centered or a model without intercept.} \end{cases}$$

(2) Combine statistics from *K* mappers to one reducer or from more than one reducers (without loss generality, assume it is also *K*) to the “finalizer”:

Denote $N^{(k)}, W^{(k)}, \bar{X}^{(k)}, \bar{Y}^{(k)},$ and $C_{xy}^{(k)}$ are the resulting statistics from the *k*th mapper or reducer.

Compute

$$N = \sum_{k=1}^{K} N^{(k)},$$

$$W = \sum_{k=1}^{K} W^{(k)},$$

$$\bar{X} = \frac{1}{W}\sum_{k=1}^{K} W^{(k)}\bar{X}^{(k)},$$

$$\bar{Y} = \frac{1}{W}\sum_{k=1}^{K} W^{(k)}\bar{Y}^{(k)},$$

$$C_{xy} = \begin{cases} \sum_{k=1}^{K} C_{xy}^{(k)} + \sum_{k=1}^{K} W^{(k)}\bar{X}^{(k)}\bar{Y}^{(k)} - W\bar{X}\bar{Y} & \text{if centered or a model with intercept;} \\ \sum_{k=1}^{K} C_{xy}^{(k)} & \text{if non-centered or a model without intercept.} \end{cases}$$

If it is in the “finalizer” for constructing the ***R*** matrix, then also calculate the weighted variances, covariance and correlation $S_{xx} = \frac{C_{xx}}{N-1}$, $S_{yy} = \frac{C_{yy}}{N-1}$, $S_{xy} = \frac{C_{xy}}{N-1}$, and $r_{xy} = \frac{S_{xy}}{\sqrt{S_{xx}S_y}}$.

A.3. Computing statistics for interaction detection for two factors

Without loss generality, suppose a pair of factors is X_1 with known R levels and X_2 with known S levels and continuous target is Y. Then the statistics needed in the $R \times S$ matrix are the number of records (N_{ij}), the target mean ($\bar{Y}_{ij}$), and the sum of squared terms of target ($C_{yy,ij}$) for all combinations of $X_1 = i,\ i = 1,\dots,R,$ and $X_2 = j,\ j = 1,\dots,S.$ Please note that regression weights will not be used here even it is specified and $C_{yy,ij}$ would be computed based on “centered or a model with intercept” condition. The computation of the matrix with N_{ij} , $\bar{Y}_{ij}$ and $C_{yy,ij}$ in each cell is similar to that in Section A.2 with frequency weight and Y value putting in the right cell. The results from the finalizer are the table:

X_2 X_1	1	2	$\cdots$	S
1	$N_{11}, \bar{Y}_{11}, C_{yy,11}$	$N_{12}, \bar{Y}_{12}, C_{yy,12}$	$\cdots$	$N_{1S}, \bar{Y}_{1S}, C_{yy,1S}$
2	$N_{21}, \bar{Y}_{21}, C_{yy,21}$	$N_{22}, \bar{Y}_{22}, C_{yy,22}$	$\cdots$	$N_{2S}, \bar{Y}_{2S}, C_{yy,2S}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$

R	$N_{R1}, \overline{Y}_{R1}, C_{yy,R1}$	$N_{R2}, \overline{Y}_{R2}, C_{yy,R2}$	$\ldots$	$N_{RS}, \overline{Y}_{RS}, C_{yy,RS}$

A.4. Computing statistics for interaction detection for a covariate and a factor

Without loss generality, suppose a covariate is X_1, a factor is X_2 with known S levels and continuous target is Y. The statistics needed in the $1\times S$ matrix are the number of records (N_j), the means for X_1 and Y ($\overline{X}_{1,j}$ and $\overline{Y}_j$), the sum of squared terms for X_1 and Y ($C_{x_1x_1,j}$ and $C_{yy,j}$), and the sum of cross product terms for X_1 and Y ($C_{x_1y,j}$) for $X_2 = j,\ j = 1,\ldots,S$. Please note that regression weights will not be used here even it is specified and $C_{x_1x_1,j}, C_{yy,j}$ and $C_{x_1y,j}$ would be computed based on "centered or a model with intercept" condition. The computation of the matrix with $N_j, \overline{X}_{1j}, \overline{Y}_j, C_{x_1x_1,j}, C_{yy,j}$ and $C_{x_1y,j}$ in each cell is similar to that in Section 1.2 with frequency weight and X_1 and Y values putting in the right cell. The results from the finalizer are the table:

$X_2 = 1$	$\ldots$	$X_2 = S$
$N_1, \overline{X}_{1,1}, \overline{Y}_1, C_{x_1x_1,1}, C_{yy,1}, C_{x_1y,1}$	$\ldots$	$N_S, \overline{X}_{1,S}, \overline{Y}_S, C_{x_1x_1,S}, C_{yy,S}$ and $C_{x_1y,S}$

Appendix B: Sweep operations

Sweep operations on matrix ***R*** (Dempster, 1969) are used to compute the standardized least squares estimation b and the associated regression statistics. The sweeping starts with the correlation matrix R. Let $\tilde{R}$ be the new matrix produced by sweeping on the kth row and column of R. The elements of $\tilde{R}$ are

$$\tilde{r}_{ij} = r_{ij} - \frac{r_{ik}r_{kj}}{r_{kk}},\ i \neq k,\ j \neq k\ ;$$

$$\tilde{r}_{ik} = -\frac{r_{ik}}{r_{kk}}, i \neq k\ ;$$

$$\tilde{r}_{kj} = \frac{r_{kj}}{r_{kk}},\ j \neq k\ ;$$

and $\tilde{r}_{kk} = \dfrac{1}{r_{kk}}$.

For a partition matrix, $R = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, where A is a $s \times s$ matrix. Sweep operation is performed on

the *s* pivot elements in *A*. resulting matrix $\tilde{R} = \begin{pmatrix} A^{-1} & A^{-1}B \\ -CA^{-1} & D - CA^{-1}B \end{pmatrix}$.

If the above sweep operations are repeatedly applied to each row of R_{11}, where R_{11} contains independent variables in the model at the current step, the result is

$$\tilde{R} = \begin{pmatrix} R_{11}^{-1} & R_{11}^{-1}R_{12} \\ -R_{21}R_{11}^{-1} & R_{22} - R_{21}R_{11}^{-1}R_{12} \end{pmatrix}.$$

Sweep operation computes the determinant of a matrix.

$$DET(R) = \prod_{i=1}^{p} \tilde{r}_{ii} \ .$$

Appendix C: A method to search $({}_{p^e}C_{p^s} + 1)$ models

1. Notations & Definitions:

a) Each model can be denoted by an array of numbers $\{t_1, t_2, \cdots, t_{p^e}\}$,

$$t_i = \begin{cases} 0 & \text{if the corresponding effect is not in the model} \\ l & \text{the \# of levels of the effect (for continuous variable } l = 1) \end{cases}, (i = 1, \cdots, p^e) \ .$$

b) The search space of models $S = S_1 \cup S_2$. S_1 represents the set of models have been detected, and S_2 represents not yet.

c) The length d_{ij} of two distinct models, that is two distinct vectors $\{t_{i1}, \cdots t_{ip^e}\}$ and $\{t_{j1}, \cdots t_{jp^e}\}$ is

computed as follows $d_{ij} = \sum_{k=1}^{p^e} \left| t_{ik} - t_{jk} \right|$.

Notes:

1. Since the number of effect is fixed at the number p^s from forward stepwise, the number of non-zero elements in vector $\{t_i\}$ is p^s .
2. The length d_{ij} means the steps of sweep operation between two distinct models. The minimum is 2. One continuous variable is swept out and another continuous variable is swept in, but the others holds on.

2. Algorithm:

This is an algorithm used for searching all the models with fixed effect size, when p^e >30 and p^s >60. Here variable RESTART in the algorithm is an integer. For numerical stability and avoiding many round errors of sweeping operation, we refresh the current swept matrix from initial matrix **R**, after doing sweeping operations up to an extent.

Step 1. $S_1 = t^0 = \{t_{01}, t_{02}, \cdots, t_{0p^e}\}$, $S_2 = S \setminus t^0$, t^0 can be select by the last p^s effects are in the model. Set t^0 as the current model and calculate the corresponding criterion value of the current model. *step* = 0.

Step 2. If $S_2 = \phi$, stop, sort the criterion value of all the models searched and output the best one, else go to Step 3.

Step 3. Computing all the length value between current model and the models in S_2. Select $t_{j^*} = \min\{\arg\min\{d_{current,j}\}\}, j = 1, \cdots |S_2|$.

Step 4. If *step* <RESTRAT, do sweep operation based on the current model to the model t_{j^*} , *step*=*step*+1. Else calculate the model t_{j^*} from the initial sweep matrix ***R***, *step*=0.

Step 5. $S_1 = S_1 \cup t_{j^*}$ and $S_2 = S_2 \setminus t_{j^*}$, Set t_{j^*} as the current model; calculate the criterion value of the current model. go to Step 2.

Notes:

1. For set S we collect all the models whose number of positive elements in vector $\{FLAG_i\}_{i=1}^{p_e}$ is fixed at p^s, from 0 to $2^{p^e}-1$. We do not need to store S$_1$, but only S$_2$.

2. In Step 3, if there are models with $d_{current,j1} = d_{current,j2}$, we select $j^* = \min\{j_1, j_2\}$.

3. As we know that the minimum length is 2, so if get the length value is 2 for the first time, we can stop and choose the model as t_{j^*}.

4. After searching all the models, we sort all the criteria value and give out the best one. For adjusted R square criterion, we output the model with max value. For other criterion, we output the one with min value.

3. Example:

Here we give out an example with 5 effects, 3 are continuous, and the other 2 are categorical variables with 2 levels and 3 levels.

The length of $\{1,0,0,2,3\}$ and $\{1,0,1,0,3\}$ is d = 1+2=3.

All the models we search is S = {(0,0,1,2,3);(0,1,0,2,3);(0,1,1,0,3);(0,1,1,2,0); (1,0,0,2,3);(1,0,1,0,3); (1,0,1,2,0); (1,1,0,0,3); (1,1,0,2,0); (1,1,1,0,0)}. The table below shows the detailed steps of all possible subset searching with fixed effect size.

step	Current model	S$_1$	S$_2$	Distance vector
0	(0,0,1,2,3)	(0,0,1,2,3)	(0,1,0,2,3);(0,1,1,0,3);(0,1,1,2,0); (1,0,0,2,3);(1,0,1,0,3); (1,0,1,2,0); (1,1,0,0,3); (1,1,0,2,0); (1,1,1,0,0)	{2,…}
1	(0,1,0,2,3)	(0,0,1,2,3);(0,1,0,2,3)	(0,1,1,0,3); (0,1,1,2,0); (1,0,0,2,3); (1,0,1,0,3); (1,0,1,2,0); (1,1,0,0,3); (1,1,0,2,0); (1,1,1,0,0)	{3,4,2,…}
2	(1,0,0,2,3)	(0,0,1,2,3); (0,1,0,2,3); (1,0,0,2,3)	(0,1,1,0,3); (0,1,1,2,0); (1,0,1,0,3); (1,0,1,2,0); (1,1,0,0,3); (1,1,0,2,0); (1,1,1,0,0)	{5,6,3,4,5,4,7}
3	(1,0,1,0,3);	(0,0,1,2,3); (0,1,0,2,3); (1,0,0,2,3); (1,0,1,0,3);	(0,1,1,0,3); (0,1,1,2,0); (1,0,1,2,0); (1,1,0,0,3); (1,1,0,2,0); (1,1,1,0,0)	{2,…}
4	(0,1,1,0,3);	(0,0,1,2,3); (0,1,0,2,3);	(0,1,1,2,0); (1,0,1,2,0); (1,1,0,0,3);	{5,7,2,…}

		(1,0,0,2,3); (1,0,1,0,3); (0,1,1,0,3);	(1,1,0,2,0); (1,1,1,0,0)	
5	(1,1,0,0,3)	(0,0,1,2,3); (0,1,0,2,3); (1,0,0,2,3); (1,0,1,0,3); (0,1,1,0,3); (1,1,0,0,3)	(0,1,1,2,0); (1,0,1,2,0); (1,1,0,2,0); (1,1,1,0,0)	{7,7,5,4}
6	(1,1,1,0,0)	(0,0,1,2,3); (0,1,0,2,3); (1,0,0,2,3); (1,0,1,0,3); (0,1,1,0,3); (1,1,0,0,3); (1,1,1,0,0)	(0,1,1,2,0); (1,0,1,2,0); (1,1,0,2,0);	{3,3,3}
7	(0,1,1,2,0)	(0,0,1,2,3); (0,1,0,2,3); (1,0,0,2,3); (1,0,1,0,3); (0,1,1,0,3); (1,1,0,0,3); (1,1,1,0,0); (0,1,1,2,0)	(1,0,1,2,0); (1,1,0,2,0);	{2,…}
8	(1,0,1,2,0)	(0,0,1,2,3); (0,1,0,2,3); (1,0,0,2,3); (1,0,1,0,3); (0,1,1,0,3); (1,1,0,0,3); (1,1,1,0,0); (0,1,1,2,0); (1,0,1,2,0);	(1,1,0,2,0);	
9	(1,1,0,2,0)	S	none	

10. Linear AS (Phase II)

For Linear AS (Linear Engine) phase II, only effect size measures and the corresponding confidence intervals (CIs) would be included. The document describes how to compute them in details. The effect size measures and confidence intervals are complementary to significance tests because, unlike significance tests, they would not be affected by the sample size.

The document is organized as follows: Section 11 gives notations. Then Section 12 gives definitions of effect sizes for model effects and coefficients and computational details of their confidence intervals. To construct a confidence interval, the bisection method is used to find the solution of probability equation for the noncentrality parameter and it is described in Section 13.

11. Notations

The following notation is used throughout the document unless otherwise stated:

SS_t	Weighted total sum of squares
SS_e	Weighted residual sum of squares

SS_r	Weighted regression sum of squares
S_j	Type III sum of square for the *j*th effect, $j=1,\cdots,p^e$, where p^e is the number of effects excluding intercept
df_t	Degrees of freedom of SS_t
df_e	Degrees of freedom of SS_e
df_r	Degrees of freedom of SS_r
r_j	Degrees of freedom of S_j
F	F statistic for corrected model , $F=\dfrac{SS_r/df_r}{SS_e/df_e}$
F_j	F statistics for the *j*th effect, $F_j=\dfrac{S_j/r_j}{SS_e/df_e}$
$\hat{\beta}_j$	The estimation of the *j*th parameter β_j
$\hat{\sigma}_{\beta_j}$	The standard error of $\hat{\beta}_j$
t_j	t statistic for the *j*th parameter, $t_j=\dfrac{\hat{\beta}_j}{\hat{\sigma}_{\beta_j}}$
$F(df_1,df_2,\lambda)$	A random variable follows the non-central F distribution with degrees of freedom df_1 and df_2, and a noncentrality parameter λ. If $\lambda=0$, then it is a random variable following the central F distribution with degree of freedom df_1 and df_2.
α	Significance level. Please note that we only assign a confidence interval level related to model effects and coefficient estimates in FDD (F0401 but not F0405 in Linear Engine FDD), so the significance level here should be related to it. For example, the default confidence interval = 95, then $\alpha=0.05$.

12. Effect Size

For model effects, the effect size measures include partial eta squared and eta squared. Their definitions and the confidence intervals are described in Section 12.1. Then for coefficient estimates, the effect size measure is the partial eta squared. The definition and computation of confidence interval would be given in Section 12.2.

12.1. Effect sizes and confidence intervals for effects

The partial eta squared for (corrected) model and the j^{th} effect are defined as

$$\eta^2_{p,r} = \frac{SS_r}{SS_r + SS_e} \quad \text{and} \quad \eta^2_{p,\,e_j} = \frac{S_j}{S_j + SS_e}, \text{ respectively.}$$

Note if there is an intercept, then SS_r is “SS for Corrected Model” and if there is no intercept, then SS_r is “SS for Model”. So we use (corrected) model to represent both situations.

To construct confidence intervals for those effects sizes, we need to connect the effect size with the noncentrality parameter of test distribution which is F distribution for tests of model effects. Based on the definition in GLM procedure, the noncentrality parameters for (corrected) model and the defined as j^{th} effect are defined as

$$\lambda_r = \frac{df_e \times SS_r}{SS_e} \quad \text{and} \quad \lambda_{e_j} = \frac{df_e \times S_j}{SS_e}, \text{ respectively.}$$

Thus the relationships between F statistics and noncententrality parameters for (corrected) model and the j^{th} effect are

$$F = \frac{SS_r / df_r}{SS_e / df_e} = \frac{\lambda_r}{df_r} \quad \text{and} \quad F_j = \frac{S_j / r_j}{SS_e / df_e} = \frac{\lambda_{e_j}}{r_j} \text{ respectively.}$$

Then the partial eta squared for (corrected) model and the j^{th} effect can be written based on the noncentrality parameter as

$$\eta^2_{p,r} = \frac{\lambda_r}{\lambda_r + df_e} \quad \text{and} \quad \eta^2_{p,\,e_j} = \frac{\lambda_{e_j}}{\lambda_{e_j} + df_e}, \text{ respectively.}$$

If we want the confidence intervals for effect sizes to be equivalent to the F tests of model effects, which employs a one-sided and upper tailed probability with significance level of α, we should employ a confidence coefficient of $(1-2\alpha)$. Thus $100(1-2\alpha)\%$ confidence interval of partial eta squared for both (corrected) model and the j^{th} effect is

$$\left(\frac{\lambda_l}{\lambda_l + df_e}, \frac{\lambda_u}{\lambda_u + df_e}\right),$$

where λ_l and λ_u are the lower and upper noncentrality parameters corresponding to the F statistics, respectively. λ_l for (corrected) model and the j^{th} effect could be obtained by solving the following equations

$$\Pr(F(df_r, df_e, \lambda_l) \le F) - (1-\alpha) = 0 \text{ and } \Pr(F(r_j, df_e, \lambda_l) \le F_j) - (1-\alpha) = 0,$$

respectively. λ_u for both (corrected) model and the j^{th} effect could be obtained solving the following equations

$$\Pr(F(df_r, df_e, \lambda_u) \le F) - \alpha = 0 \quad \text{and} \quad \Pr(F(r_j, df_e, \lambda_u) \le F_j) - \alpha = 0,$$

respectively. Please see Section 4 for details on how to obtain λ_l and λ_u.

The eta squared for (corrected) model and the j^{th} effect are defined as

$$\eta^2 = \frac{SS_r}{SS_t} \text{ and } \eta^2_{e_j} = \frac{S_j}{SS_t}, \text{ respectively.}$$

An exact confidence interval for eta squared is not available, but if we write the formula for η^2 as

$$\eta^2 = \frac{SS_r}{SS_t} = \frac{SS_r}{SS_r + (SS_t - SS_r)},$$

then a conservative confidence interval can be constructed as for $\eta^2_{p,r}$ by treating $SS_t - SS_r$ as SS_e and $df_t - df_r$ as df_e. Thus $100(1-2\alpha)\%$ confidence of eta squared for (corrected) model is defined as

$$\left(\frac{\lambda_l}{\lambda_l + df_t - df_r}, \frac{\lambda_u}{\lambda_u + df_t - df_r} \right),$$

where λ_l and λ_u can be computed by solving the below equation:

$$\Pr\left(F(df_r, df_t - df_r, \lambda_l) \le \frac{SS_r / df_r}{(SS_t - SS_r)/(df_t - df_r)} \right) - (1-\alpha) = 0$$

and

$$\Pr\left(F(df_r, df_t - df_r, \lambda_u) \le \frac{SS_r / df_r}{(SS_t - SS_r)/(df_t - df_r)} \right) - \alpha = 0$$

Similarly, $100(1-2\alpha)\%$ confidence of eta squared for the *j*th effect is defined as

$$\left(\frac{\lambda_l}{\lambda_l + df_t - r_j}, \frac{\lambda_u}{\lambda_u + df_t - r_j} \right),$$

where λ_l and λ_u can be computed by solving the below equation:

$$\Pr\left(F(r_j, df_t - r_j, \lambda_l) \le \frac{S_j / r_j}{(SS_t - S_j)/(df_t - r_j)} \right) - (1-\alpha) = 0$$

and

$$\Pr\left(F(r_j, df_t - r_j, \lambda_u) \le \frac{S_j / r_j}{(SS_t - S_j)/(df_t - r_j)} \right) - \alpha = 0, \text{ respectively.}$$

12.2. Effect sizes and confidence intervals for coefficients

The partial eta squared for the *j*th coefficient is defined as

$$\eta^2_{p,\beta_j} = \begin{cases} \hat{\beta}_j^2 / (\hat{\beta}_j^2 + df_e \times \hat{\sigma}^2_{\beta_j}) & \text{if } df_e > 0 \text{ and } \hat{\beta}_j \neq 0 \\ 1 & \text{if } \hat{\sigma}^2_{\beta_j} = 0 \text{ and } \hat{\beta}_j \neq 0 \\ SYSMIS & \text{Otherwise} \end{cases}$$

Then the noncentrality parameter, λ_{β_j}, and the test statistic related to it are defined as

$$\lambda_{\beta_j} = \frac{\hat{\beta}_j^2}{\hat{\sigma}^2_{\beta_j}} \quad \text{and} \quad F_j = t_j^2 = \lambda_{\beta_j}, \text{ respectively.}$$

If $df_e > 0$ and $\hat{\beta}_j \neq 0$, then $100(1-2\alpha)\%$ confidence interval of partial eta squared for $\hat{\beta}_j$ is

$$\left(\frac{\lambda_l}{\lambda_l + df_e}, \frac{\lambda_u}{\lambda_u + df_e} \right),$$

where λ_l and λ_u can be computed by solving the below equations:

$$\Pr\left(F(1, df_e, \lambda_l) \leq t_j^2\right) - (1-\alpha) = 0$$

and

$$\Pr\left(F(1, df_e, \lambda_u) \leq t_j^2\right) - \alpha = 0, \text{ respectively.}$$

If partial eta squared is 1 or system missing, then confidence interval will not be computed.

13.Bisection method for noncentality parameter

We would use the bisection method to solve the following equation for noncentrality parameter (λ) of noncentral F distribution

$$\Pr(F(df_1, df_2, \lambda) \leq Fvalue) - prob = 0,$$

where df_1, df_2, $Fvalue$ and $prob$ are known value.

Denote $f(\lambda) = \Pr(F(df_1, df_2, \lambda) \leq Fvalue) - prob,$ then the bisection method is described as follows:

Step 1. If $f(0) \leq 0,$ then stop and output $\lambda = 0.$

Step 2. Let $x = Fvalue \times df_1$. If $f(x) = 0$, then stop and output $\lambda = x$; otherwise, go to step 3 to find out two values, x_1 and x_2 ,such that $f(x_1) \times f(x_2) < 0$.

Step 3. If $f(x) > 0$, then $x_1 = 2^{J-1} \times x$ and $x_2 = 2^J \times x$, where J is the minimum positive integer such that $f(x_1) \times f(x_2) \le 0$.

If $f(x) < 0$, then $x_1 = \frac{1}{2^J} \times x$ and $x_2 = \frac{1}{2^{J-1}} \times x$, where J is the minimum positive integer such that $f(x_1) \times f(x_2) \le 0$.

Step 4. If $f(x_1) = 0$ or $f(x_2) = 0$, then stop and output $\lambda = x_1$ if $f(x_1) = 0$ or $\lambda = x_2$ if $f(x_2) = 0$; otherwise, let $x = \frac{x_1 + x_2}{2}$ and go to step 5.

Step 5. If $|f(x)| \le \varepsilon$ and $|x_2 - x_1| \le \varepsilon$, where ε is a tolerance level and the default is tentatively set to 10^{-6}, then stop and output $\lambda = x$. Otherwise, go to step 6.

Step 6. If $f(x) > 0$, let $x_1 = x$, else let $x_2 = x$. Let $x = \frac{x_1 + x_2}{2}$, and go back to step 5.

References – Phase II

[20]. Maxwell, S. E. (2000), “Sample Size and Multiple Regression Analysis,” Psychological Methods, 5, 434–458.

[21]. Smithson, M. (2003), Confidence Intervals, Thousand Oaks, CA: Sage Publications.

[22]. Steiger, J. H. and Fouladi, R. T. (1997), “Noncentrality Interval Estimation and the Evaluation of Statistical Models,” in L. Harlow, S. Mulaik, and J. H. Steiger, eds., *What If There Were No Significance Tests?*, 222–257, Hillsdale, NJ: Erlbaum.

Linear SVM Algorithm

1. Introduction

The support vector machine (SVM) is a supervised learning method that generates input-output mapping functions from a set of labeled training data. The mapping function can be either a classification function or a regression function. For classification, non-linear kernel functions are often used to transform input data to a high-dimensional feature space in which the input data become more separable compared to the original input space. Maximum-margin hyperplanes are then created. The produced model depends on only a subset of the training data near the class boundaries. Similarly, the model produced by support vector regression ignores any training data that is sufficiently close to the model prediction (support vectors can appear only on the error tube boundary or outside the tube). SVMs are also said to belong to "kernel methods".

SVMs improve on classic classification models in the following ways: (1) Avoids underfitting. When the sample size is small, the model may be too simple. Simple models don't generalize well—that is, they aren't very valid on test data. (2) Avoids overfitting. When the sample size is large, the model may be too complex. Complex models also do not generalize well. (3) Work well when the number of predictors is small. (4) Work well when the number of predictors is large. Outperforms and is more valid than C5.0, C&RT, and Neural Net.

The disadvantage of the traditional SVMs is their high time complexity w.r.t the number of records, that is, the time complexity is $O(n^2)$, although it can be solved by a fast algorithm, sequential minimal optimization (SMO). In addition, the SMO algorithm is hard to be parallelized. To overcome it, linear SVM (LSVM) is often used.

LSVM, its feature space being the same as the input space of the problem, is the newest extremely fast machine learning algorithm. LSVM can be linearly scalable, which means that it builds a SVM model in a CPU time which scales linearly with the number of the records. Thus, LSVM is very suited to the large scale problems in terms of the volume of records and the number of variables (parameters). In addition, LSVM can easily handle the sparse data where the average number of non-zero elements in one record is small.

LSVM is different from the existing SVM in IBM SPSS Modeler in the following aspects: (1) the former is linear while the latter can be linear or nonlinear; (2) they use different optimization methods, wherethe former focuses on the primal optimization while the latter goes directly the dual formation; (3) the former can handle large number of records, while it is hard for the latter.

This document describes LSVM. The functions of LSVM will contain two main data mining tasks: (1) classification, including binary and multi-class classification; (2) regression. In addition, we provide a few post-estimation statistics: for the task of classification, we will provide an approximation probability for each prediction, and for regression, we will provide the standard deviation of the predictive value.

All optimization included in the LSVM will be solved by ADMM algorithm (Zhong, 2014), which will be implemented in a distributed computing environment, specifically, the map-reduce environment.

Section 2 describes the classification and regression models of LSVM. Section 3 presents the parameter estimation. Section 4 gives the post-estimation statistics.

2. Models

n	The total number of records
p	The number of parameters
$\mathbf{x}_i$	The i-th record, $\mathbf{x}_i \in \mathbb{R}^p$
ω_i	The case weight for the i-th record
y_i	The target, $y_i \in \{+1, -1\}$ for the binary classification, $y_i \in \{1, 2, \cdots, m\}$ for the multi-class classification, and $y_i \in \mathbb{R}$ for the regression.
$\mathbf{w}$	The parameter vector for classification and regression; $\mathbf{w} \in \mathbb{R}^p$for the binary classification and regression; $\mathbf{w} = [\mathbf{w}^{1,T}, \cdots, \mathbf{w}^{m,T}]^T \in \mathbb{R}^{pm}$ for the multi-class classification.
$\mathbf{w}^1, \cdots, \mathbf{w}^m$	The parameter vectors for the multi-class classification, and $\mathbf{w}^j \in \mathbb{R}^p, j \in [1, m]$
$\lVert\mathbf{x}\rVert_1$	The L_1 norm of the vector $\boldsymbol{x}$, which is defined as $\lVert\boldsymbol{x}\rVert_1 = \lvert x_1\rvert + \cdots + \lvert x_p\rvert$
$\lVert\mathbf{x}\rVert_2$	The L_2 norm of the vector $\boldsymbol{x}$, which is defined as $\lVert\boldsymbol{x}\rVert_2 = \left(x_1^2 + \cdots + x_p^2\right)^{1/2}$
N	The number of data blocks (parts)
B_i	$\{B_1, \cdots, B_N\}$ be a partition of all data indices $\{1, \cdots, n\}$.
p_c	The threshold denoting whether there is a large number of parameters (large p). If $p > p_c$, it is called large p situation, otherwise, it is called small to medium p situation.
$\boldsymbol{s}$	The gradient vector (function)
$\mathbf{H}$	The Hessian matrix (function)
λ	The parameter denoting the penalty
ϵ	The parameter denoting the sensitivity of the loss for regression

Two main tasks, classification and regression, are included. Their mathematical representations are given in Section 2.1 and 2.2.

2.1 Classification

The classification is used to classify cases into a group of defined categories of a target (response) variable of interest using a set of predictors. If the target has two categories, it is called the binary classification problem; if the target has more than two categories, it is called the multi-class classification problem.

2.1.1 Binary classification

For the binary classification, let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ denote a dataset, where $y_i \in \{+1, -1\}$, then LSVM has a general form

$$\min_{\mathbf{w}} \frac{1}{q}\lVert\mathbf{w}\rVert_q^q + C\sum_{i=1}^n \omega_i[\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)]^2 \tag{2.1}$$

where $q \in \{1,2\}$ and C denotes a penalty parameter.

When $q = 1$, it is called L1-regularized L2-loss LSVM, while when $q = 2$, it is called L2-regularized L2-loss LSVM. You could find more details related to Eq. (2.1) in Fan et al. (2008).

Eq. (1) can often be represented as another form

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \omega_i [\max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)]^2 + \lambda \|\mathbf{w}\|_q^q \tag{2.2}$$

Where λ has a relationship with C, $\lambda = \frac{1}{q \times C}$.

The decision function is

$$d(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^{\mathrm{T}} \mathbf{x}) \tag{2.3}$$

where sgn denotes the sign function, denoting the sign of a real number.

Notes:

- If the binary target is not a form of $\{+1, -1\}$, it should be mapped into $\{+1, -1\}$.

2.1.2 Multi-class classification

For the multi-class classification, LSVM has a general form of

$$\min_{\{\mathbf{w}^1, \cdots, \mathbf{w}^m\}} \sum_{i=1}^{n} \omega_i \sum_{j \neq y_i} [\max(0, 2 - (\mathbf{w}^{y_i} - \mathbf{w}^j)^T \mathbf{x}_i)]^2 + \lambda \sum_{j=1}^{m} \left\|\mathbf{w}^j\right\|_q^q \tag{2.4}$$

The decision function is

$$d(\mathbf{x}) = \arg\max_j (\mathbf{w}^j)^T \mathbf{x} \tag{2.5}$$

Notes:

- If the target is not a form of $\{1, 2, \cdots, m\}$, it should be mapped into $\{1, 2, \cdots, m\}$.
- For $m = 2$, let $\mathbf{w}^{+1} = -\mathbf{w}^{-1} = \mathbf{w}$ and $\lambda' = \frac{\lambda}{2}$, the optimization problem becomes

  $$\begin{aligned} &4 \sum_{i=1}^{n} \omega_i [\max(0, 1 - \mathbf{w}^T \mathbf{x}_i)]^2 + 2\lambda \|\mathbf{w}\|_q^q \\ &\propto \sum_{i=1}^{n} \omega_i [\max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)]^2 + \lambda' \|\mathbf{w}\|_q^q \end{aligned}$$

  This means that the binary classification problem is a special case of the multi-class classification problem.

- Eq. (2.4) originally comes from Eq. (2) of Weston and Watkins (1999). The main difference is that we use L2-loss while they use L1-loss.

2.2 Regression

Support vector regression solves the following primal problems

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \omega_i [\max(0, |y_i - \mathbf{w}^T \mathbf{x}_i| - \epsilon)]^2 + \lambda \|\mathbf{w}\|_q^q \tag{2.6}$$

where ϵ is a parameter to specify the sensitiveness of the loss. By default, $\epsilon = 0.1$.

The prediction function is

$$d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} \tag{2.7}$$

3. Parameter estimation

The problems (Eqs.2.2, 2.4 and 2.6) can be written as a general form

$$\min_{\mathbf{w}} f(\mathbf{w}) + g(\mathbf{w}) \tag{3.1}$$

where $f(\mathbf{w})$ denotes the fitting function corresponding to the first term of Eqs.(2.2), (2.4) and (2.6), while $g(\mathbf{w})$ denotes the penalty function corresponding to the second term $\left(\lambda\|\mathbf{w}\|_q^q\right)$.

We use ADMM algorithm (Zhong, 2014) to solve the optimization problems. ADMM denotes Alternating Direction Method of Multipliers algorithms, which is to solve large scale problems in terms of the volume of records and the number of variables. In addition, it is implemented in distributed computing environment, more specifically, the map-reduce environment.

If $f(\mathbf{w})$ can be separable w.r.t records, the form of ADMM for solving Eq. 3.1 can be written as,

$$\begin{array}{ll} \min_{x_i, \boldsymbol{z}} & \sum_{i=1}^{N} f_i(\mathbf{w}_i) + g(\boldsymbol{z}) \\ s.t. & \mathbf{w}_i - \mathbf{z} = \mathbf{0}, i = 1, \cdots, N. \end{array} \tag{3.2}$$

The steps of ADMM can be described as follows:

$$\begin{aligned} \mathbf{w}_i^{k+1} &= \arg\min_{\mathbf{w}_i} \left(f_i(\mathbf{w}_i) + (\rho/2)\left\|\mathbf{w}_i - \mathbf{z}^k + \mathbf{u}_i^k\right\|_2^2 \right) \\ \mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}} (g(\mathbf{z}) + (N\rho/2)\|\mathbf{z} - \overline{\mathbf{w}}^{k+1} - \overline{\mathbf{u}}^k\|_2^2) \\ \mathbf{u}_i^{k+1} &= \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1} \end{aligned} \tag{3.3}$$

where $\overline{\mathbf{w}}^{k+1} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{w}_i^{k+1}$ is the average of $\mathbf{w}_1^{k+1}, \cdots, \mathbf{w}_N^{k+1}$; Similarly, $\overline{\mathbf{u}}^k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{u}_i^k$.

In order to call ADMM algorithm, LSVM should prepare four pieces of information: optimization function, gradient function and Hessian function of $f(\mathbf{w})$, and initial values. Please note that for the large p situation (where the number of parameters p is greater than a threshold value of p_c), it does not need to provide the information of Hessian function.

3.1 Classification

Although the binary classification is a special case of the multi-class classification problem, we still estimate parameters for it rather than estimate them by solving the multi-class classification problem. The reasons are: (1) the number of parameters of the multi-class classification problem is twice of that of the binary classification problem, thus, it may lead to a slow convergence of the optimization method, especially for large p situation; (2) the initial values for the multi-class classification problem is not as good as those for the binary classification, because the multi-class classification is more complicated than the binary one and it is hard to obtain good initial values.

3.1.1 Binary classification

For the binary classification, $f_i(\mathbf{w}_i)$ is defined as

$$f_i(\mathbf{w}_i) = \sum_{\ell\in B_i} \omega_\ell [\max(0, 1 - y_\ell \mathbf{w}_i^T \mathbf{x}_\ell)]^2 \tag{3.4}$$

The gradient for $f_i(\mathbf{w}_i)$ is

$$\mathbf{s}_i = \sum_{\ell\in B_i^{\mathrm{sv}}} 2\omega_\ell(-y_\ell + \mathbf{w}_i^T \mathbf{x}_\ell)\mathbf{x}_\ell \tag{3.5}$$

The Hessian matrix for $f_i(\mathbf{w}_i)$ is

$$\mathbf{H}_i = \sum_{\ell\in B_i^{\mathrm{sv}}} 2\omega_\ell \mathbf{x}_\ell \mathbf{x}_\ell^T \tag{3.6}$$

where $B_i^{\mathrm{sv}} = \{\ell \in B_i | 1 - y_\ell \mathbf{w}_i^T \mathbf{x}_\ell > 0\}$ denotes the index set of support vectors.

For the small to medium p situations, the initial values can be the least square solution using all data

$$\mathbf{w}^0 = (\mathbf{X}^T \mathbf{\Omega} \mathbf{X} + \lambda \mathbf{I})^{-} \mathbf{X}^T \mathbf{y} \tag{3.7}$$

where $\mathbf{X} = [\cdots, \mathbf{x}_i, \cdots]^T$ denotes the data matrix, and $\mathbf{\Omega} = \mathrm{diag}(\omega_1, \cdots, \omega_n)$ denotes the weight matrix.

For the large p situation (where $p \geq p_c$), the initial values are computed as follows

$$\mathbf{w}^0 = {}^{\eta(\bar{\mathbf{x}}^{+1} - \bar{\mathbf{x}}^{-1})}\Big/_{\|\bar{\mathbf{x}}^{+1} - \bar{\mathbf{x}}^{-1}\|_2^2} \tag{3.8}$$

where $\eta > 2$, and we tentatively choose $\eta = 2.5$; $\bar{\mathbf{x}}^{+1}$ and $\bar{\mathbf{x}}^{-1}$ denote the weighted mean predictor vectors for class +1 and class -1, respectively.

Notes:

- To call ADMM, the settings of parameters are: $q = 2$ (L_2-penalty), $\lambda = 0.1$; $\rho = 1$; use function value convergence and parameter convergence; use default settings for other parameters of ADMM.
- If want to select variables, we will set $q = 1$ (L_1-penalty). In addition, if there are factor variables, we would use group penalty (or regularization) rather L_1-penalty. This means that all parameters related to a factor variable will bind together, and they will be selected or removed together.
- The threshold $p_c = 5000$ by default.

3.1.2 Multi-class classification

For the multi-class classification, $f_i(\mathbf{w}_i)$ is defined as

$$f_i(\mathbf{w}_i) = \sum_{\ell\in B_i} \omega_\ell \sum_{j\neq y_l} \left[\max\left(0, 2 - \left(\mathbf{w}_i^{y_\ell} - \mathbf{w}_i^j\right)^T \mathbf{x}_\ell\right)\right]^2 \tag{3.9}$$

The gradient for $f_i(\mathbf{w}_i)$ is

$$\mathbf{s}_i = \frac{\partial f_i(\mathbf{w}_i)}{\partial \mathbf{w}_i} = \begin{bmatrix} \vdots \\ \frac{\partial f_i(\mathbf{w}_i)}{\partial \mathbf{w}_i^a} \\ \vdots \end{bmatrix}, a \in [1, m] \tag{3.10}$$

where

$$\frac{\partial f_i(\mathbf{w}_i)}{\partial \mathbf{w}_i^a} = \begin{cases} \sum_{\substack{\ell,j \\ (\ell,j)\in B_i^{\mathrm{sv}}}} -2\omega_\ell \left[2 - \left(\mathbf{w}_i^{y_\ell} - \mathbf{w}_i^j\right)^T \mathbf{x}_\ell\right] \mathbf{x}_\ell & a = y_\ell \\ \sum_{\substack{\ell \\ (\ell,j)\in B_i^{\mathrm{sv}}}} 2\omega_\ell \left[2 - \left(\mathbf{w}_i^{y_\ell} - \mathbf{w}_i^j\right)^T \mathbf{x}_\ell\right] \mathbf{x}_\ell & a = j \end{cases} \tag{3.11}$$

and $B_i^{\mathrm{sv}} = \left\{(\ell,j) | 2 - \left(\mathbf{w}_i^{y_\ell} - \mathbf{w}_i^j\right)^T \mathbf{x}_\ell > 0, l \in B_i, j \in [1,m], y_\ell \neq j\right\}$, and those$\ell$’s belonging to B_i^{sv} which denotes the index set of support vectors.

The Hessian matrix for $f_i(\mathbf{w}_i)$ is

$$\mathbf{H}_i = \begin{bmatrix} \cdots & \vdots & \cdots \\ \cdots & \frac{\partial f_i(\mathbf{w}_i)}{\partial \mathbf{w}_i^a \partial \mathbf{w}_i^{b,T}} & \cdots \\ \cdots & \vdots & \cdots \end{bmatrix} \tag{3.12}$$

where

$$\frac{\partial f_i(\mathbf{w}_i)}{\partial \mathbf{w}_i^a \partial \mathbf{w}_i^{b,T}} = \begin{cases} \sum_{\substack{\ell,j \\ (\ell,j)\in B_i^{\mathrm{sv}}}} 2\omega_\ell \mathbf{x}_\ell \mathbf{x}_\ell^T & a = b = y_\ell \\ \sum_{\substack{\ell \\ (\ell,j)\in B_i^{\mathrm{sv}}}} -2\omega_\ell \mathbf{x}_\ell \mathbf{x}_\ell^T & a = y_\ell, b = j \text{ or } a = j, b = y_\ell \\ \sum_{\substack{\ell \\ (\ell,j)\in B_i^{\mathrm{sv}}}} 2\omega_\ell \mathbf{x}_\ell \mathbf{x}_\ell^T & a = b = j \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{3.13}$$

Note that for a record with target value y_l, we need to update the y_ℓ's portion of the gradient and the (y_ℓ, y_ℓ) block of the Hessian for all indices j, where (ℓ, j) belongs to B_i^{sv}; and for each such a pair we will also update the j's portion of the gradient and (j,j), (j, y_ℓ), and (y_ℓ, j) blocks of the Hessian.

The initial values can be calculated, no matter it’s large p situation or not, as

$$\mathbf{w}^0 = \begin{bmatrix} \vdots \\ \mathbf{w}^{a,0} \\ \vdots \end{bmatrix}, \text{where } \mathbf{w}^{a,0} = \frac{\eta \bar{\mathbf{x}}^a}{\|\bar{\mathbf{x}}^a\|_2^2} \tag{3.14}$$

3.2 Regression

For the regression, $f_i(\mathbf{w}_i)$ is defined as

$$f_i(\mathbf{w}_i) = \sum_{\ell \in B_i} \omega_l [\max(0, |y_\ell - \mathbf{w}_i^T \mathbf{x}_\ell| - \epsilon)]^2 \tag{3.15}$$

The gradient for $\mathbf{w}_i$ is

$$\mathbf{s}_i = \sum_{\ell \in B_i^{\mathrm{sv1}}} 2\omega_\ell(-y_\ell + \mathbf{w}_i^T \mathbf{x}_\ell + \epsilon)\mathbf{x}_\ell + \sum_{\ell \in B_i^{\mathrm{sv2}}} 2\omega_\ell(-y_\ell + \mathbf{w}_i^T \mathbf{x}_\ell - \epsilon)\mathbf{x}_\ell \tag{3.16}$$

The Hessian matrix for $\mathbf{w}_i$ is

$$\mathbf{H}_i = \sum_{\ell \in B_i^{\mathrm{sv1}} \cup B_i^{\mathrm{sv2}}} 2\omega_\ell \mathbf{x}_\ell \mathbf{x}_\ell^T \tag{3.17}$$

where $B_i^{\mathrm{sv1}} = \{\ell \in B_i | y_\ell - \mathbf{w}_i^T \mathbf{x}_\ell - \epsilon > 0\}$ and $B_i^{\mathrm{sv2}} = \{\ell \in B_i | -y_\ell + \mathbf{w}_i^T \mathbf{x}_\ell - \epsilon > 0\}$ denote the sets of support vectors.

For the small to medium p situations, the initial values can be the least square solution using all data

$$\mathbf{w}^0 = (\mathbf{X}^T\mathbf{\Omega}\mathbf{X} + \lambda\mathbf{I})^{-}\mathbf{X}^T\mathbf{\Omega}\mathbf{y} \tag{3.18}$$

For the large p situation, the initial values are computed as follows: generate a random vector $\widetilde{\mathbf{w}}$ with Gaussian distribution $N(\mathbf{0}_{p\times 1}, \mathbf{I}_{p\times p})$, then the initial values are

$$\mathbf{w}^0 = \left({\widetilde{\mathbf{w}}^T\mathbf{X}^T\mathbf{\Omega}\mathbf{y}} \Big/ {\lambda\|\widetilde{\mathbf{w}}\|_2^2 + \|\mathbf{X}\widetilde{\mathbf{w}}\|_{\mathbf{\Omega}}^2} \right)\widetilde{\mathbf{w}} \tag{3.19}$$

where $\|\mathbf{X}\widetilde{\mathbf{w}}\|_{\mathbf{\Omega}}^2 = \widetilde{\mathbf{w}}^T\mathbf{X}^T\mathbf{\Omega}\mathbf{X}\widetilde{\mathbf{w}}$.

Notes:

- The settings of parameters are the same to those given in Section 3.1.

4. Post-estimation statistics

For the task of classification, we provide the probability output; while for the task of regression, we provide the standard deviation.

4.1 Prediction

For the task of binary classification, the predicted category for a given $\mathbf{x}_\ell$ (not limit to the training records) is

$$\hat{y}_\ell = \mathrm{sgn}(\widehat{\mathbf{w}}^T\mathbf{x}_\ell) \tag{4.1}$$

where $\mathrm{sgn}(a) = \begin{cases} +1 & a > 0 \\ 0 & a = 0. \\ -1 & a < 0 \end{cases}$

Note that if $\widehat{\mathbf{w}}^T\mathbf{x}_\ell = 0$, then it is assigned to the majority class. If two classes have the same number of records, it is assigned the positive class.

For the multi-class classification, the predicted category for given $\mathbf{x}_\ell$ is given by

$$\hat{y}_\ell = \arg\max_j (\widehat{\mathbf{w}}^j)^T\mathbf{x}_\ell \tag{4.2}$$

Please note that if there are ties, it is assigned to the class with the maximal number of records. If there are still ties, it is assigned to the class with the smallest superscript.

For the task of regression, the predicted value for given $\mathbf{x}_\ell$ is

$$\hat{y}_\ell = \widehat{\mathbf{w}}^T\mathbf{x}_\ell \tag{4.3}$$

4.2 Performance measure

For the classification task, we will provide the percentage of total correct predictions of the model as well as the classification table, while for the regression task, we will provide the average square error of the model.

The process of calculating the percentage of total correct predictions of the model and the classification table is

(1) Suppose that $c(j,j')$ is the sum of the frequency for the observations whose actual target category is j (as row) and predicted target category is j' (as column), $j,j' = 1,\cdots,m$ (note that $m = 2$ for binary classification), then

$$c(j,j') = \sum_{\ell=1}^{n} \omega_\ell I(y_\ell = j, c(\boldsymbol{x}_\ell) = j')$$

where $I(\cdot)$ is indicator function and $c(\boldsymbol{x}_\ell)$ denotes the predicted category.

(2) Suppose that $p_{j,j'}$is the $(j,j')^{\text{th}}$ element of the classification table, which is row percentage, then

$$p_{j,j'} = \left(\frac{c(j,j')}{\sum_{k=1}^{m} c(j,k)}\right) \times 100\%$$

(3) The percentage of total correct predictions of the model is

$$p_{total} = \left(\frac{\sum_{j=1}^{m} c(j,j)}{\sum_{j=1}^{m}\sum_{j'=1}^{m} c(j,j\prime)}\right) \times 100\% \tag{4.4}$$

The average square error (ASE) for the regression can be calculated as

$$ASE = \frac{1}{\sum_{\ell=1}^{n} \omega_\ell} \sum_{\ell=1}^{n} \omega_\ell (y_\ell - \hat{y}_\ell)^2 \tag{4.5}$$

4.3 Probability

For the binary classification problem, we provide a probability model to approximate the posterior class probability. For a given $\mathbf{x}_\ell$ and $\hat{y}_\ell$, we have

$$\hat{p}_\ell = {}^{1}\Big/\left(1 + e^{-\hat{y}_\ell \hat{\mathbf{w}}^T \mathbf{x}_\ell}\right) \tag{4.6}$$

Notes

- We usually give the probability of being positive class, that is,

$$\hat{p}_\ell = {}^{1}\Big/\left(1 + e^{-\hat{\mathbf{w}}^T \mathbf{x}_\ell}\right)$$

- The probability model is not very accurate, thus it may not make sense to compute gain, lift, etc. based on the sorting probabilities.
- We do not use Platt (2000) method used by SVM node in SPSS Modeler (Tian and Zhong, 2007). The reason is that Platt method involves two additional parameters, which needs an iterative optimization method (Newton-Raphson). This means that multiple data passes are needed to obtain the estimation of parameters.

For the multi-class classification, the probability for y, $y = 1, \dots, m,$ can be calculated as

$$\Pr(y|\mathbf{x}_\ell) = {}^{1}\Big/\left(1 + \sum_{j=1, j\neq y}^{m} e^{(\mathbf{w}^j - \mathbf{w}^y)^T \mathbf{x}_\ell}\right) \tag{4.7}$$

Please note that we will provide a probability for each class, that is to say, we have m probabilities for each record.

4.4 Standard deviation

Lin and Wen (2004) pointed out that residuals $\hat{r}_\ell = y_\ell - \hat{\mathbf{w}}^T\mathbf{x}_\ell$ can be fit for a Laplace distribution with zero mean, which is given as follow

$$\text{Prob}(r) = \frac{1}{2\sigma} e^{-\frac{|r|}{\sigma}} \tag{4.8}$$

Here $\sigma > 0$ is a scale parameter.

Assume that $\hat{r}_i$ are independent, the scale parameter can be estimated by ML method, that is, the $\hat{\sigma}$ is

$$\hat{\sigma} = \frac{1}{\sum_{\ell=1}^{n} \omega_\ell} \sum_{l=1}^{n} \omega_\ell |\hat{r}_\ell| \tag{4.9}$$

But the ML method will be affected by some “very extreme” $\hat{r}_\ell$ and causes inaccurate estimation of σ. One improved method is to estimate the scale parameter by discarding $\hat{r}_\ell$ which exceed $\pm 5\hat{\sigma}$. The other method is to use median of $|\hat{r}_\ell|$ as an estimation of σ, especially when there is a large number of examples. The first method is accurate but needs two data passes, while the second method needs only one data pass but it is approximate. We let the software engineers to decide which method is used.

Thus, for any record $\mathbf{x}_\ell$, the confidence interval for the prediction of y_ℓ with $100(1-\alpha)\%$ confidence is given by

$$[\hat{y}_\ell \pm \eta_{1-\alpha/2}] \tag{4.10}$$

where η_p is the $(100p)^{\text{th}}$ percentile of the Laplace distribution with zero mean and standard deviation $\hat{\sigma}$.

References

[1] Zhong, W. (2014), Algorithm: ADMM, IBM SPSS Internal Document.

[2] Fan et al. (2008), “LIBLINEAR: A library for large linear classification”, Journal of Machine Learning Research, 9, 1871-1874.

[3] Platt, (2000), “Probabilistic outputs for support vector machines and comparison to regularized likelihood methods”: Advances in Large Margin Classifiers. Cambridge, MA.

[4] Tian, G. and Zhong, W. (2007), Algorithm: SVM, SPSS Internal Document.

[5] Lin C.J. and Wen R.C. Simple probabilistic predictions for support vector regression. Technical report, Department of Computer Science, National Taiwan University, 2004.

[6] Weston, J. and Watkins, C., Support vector machine for multi-class pattern recognition, ESANN’1999, 219-224.

Random Trees Modeling Algorithms

1. Introduction

Random Trees is a powerful new approach for strong (accurate) predictive models. It is comparable and sometimes better than other state-of-the-art methods in classification or regression problems.

Random Trees is an ensemble model consisting of multiple CART-like trees. Each tree grows on a bootstrap sample which is obtained by sampling the original data cases with replacement. Moreover, during tree growth, for each node the best split variable is selected from a specified smaller number of variables which are drawn randomly from the full set of variables. And each tree grows to the largest extent possible. There is no pruning. In scoring, random trees combines individual tree scores by majority voting (for classification) or average (for regression).

Because each tree model can be built independently, random trees are very suitable to be applied in distributed setting. However, a big challenge is to handle massive data, since building even a single tree is expensive in this case. There are several implementations which have addressed this issue. One implementation in Apache Mahout just partitions the data and builds trees on smaller data blocks. Clearly this method could result in weak and biased trees because data blocks could have biased distributions from the training data. Another implementation on Apache Spark follows Google’s PLANET implementation which can build single tree models efficiently on massive data. Spark has the ability to cache data in memory for interactive data analysis. This implementation has benefited from this ability greatly. For example, it can remember the last node that a case belongs to. This speeds up the process considerably since it does not need to pass large trees to executors any more.

Our implementation is based on Apache Hadoop framework. We also adopt Google’s PLANET implementation to build single trees. But unfortunately, Hadoop does not have the ability of caching data for interactive data analysis. We will have to resort to other solutions to achieve desired performance. Meanwhile, the challenging issues about large data, imbalance data, etc., will also be considered in our implementation.

In this chapter, we describe the algorithms used to build a random trees model under the map-reduce framework. In addition to generating the predictive solution, we also provide an enhanced set of evaluation and diagnostic features enabling insight, interactivity, and an improved overall user experience as required by the Analytic Catalyst and other applications.

The document is organized as follows. We first declare some general notes about algorithms, development, etc. Then we define the notations used in the document. In section 4, we present the general workflow of the random trees engine. Operations for data pre-processing are introduced in section 5, along with some summary statistics that are required for model building. Section 6 describes the key components in model building. In section 7, we present various measures used for model evaluation and model diagnostics, and they will be computed along with the process of model building. Insights and interestingness are also derived. Finally section 8 shows how to score new cases.

2. Notes

1. The Random Trees engine is implemented in a parallel distributed algorithm within Analytic Engine (AE), based on the map-reduce framework.

3. Notations

The following notations are used throughout the document unless otherwise stated:

Y	Dependent variable or target. If Y is categorical with J categories, its set of categories is given by $C = \{1, \dots, J\}$.
$X_m, m = 1, \dots, M$	Set of all predictor variables. If X_m is categorical with I_m categories, its categories are given by $D = \{1, \dots, I_m\}$.
$\mathcal{H} = \{x_{m,k}, y_k\}_{k=1}^{K}$	Complete set of training cases
$\mathcal{H}_q, q = 1, \dots, Q$	Bootstrap sample q, $q = 1, \dots, Q$
$\mathcal{H}(t)$	Cases that belong to node t
w_k	Analysis weight associated with case k
f_k	Frequency weight associated with case k. Non-integral positive value is rounded to its nearest integer.
$\mathrm{I}(a = b)$	Indicator function taking value 1 when $a = b$ and 0 otherwise.
$\pi(j), j = 1, \dots, J$	Priority probability of $Y = j$, $j = 1, \dots, J$
$p(j, t), j = 1, \dots, J$	Probability of a case in class j and node t
$p(t)$	Probability of a case in node t
$p(j\|t), j = 1, \dots, J$	Probability of a case in class j given that it falls into node t
$C(i\|j)$	Cost of miss-classifying a class j case as a class i case

4. General Workflow

The Random Trees engine builds random trees through several stages in sequence. In each stage, one or more map-reduce jobs will be launched. The general workflow is typically as follows.

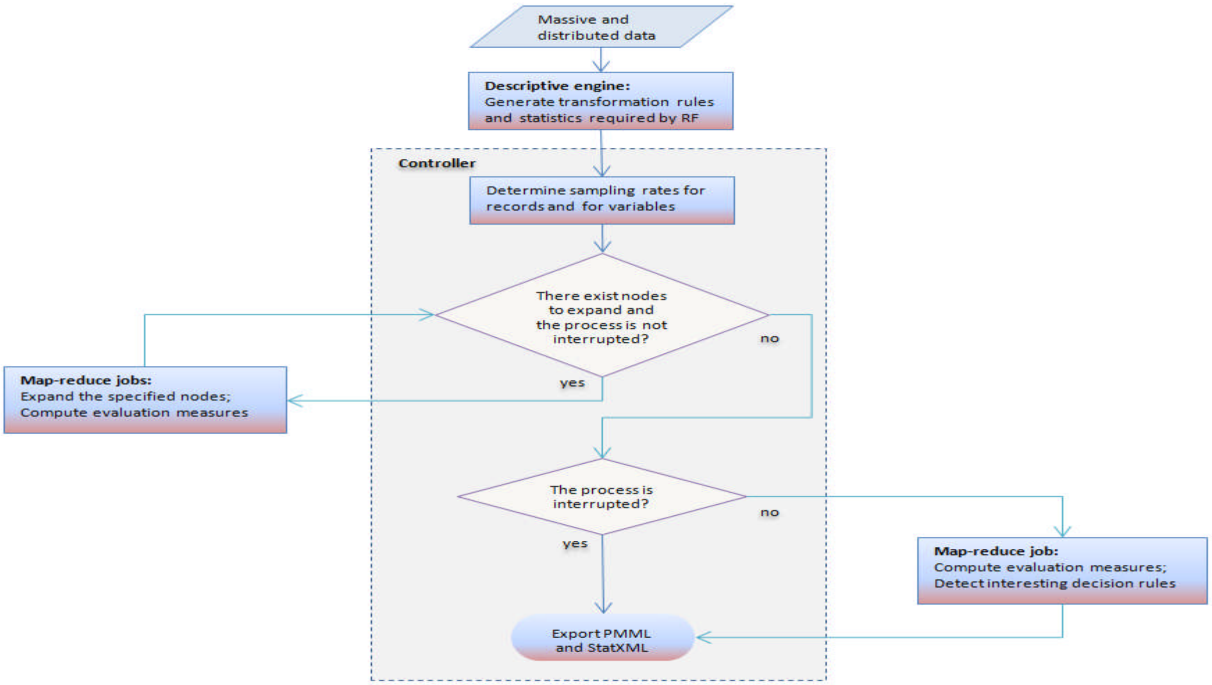

Massive and distributed data
Descriptive engine: Generate transformation rules and statistics required by RF
Controller
Determine sampling rates for records and for variables
There exist nodes to expand and the process is not interrupted?
no
yes
Map-reduce jobs: Expand the specified nodes; Compute evaluation measures
The process is interrupted?
no
yes
Map-reduce job: Compute evaluation measures; Detect interesting decision rules
Export PMML and StatXML

Figure 1. General workflow

5. Data Pre-processing

The Random Trees engine supports distributed data in column-based format. It requires at least one predictor that can be flag, ordinal categorical, nominal categorical, or continuous, and a single target that can be categorical or continuous. Flag or ordinal target is considered as categorical.

5.1. Filtering Variables

Based on the summary statistics produced by DE, the Random Trees engine will perform an initial analysis and determine the variables which are not useful for modelling.

Specifically, the following variables will be excluded.

#	Rule	Status	Comment
1	Identity variables	Required	
2	Constant variables	Required	
3	The percentage of missing values in any variable is larger than δ (default 0.7)	Required	
4	One category makes up the overwhelming majority of total population above a given percentage threshold δ (default 95%)	Required	
5	The number of categories of a categorical variable is larger than δ (default 49)	Required	
6	The absolute coefficient of variation of a continuous variable is smaller than δ (default 0.05)	Required	

7	Date/time variables	Required	

5.2. Transformations

The Random Trees engine supports frequency/analysis weights. Real frequency weights are rounded to the nearest integer.

System/user missing and invalid values are all considered as missing. If the target of a case is missing, this case will be ignored in the analysis. If all predictor variables of a case are missing, this case will also be ignored. If the analysis or frequency weight is missing, zero, or negative, the case is ignored. Otherwise, missing values will be imputed with mean for a continuous predictor or mode for a categorical predictor.

For each continuous predictor, a list of points $p_1, p_2, \ldots, p_{I_m}$ (in ascending order) is determined by the tiling method, i.e. equal-frequency binning, which has been implemented by the Descriptive engine. Notice that the transformation rule of equal-frequency binning will not be actually applied on the variables. Instead, it just provides the set of bin boundaries that will be checked as candidate splitting points. In default, we set the number of bins as 10, which means we will have 9 splitting points to check for each continuous variable.

Another transformation is to encode categorical predictors, that is, to map category values into integer. This transformation is particularly useful for string predictors.

5.3. Summary Statistics

The following summary statistics are required and computed by DE:

- Total number of cases
- Distribution of target categories (required if the option of imbalance classification is turned on)
- Interestingness indexes of the associations between predictors and target (required if the option of weighted sampling of predictors is turned on)

6. Building Base Trees

A specified number of base trees will be built in parallel. Firstly, we initialize each tree with a root node. Then a series of map-reduce jobs will be used to grow the trees, and each of the jobs will be responsible for expanding a particular set of tree nodes.

For a certain map-reduce job, we suppose that the involved trees are T_q, $q \in \mathbb{Q}$, where $\mathbb{Q}$ is the set of labels of involved trees, and the set of tree nodes to expand $E = \{t_{q,r} | q \in \mathbb{Q}, r \in R_q,\}$, where R_q is the set of node IDs in T_q which are to expand. For example, in the first map-reduce job, base trees T_q are only with root nodes and the set of tree nodes E contains root nodes as well.

Notice that when creating the root nodes $t_{q,0}$ we will have an initial estimation of the number of training cases, as follows.

$$|t_{q,0}| = \begin{cases} N_{j_m} * J, & \text{in case of imbalance classification} \\ N * \alpha, & \text{otherwise} \end{cases},$$

where j_m is the minority class, α is the ratio for under-bagging, and

$N = \sum_{k=1}^{K} f_k$,

$N_{j_m} = \sum_{k=1}^{K} f_k \mathrm{I}(y_k = j_m)$.

6.1. Generating Bootstrap Samples

Base trees are built on Q bootstrap samples. To generate bootstrap samples, cases will be sampled with replacement. But notice that frequencies will be produced on the fly for each case at the time when it is processed.

In a regular bootstrap sample, the sampling rate for each case k is f_k/N. Then the times replicated for case k will be $rv.binom(N * \alpha, f_k/N)$.

If the option of imbalance classification is turned on, random trees will be built on balanced bootstrap samples. We achieve this by adjusting the sampling rates specific for each target category. Suppose that the J target categories are with counts of $N_1, N_2, \dots, N_J$, respectively. Let $j_m = arg\min\{N_j\}$. Then for each case k with target category j, the sampling rate is f_k/N_j, and the times replicated for case k will be $rv.binom(N_{j_m}, f_k/N_j)$. This is equivalent to drawing a bootstrap sample from the minority category and drawing randomly the same number of cases, with replacement, from the other categories.

Denote the generated bootstrap frequencies as f_k^q, $k = 1,2,\dots,K, q = 1,2,\dots,Q$.

Notice that drawn bootstrap samples should be identical across different map-reduce jobs. This can be achieved by using the same random seeds across different jobs. But notice that the seeds should be different across mappers within a single job in order to get different bootstrap frequencies in each data split.

6.2. Defining <key, value> Pairs

Pairs of <key, value> indicate the minimal unit of tasks. They are defined and generated by Mappers, and passed to Reducers.

One definition of keys is by tree id, node id, and predictor id. Such keys are defined when the condition of in-memory building is not satisfied.

Given the set $E = \{t_{q,r} | q \in \mathbb{Q}, r \in R_q\}$, for each $t_{q,r}$, we randomly select M_t (default value is $\lfloor sqrt(M) \rfloor$ for classification and $\lfloor M/3 \rfloor$ for regression) predictors from the total set of predictors. If the option of weighted sampling is turned off, each predictor will be selected with equal probability. Otherwise, the selection probabilities will be $\frac{I_{index}^m}{\sum I_{index}^m}$, where I_{index}^m is the interestingness index corresponding to predictor m, as computed in section 5.3.

Notice that different random seeds should be used in order to make the selection of predictors different across nodes, but on the other hand the selection should be the same for each node across all mappers. For this purpose, we define random seed as a Hash function of tree id and node id.

Denote the set of selected predictors for $t_{q,r}$ as $X_{q,r}$. Then the keys are defined as triplets of$< q, r, m >$, $q \in \mathbb{Q}$, $r \in R_q$, and $m \in X_{q,r}$.

The value corresponding to a triplet key $< q, r, m >$ is a set of statistics used to determine the best splitting point. These statistics are summarized in appendix A.

The other form of keys is to define them as pairs of $< q, r >$, $q \in \mathbb{Q}$, and $r \in R'_q$, where R'_q denotes the set of nodes in tree q which satisfy the condition of in-memory building. The value corresponding to such keys is just the cases of interest including all predictors, target, analysis weight w_k, and frequency weight f_k^q.

Notice that if the option of correcting importance bias is turned on, the value will include two sets of statistics, one computed on training cases while the other computed on validation cases. These statistics can be computed by setting $\aleph(q,r) = \ell^s(q,r)$ and $\aleph(q,r) = \mathscr{g}^s(q,r)$ in the calculation of local statistics, where $\ell^s(q,r)$ and $\mathscr{g}^s(q,r)$ denote the training and validation cases in data split s which fall in node $t_{q,r}$, respectively.

Notice that the number of distinct cases in $\ell^s(q,r)$ and $\mathscr{g}^s(q,r)$ will also be computed.

6.3. Partitioning OOB Cases

Bootstrap sample is generated by sampling each case with replacement. That means some cases will be selected in the sample while the others are not included. We call the cases that are not included out-of-bag (OOB) cases. Clearly, OOB cases are defined for a particular bootstrap sample. Given multiple bootstrap samples, a case can be a training case for some trees, and it can be an OOB case for other trees.OOB cases will be partitioned into validation data and testing data if the option of correcting importance bias is turned on. To partition OOB cases, instead of generate separate partitions for each tree, we partition the data once into validataion and testing. In this way, each case can be scored by a complete set of trees that take it as an OOB case.

The size of the validation data could be very large if users take a small ratio α for under-bagging. This is likely to incur performance issue for tree growth, particularly for in-memory building as described later. In this regard, we propose the following procedure to limit its size:

1. Let $\widetilde{K} = K * \alpha * 63\%$ be the expected number of distinct cases used by a single bootstrap sample.
2. Suppose the initial sampling rate for validation data is β (default 50%). Then we let the actual sampling rate be $\widetilde{\beta} = MIN(\beta, \widetilde{K}/K)$.
3. For each case in the data, determine whether it is a validation case or a testing case according to the actual sampling rate.

The option of correcting importance bias is disabled for imbalance classification.

6.4. Processing Each Case

Each mapper handles a particular local data split, and cases in the data split are processed sequentially.

Each case can be used by any base tree with three roles, i.e. training, validation, or testing. If the case has a non-zero bootstrap frequency, it will be considered as a training case for the involved tree. Otherwise, it will be used as either validation or testing, depending on how the OOB sample is partitioned for the base tree.

For each training or validation case, we will pass the tree and find the node that the case falls into. Then we will update the training or validation values collected on the data split for related keys.

Specifically, the procedure is as follows.

```
ProcessingCase( )
Inputs:
   –  $T_q$, $q \in \mathbb{Q}$                      // Current base trees
   –  $t_{q,r}$, $r \in R_q, q \in \mathbb{Q}$             // Set of nodes to expand
   –  $X_{q,r}$, $r \in R_q, q \in \mathbb{Q}$             // Set of predictors selected for $t_{q,r}$
   –  $< key, value >_{k-1}$                 // Current <key, value> pairs
   –  Case $k$                           // A valid case
   –  Sampling rate(s)
      // Generating bootstrap frequencies, and partitioning OOB samples
Outputs:
   –  $< key, value >_k$                   // New <key, value> pairs
Procedure:
For each tree $T_q$, $q \in \mathbb{Q}$, repeat the follows:
1. Generate bootstrap frequencies $f_k^q$, as described in section 6.1;
2. If ($f_k^q > 0$),{
      Pass tree $T_q$, and get node $t_{q,r}$ that case $k$ falls in;
      If ($r \in R_q$), update $< key, value >_{k-1}$ with case $k$;
   }
3. Else if the option of importance correction is turned on,{
      Determine whether case $k$ is a validation case using sampling rate $\tilde{\beta}$;
      If yes,{
         Pass tree $T_q$, and get node $t_{q,r}$ that case $k$ falls in;
         If ($r \in R_q$), update $< key, value >_{k-1}$ with case $k$;
      }
   }
```

6.5. Splitting Nodes

We first introduce the splitting criterion and also the impurity measure which will be used to split nodes.

For a categorical target, the Gini impurity measure is

$$i(t_{q,r}) = \sum_{i,j} C(i|j) p(i|t_{q,r}) p(j|t_{q,r}),$$

where we let

$$p(j, t_{q,r}) = \frac{\pi(j) N_{w,j}(t_{q,r})}{N_{w,j}},$$

$$p(t_{q,r}) = \sum_j p(j, t_{q,r}),$$

$p(j|t_{q,r}) = \frac{p(j,t_{q,r})}{p(t_{q,r})}$.

And the splitting criterion is the decrease of the Gini impurity measure defined as

$\Delta i(p, t_{q,r}) = i(t_{q,r}) - P_L i(t_L) - P_R i(t_R)$,

where P_L and P_R are probabilities of sending a case to the left child node T_L and to the right child node T_R respectively. They are estimated as $P_L = \frac{p(t_L)}{p(t_{q,r})}$, $P_R = \frac{p(t_R)}{p(t_{q,r})}$.

Notice that when user-specified costs are involved, the altered priors can optionally be used to replace the priors. The altered prior is defined as $\pi'(j) = \frac{C(j)\pi(j)}{\sum_j C(j)\pi(j)}$, where $C(j) = \sum_i C(i|j)$.

For a continuous target, the splitting criterion $\Delta i(p, t_{q,r}) = i(t_{q,r}) - P_L i(t_L) - P_R i(t_R)$ is used with the Least Squares Deviation (LSD) impurity measures $i(t_{q,r}) = V(t_{q,r})$, where we let $P_L = \frac{N_w(t_L)}{N_w(t_{q,r})}$, $P_R = \frac{N_w(t_R)}{N_w(t_{q,r})}$.

For node $t_{q,r}$ and predictor X_m, we denote the set of splitting points as Ω_m. Then we find the best splitting point of X_m by $p_{X_m} = arg\ \max_{p\in\Omega_m}(\Delta i(p, t_{q,r}))$, and we let $\Delta i_{X_m} = \Delta i(p_{X_m}, t_{q,r}), m \in X_{q,r}$.

Selecting the best splitting point from Ω_m for a continuous or ordinal predictor is efficient because there are only a few points to check. But for a categorical predictor with many categories, the searching is nontrivial. Instead of making an exhaustive search, we find the splitting point using the optimal partitioning algorithm proposed by Chou (1991), as described in appendix B.

Notice that when computing p_{X_m} and Δi_{X_m}, we set $\aleph(q,r) = \ell^s(q,r)$, that is to compute them on training cases. The node will be split by the point $p^*_{X_m}$which corresponds to $\max_{m\in X_{q,r}}(\Delta i_{X_m})$.

If the option of correcting importance bias is turned on, we will also compute the splitting criterion on validation cases for the splitting point p_{X_m}. We denote this splitting criterion as $\Delta i'_{X_m}$. Then the node will be split by the point $p^*_{X_m}$which corresponds to $\max_{m\in X_{q,r}}(\Delta i'_{X_m})$. Here, the splitting criterion is recomputed for the OOB cases based on the splitting point obtained from the training data at each node. Furthermore, we will use only the OOB cases later to compute the importance measure. The principle here is similar to a conditional inference framework. The predictor selection criterion and splitting criterion are separated. Please refer to Deng (2011) for details.

6.6. In-Memory Building

As tree induction progresses, the size of the input dataset for many nodes becomes small enough to fit in memory. At any such point, rather than continuing tree induction using map-reduce jobs, we load the training cases into memory and complete sub-tree construction. We call this process in-memory building.

Suppose that the number of distinct training cases that fall in node $t_{q,r}$ is $|\ell(q,r)|$. Then whenever the condition $|\ell(q,r)| < K_{in}$ is satisified, in-memory building will be triggered, where K_{in} is a specified threshold with default 5,000. Mappers simply output all the cases that belong to the node $t_{q,r}$ as values in the <key,values> pairs according to the description in 6.2. Reducer that collects all the cases for the given node will perform all subsequent node splitting through the following steps.

```
InMemoryBuilding( )
Inputs:
   –  $\ell(q,r)$                    // Training cases that fall in node $t_{q,r}$
   –  $\mathscr{g}(q,r)$                    // Required to correct importance bias
   –  $T_q$                          // Current base tree $T_q$
Outputs:
   –  $T_q$                          // Updated base tree $T_q$
Procedure:
A tree is grown starting from node $t_{q,r}$ by repeatedly using the following steps on
each node:
1. Randomly select $M_t$ predictors from the total set of predictors;
// Using simple or weighted sampling depending on the setting.
2. Find the best split for each selected predictor using data $\ell(q,r)$;
// For each continuous predictor, rather than checking a limited number of
// points, we sort and check all its values from the smallest to the largest.
3. If the option of correcting importance bias is turned on, recompute the
splitting criterion for each predictor's best split using data $\mathscr{g}(q,r)$;
4. Among the best splits found in step 2, choose the one that maximizes the
splitting criterion;
5. Split the node using its best split found in step 4 if the stopping rules 1-4
in section 6.7 are not satisfied;
```

6.7. Stopping Rules

Stopping rules control if the tree growing process should be stopped or not. The following stopping rules are used:

1. If a node becomes pure; that is, all cases in a node have identical values of the target variable, the node will not be split.
2. If all cases in a node have identical values for each selected predictor, the node will not be split.
3. If the current tree depth reaches the user-specified maximum tree depth limit value, the node will not be split.
4. If the split of a node results in a child node whose node size is less than the user-specified minimum child node size value, the node will not be split.
5. If the number of nodes in the current tree exceeds the maximum number (default 10,000), the involved tree will stop growing.
6. If the accuracy of the random trees is not improved any more, the modeling process will stop. The accuracy measure is R-square for regression, classification accuracy for regular classification, and Gmean for imbalance classification.

The following procedure is used to implement stopping rule 6:

1. Let $\delta = 1\%$ and trials=10;
2. Let the ensemble contain the first tree T_1 that has grown completely, and denote its accuracy as Acc;
3. Let Count=0 and BestAcc=Acc;
4. Continue to check if there are new trees that have grown completely. If yes, suppose the new trees are $T_{q_1}, T_{q_1+1}, \dots, T_{q_2}$;

```
5.  For $i$=$q_1$: $q_2$, {
        Add the $i$th tree to the ensemble, and compute the new accuracy $Acc'$;
        Count++;
        If $Acc'$>BestAcc, { BestAcc=$Acc'$; BestPos=Count;}
        If Count==trials, {
            If (BestAcc-$Acc$)/$Acc$> $\delta$, {
                $Acc$=BestAcc;
                Count=Count-BestPos;
            }
            Else, return the ensemble with the best accuracy;
        }
    }
6.  Go to step 4;
```

In random trees, we allow base trees with much larger depth, e.g. 10. The minimum node size is one for classification, and five for regression. Alternatively, users can also set the minimum node size by percentage values, say one percent with respect to the root node.

6.8. Controller Design

The controller maintains a set of tree nodes that need to be expanded. In particular, we use a stack to manage these nodes in order to support stepwise random trees building. The controller schedules a series of map-reduce jobs off of the stack until the stack is empty. Each job is responsible for expanding a specified number of nodes. When a job is finished, the stack is updated with the new nodes that can now be expanded. Notice that when some nodes are expanded by in-memory building, no updates are made to the stack because tree induction at such nodes is complete.

Specifically, the controller does as follows:

1. Initialize the stack to be empty.
2. Push the root nodes belonging to tree $T_Q, T_{Q-1}, \ldots, T_1$ into the stack respectively.
3. Let N_E nodes off the stack. If there are multiple data splits, run a mixture pattern of map-reduce job to expand the nodes; Otherwise, run a task parallel pattern of map-reduce job. See sections 6.8.1, 6.8.2, and 6.8.3 for details.
4. Meanwhile, checking if there are new evaluation measures available. If yes, check stopping rule 6. If the rule is satisfied, go to step 10. Otherwise, update the best ensemble model and report particular evaluation measures.
5. For some new nodes, if the involved trees satisfy stopping rule 5, such new nodes will not enter the stack, and the involved trees are considered to be fully grown.
6. For remaining new nodes, we sort them by tree labels, $q = 1,2, \ldots, Q$. And push the nodes belonging to tree $T_Q, T_{Q-1}, \ldots, T_1$ into the stack respectively.
7. Check if there are a specified number of new trees (default 5) that have been fully grown. If yes, we will add the new trees one by one into the current ensemble model, and compute evaluation measures for the new ensemble models by launching a map-reduce job that runs separately and in parallel with the next job for tree growth.
8. Repeat step 3 until the stack is empty.
9. If the process is interrupted, return the ensemble model that consists of all fully grown trees.

10. Perform post-modelling analysis, including model evaluation if there are new trees generated but not evaluated, and model interpretation.

Model evaluation measures and interpretations will be described below in section 7.

6.8.1. Mixture Pattern of Map-Reduce Job

Mixture pattern of map-reduce job, as illustrated in Figure 2, is a mixture of data parallel and task parallel jobs.

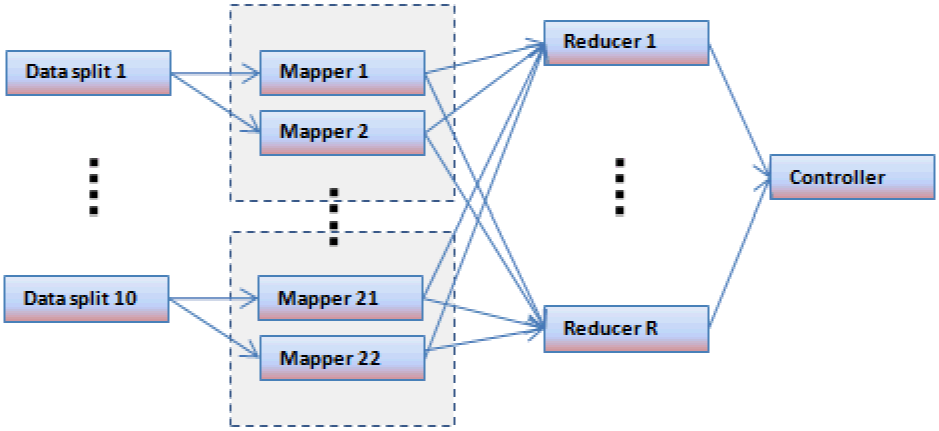


Figure 2. An example of mixture pattern of map-reduce job

In this example, mapper 1 and mapper 2 share the same data split, but they are fed with different settings. In our case, we let them generate and handle different bootstrap samples or trees.

Given the trees $T_q, q \in \mathbb{Q}$, that are involved in a certain job for tree growth, and the set of tree nodes to expand $E = \{t_{q,r} | r \in R_q, q \in \mathbb{Q}\}$, we apply the following procedure to allocate the trees to mappers that are working on the same data split:

```
1.  Sort trees $T_q$ according to tree labels in ascending order. Suppose the sorted trees
    are $T_{(1)}, T_{(2)}, \ldots, T_{(|\mathbb{Q}|)}$, and the sets of nodes to expand are $R_{(1)}, R_{(2)}, \ldots, R_{(|\mathbb{Q}|)}$;
2.  Let $MIN_m = 20$ and $MIN_n = 10$;
    // $MIN_m$ is the minimum number of mappers, default 20
    // $MIN_n$ is the minimum number of nodes handled by a mapper, default 10
3.  Get the number of data spits $S$;
4.  If $S < MIN_m$, {
        Compute $num = \min\left(\left\lceil \frac{MIN_m}{S} \right\rceil, \left\lfloor \frac{|E|}{MIN_n} \right\rfloor, \left\lfloor \frac{|E|}{|R_{(1)}|} \right\rfloor\right)$;
        Compute $Node_{min} = \frac{|E|}{num}$;
        Initialize $j = 1$ and $n_e = 0$;
        For $i$ in $1{:}\,|\mathbb{Q}|$, {
            Assign the $i$th tree to the $j$th mapper;
            Compute $n_e = n_e + |R_{(i)}|$;
            If $(n_e \geq Node_{min})$ and $i \neq |\mathbb{Q}|$, {
                Let $j = j + 1$;
                $n_e = 0$;
            }
        }
        Let $j$ be the number of mappers for each data split;
    }
    Else, assign all trees to every mapper;
    // A data parallel pattern of map-reduce job will be used for tree growth
```

6.8.2. Task Parallel Pattern of Map-Reduce Job

If there is only one data split, task parallel pattern of map-reduce jobs will be used to build trees. In each job, we have multiple mappers and each mapper builds a tree on a replicate of the data split.

The mapper first generates a particular bootstrap sample, and then it builds a tree on the sample in a way that is similar to the option of in-memory building. OOB cases will be partitioned as usual if necessary.

The number of mappers in each job is determined by the running environment, and the maximum number of mappers equals to the total number of base trees.

6.8.3. Selecting Nodes to Expand

In a task parallel pattern of job, the number of nodes off the stack equals to the number of mappers MIN_m.

For a mixture pattern of job, we select the nodes as follows:

1. Suppose the first node in the current stack belongs to tree T_q. We let all the following nodes in the stack that belong to tree T_q off the stack. Denote the number of such nodes as N_E.
2. Repeat the follows if $N_E < MAX_n$,
   // MAX_n is the maximum number of nodes to expand by a single job, default 100 (needs to tune),
   a) Let the first node in the current stack belongs to tree T_i. We let all the following nodes in the stack that belong to tree T_i off the stack.
   b) Update N_E.

The maximum number of nodes to expand by a single job is a setting which could be deployed with respect to concrete clusters. Basically we can set much higher maximum numbers for clusters with high computing capability.

Note that the procedure described above ensures that each tree grows in a width-first way. This point is important to the stopping rule of limiting the total number of nodes in a tree.

7. Model Evaluation and Insights

Suppose the ensemble model under evaluation consists of base trees T_q, $q = 1,2, \dots, Q$. Then we let each case in the testing partition of the OOB samples traverse the corresponding tree(s), and take the final prediction of a case as the combination of individual predictions by average or voting.

For convenience, we summarize the notations used for computing evaluation measures as follows.

$T_q, q = 1,2, \ldots, Q$	Q trees form an ensemble model to evaluate.
$\mathcal{L}_q$	The testing partition corresponding to the qth tree.
$\hat{y}_k^q$	The prediction of the qth tree on case k, $k \in \mathcal{L}_q$.
$\hat{y}_k$	The prediction of the ensemble model on case k, $k = 1,2, \ldots, K$.
$\mathcal{T}_k$	The set of trees that take case k as a testing case.

7.1. Evaluation Measures

7.1.1. Classification Model Evaluation

For a classification model, we compute

$$Acc = \frac{1}{N} \sum_{k=1}^{K} f_k \mathrm{I}(\hat{y}_k = y_k),$$

where $\hat{y}_k = arg\ \max_j \sum_{q \in \mathcal{T}_k} \mathrm{I}(\hat{y}_k^q = j)$, breaking ties arbitrarily.

Moreover, we compute the classification table. Suppose j is one of the observed category, and j^* is one of the predicted category, then the count of cell $< j^*, j >$ in the classification table is computed

$$C_{<j^*,j>} = \sum_{k=1}^{K} f_k \mathrm{I}(\hat{y}_k = j^*\ and\ y_k = j), j^* = 1,2, \ldots, J, j = 1,2, \ldots, J.$$

Note that if the option of imbalance classification is turned on, the evaluation measures above will not be computed.

7.1.2. Regression Model Evaluation

For a regression model, we compute

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^{K} f_k (y_k - \hat{y}_k)^2},$$

where $\hat{y}_k = \frac{1}{|\mathcal{T}_k|} \sum_{q \in \mathcal{T}_k} \hat{y}_k^q$. If $\mathcal{T}_k$ is empty, case k will be ignored.

Moreover, we compute

$$Rsquare = 1 - \frac{\sum_{k=1}^{K} f_k (y_k - \hat{y}_k)^2}{\sum_{k=1}^{K} f_k (y_k - \bar{y})^2},$$

where $\bar{y} = \frac{1}{N} \sum_{k=1}^{K} f_k y_k$.

7.1.3. Imbalance Classification Model Evaluation

If the option of imbalance classification is turned on, we will compute some measures that are specific to imbalance classification.

For target class j, we compute true positive rate, i.e. recall rate,

$$TPR_j = \frac{C_{<j,j>}}{\sum_i^J C_{<i,j>}}, j = 1, \dots, J.$$

Then we compute G-mean

$$Gmean = \left(\prod_{j=1}^J TPR_j\right)^{1/J}.$$

Notice that any class whose recall rate is constant zero across groups will be excluded from the calculation of the G-mean measure, and the number of J in the formula will be adjusted accordingly.

7.2. Interpretation and Insights

7.2.1. Gini Importance

Every time a split of a node is made on predictor X_m the Gini impurity criterion for the two descendent nodes is less than the parent node. Adding up the Gini decreases for each individual predictor over all trees in the trees gives a fast predictor importance measure that is often very consistent with the permutation importance measure.

Denote $Imp(X_m, T_q)$ as the importance of predictor X_m at tree T_q, then

$$Imp(X_m, T_q) = \sum_{t \in T_q} \Delta i(p^*_{X_m}, t) \, ,$$

where $p^*_{X_m}$ denotes the splitting point used by node t. Note that if predictor X_m is not the splitting variable, the corresponding Gini decrease from node t will be zero.

The Gini importance of predictor X_m is

$$Imp(X_m) = \sum_{q=1}^Q Imp(X_m, T_q).$$

Alternatively, the importance values can be normalized relative to the predictor having the largest measure of importance. That is,

$$\widetilde{Imp}(X_m) = \frac{Imp(X_m)}{\max_m(Imp(X_m))}.$$

Notice that if the option of correcting importance bias is turned on, the Gini decreases $\Delta i(p^*_{X_m}, t)$ will be computed using the validation OOB cases, as described in section 6.5.

7.2.2. Interesting Decision Rules

Random trees is formed by multiple decision trees, and each of them consists of tree nodes that represent decision rules. Since the rules will work together as a committee in scoring, it is not easy to interpret the results by individual rules. But on the other hand, the number of nodes or rules could be very large in random trees. This also provides the potency to dig out some interesting rules for the purpose of model interpretation.

For convenience, the following notations are defined.

I_t	Set of candidate nodes from which to detect interesting decision rules
δ	Threshold of minimal support of candidate nodes, default 1000
n_I	Number of interesting decision rules to report, default 5
δ_I	Interestingness threshold of filtering interesting decision rules to report, default 0.9
$A(t)$	Event that the prediction of random trees is correct on the data group determined by node t
$\bar{A}(t)$	Event that the prediction of random trees is wrong on the data group determined by node t
$B(t)$	Event that the prediction of node t is correct on the data group determined by node t
$P(\cdot)$	Probability of an event

Interesting decision rules are defined as those which have high prediction accuracy and also high agreement with the predictions of random trees. Clearly, such rules can be used to interpret random trees predictions. Specifically, the following procedure is used for the detection.

1. Identify the set of candidate interesting nodes I_t.
   a) Compute and save the count of testing cases for each leaf node in the job of model evaluation.
   b) Collapse any pair of nodes into their parent node if they have the same parent and both of their counts of testing cases are less than δ.
   c) Let I_t be the set of remaining leaf nodes whose counts of test cases are not less than δ.
2. For each node t in I_t, obtain its node assignment and let$P(B(t))$ be the prediction accuracy of node t on the data group determined by node t.
3. Launch a map-reduce job and for each node t in I_t,
   a) Let $P(A(t))$ be the prediction accuracy of random trees (not collapsed) on the data group determined by node t. Clearly, $P\left(\bar{A}(t)\right) = 1 - P(A(t))$.
   b) Let $P(A(t)B(t))$ be the ratio of cases that are predicted correctly by both random trees and node t to the total cases in the data group determined by node t.
   c) Let $P(\bar{A}(t)\bar{B}(t))$ be the ratio of cases that are predicted wrong by both random trees and node t to the total cases in the data group determined by node t.
   d) Compute the interestingness index $I_{index}(t) = P(A(t)) * P(B(t)) * (P(A(t)B(t)) + P(\bar{A}(t)\bar{B}(t)))$.
4. Report the top n_I nodes with the highest interestingness index $I_{index}(t)$; Optionally, users can select to report all nodes whose interestingness index is larger than δ_I.

8. Random Trees Scoring

8.1. Node Assignment

Suppose a random trees model consists of trees T_q, $q = 1,2, \dots, Q$. An assignment (also called action or decision) is computed for each node in the trees. To predict the target value for an incoming case, we first find in which terminal nodes it falls, and then we combine the assignments of these terminal nodes for the final prediction.

For any node t, let d_t be the assignment given to node t,

$$d_t = \begin{cases} j^*(t), & Y \text{ is categorical} \\ \bar{y}(t), & Y \text{ is continuous} \end{cases}$$

$j^*(t) = arg\min_i \sum_{\mathrm{j}} C(i|j)p(j|t),$

$\bar{y}(t) = \frac{\sum_{k\in\mathcal{H}(t)} w_k f_k^q y_k}{N_w(t)}.$

If there is more than one category j that achieves the minimum, choose $j^*(t)$ to be the smallest such j for which $N_{f,j}(t) = \sum_{k\in\mathcal{H}(t)} f_k^q \mathrm{I}(y_k = j)$ is greater than 0, or just the smallest j if $N_{f,j}(t)$ is zero for all of them.

8.2. Case Assignment

Given a case k, we first compute the score from tree T_q as $\hat{y}_k^q$, that is, the assignment of the terminal node in which the case fall. Then we combine the individual scores as

$$\hat{y}_k = \begin{cases} arg\ \max_j \sum_{q=1}^{Q} \mathrm{I}(\hat{y}_k^q = j)\,, & Y \text{ is categorical} \\ \quad\frac{1}{Q}\sum_{q=1}^{Q} \hat{y}_k^q\,, & Y \text{ is continuous} \end{cases}.$$

If the target variable is categorical, for each target category j, a confidence value will be calculated as

$\hat{p}_k(j) = \frac{\sum_{q=1}^{Q} \mathrm{I}(\hat{y}_k^q = j)}{Q}.$

Note that trees with null predictions will not be counted in case assignment.

8.3. Predictor Contribution

Predictor contribution is an evaluation of the influence of each predictor on the model prediction for an individual case. Please see Kuz'min (2011) and Palczewska (2013) for more details.

8.3.1. Regression Predictor Contribution

Each tree node, except the root node, has an associated rule according to which cases fall into this node. The difference between mean values in the current and parent nodes represents a local increment of contribution of the corresponding predictor, which is included in the rule of this node.

We let $LS_{m,t} = \bar{y}(t) - \bar{y}(t_{parent})$, where t_{parent} denotes the parent node of node t. Then the contribution of predictor X_m on the prediction of case k is

$$S_{k,m} = \frac{\frac{1}{Q}\sum_{t\in\Theta_m} LS_{m,t}}{\hat{y}_k - \bar{y}_{intercept}}$$

where Θ_m is the set of nodes in all trees of the trees, which contain case k and have predictor X_m in their rule, and

$\bar{y}_{intercept} = \frac{1}{Q}\sum_{q=1}^{Q} \bar{y}(t_{root}^q),$

where t_{root}^{q} is the root node of tree T_q.

In default, we report the top 3 predictors which have the largest contributions.

8.3.2. Classification Predictor Contribution

To present the predictor contribution procedure for a classification model, we need a probabilistic interpretation of the trees prediction process.

Let e_j be a J-dimensional vector with 1 at position j, and 0 otherwise. If tree T_q predicts that case k belongs to class j, then we write $\hat{Y}_k^q = e_j$. The prediction of the random trees for case k is

$$\hat{Y}_k = \frac{1}{Q}\sum_{q=1}^{Q} \hat{Y}_k^q.$$

We let $\bar{Y}(t)$ be a J-dimensional vector whose jth coordinate, $j = 1,2,\dots,J$, is defined as $p(j|t)$. Then, we define local contribution as $LS_{m,t} = \bar{Y}(t) - \bar{Y}(t_{parent})$. The overall contribution of predictor X_m on the prediction of case k is

$$S_{k,m} = \frac{\frac{1}{Q}\sum_{t\in\Theta_m} LS_{m,t}}{\hat{Y}_k - \bar{Y}_{intercept}},$$

where $\bar{Y}_{intercept} = \frac{1}{Q}\sum_{q=1}^{Q} \bar{Y}(t_{root}^{q})$, and the contributions are computed coordinate-wise.

Suppose the predicted target label $\hat{y}_k = j$. Then in default, we report the top 3 predictors which have the largest contributions at position j in $S_{k,m}$.

Appendix A. Computing Statistics

A.1. Local Statistics

Specifically, for a categorical predictor and a categorical target, the local statistics collected by a mapper on data split s will be

$$W_{i,j}^{s} = \sum_{k\in\aleph(q,r)} w_k\, f_k^q \mathrm{I}(x_{m,k} = i \text{ and } y_k = j),\ i = 1,\dots,I_m,\ j = 1,\dots,J,$$

where $\aleph(q,r)$ denotes the cases of interest in the data split that fall in node $t_{q,r}$. These cases may be training $\ell^s(q,r)$ or validation $\mathscr{g}^s(q,r)$ cases in particular scenarios.

For a categorical predictor and a continuous target, the statistics are

$$W_i^s = \sum_{k\in\aleph(q,r)} w_k\, f_k^q \mathrm{I}(x_{m,k} = i),\ i = 1,\dots,I_m,$$

$$\bar{Y}_i^s = \frac{\sum_{k\in\aleph(q,r)} w_k f_k^q y_k \mathrm{I}(x_{m,k}=i)}{W_i^s},\ i = 1,\dots,I_m,$$

$$V_i^s = \frac{\sum_{k\in\aleph(q,r)} w_k f_k^q \left(y_k-\bar{Y}_i^s\right)^2 \mathrm{I}(x_{m,k}=i)}{W_i^s},\ i = 1,\dots,I_m.$$

For a continuous predictor X_m, suppose the splitting points are $p_1, p_2, \dots, p_{I_m}$ (in ascending order). Then if the target is categorical, the statistics will be

$$W_{p_i,j}^{s} = \sum_{k\in\aleph(q,r)} w_k\, f_k^q \mathrm{I}(x_{m,k} \le p_i \text{ and } y_k = j),\ i = 1,\dots,I_m,\ j = 1,\dots,J,$$

$$W_{\cdot j}^{s} = \sum_{k\in\aleph(q,r)} w_k\, f_k^q \mathrm{I}(y_k = j),\ j = 1,\dots,J.$$

For a continuous predictor and a continuous target, the statistics are

$$W_{p_i}^s = \sum_{k\in\aleph(q,r)} w_k\, f_k^q \mathrm{I}(x_{m,k} \le p_i),\ i = 1,\dots,I_m,$$

$$\bar{Y}_{p_i}^s = \frac{\sum_{k\in\aleph(q,r)} w_k f_k^q y_k \mathrm{I}(x_{m,k}\le p_i)}{W_{p_i}^s},\ i = 1,\dots,I_m,$$

$$V_{p_i}^s = \frac{\sum_{k\in\aleph(q,r)} w_k f_k^q \left(y_k-\bar{Y}_{p_i}^s\right)^2 \mathrm{I}(x_{m,k}\le p_i)}{W_{p_i}^s},\ i = 1,\dots,I_m,$$

$$W^s = \sum_{k\in\aleph(q,r)} w_k\, f_k^q,$$

$$\bar{Y}^s = \frac{\sum_{k\in\aleph(q,r)} w_k f_k^q y_k}{W^s},$$

$$V^s = \frac{\sum_{k\in\aleph(q,r)} w_k f_k^q (y_k-\bar{Y}^s)^2}{W^s}.$$

A.2. Global Statistics

Local statistics computed on data splits are merged into global statistics as follows.

For a categorical predictor and a categorical target,

$W_{i,j} = \sum_s W_{i,j}^s,\ i = 1, \dots, I_m,\ j = 1, \dots, J.$

For a categorical predictor and a continuous target,

$W_i = \sum_s W_i^s,\ i = 1, \dots, I_m,$

$\bar{Y}_i = \sum_s \frac{W_i^s}{W_i} \bar{Y}_i^s,\ i = 1, \dots, I_m,$

$V_i = \sum_s \frac{W_i^s}{W_i} V_i^s + \sum_s \frac{W_i^s}{W_i} (\bar{Y}_i^s - \bar{Y}_i)(\bar{Y}_i^s + \bar{Y}_i),\ i = 1, \dots, I_m.$

For a continuous predictor and a categorical target,

$W_{p_i,j} = \sum_s W_{p_i,j}^s,\ i = 1, \dots, I_m,\ j = 1, \dots, J,$

$W_{\cdot j} = \sum_s W_{\cdot j}^s,\ j = 1, \dots, J.$

For a continuous predictor and a continuous target,

$W_{p_i} = \sum_s W_{p_i}^s,\ i = 1, \dots, I_m,$

$\bar{Y}_{p_i} = \sum_s \frac{W_{p_i}^s}{W_{p_i}} \bar{Y}_{p_i}^s,\ i = 1, \dots, I_m,$

$V_{p_i} = \sum_s \frac{W_{p_i}^s}{W_{p_i}} V_{p_i}^s + \sum_s \frac{W_{p_i}^s}{W_{p_i}} \left(\bar{Y}_{p_i}^s - \bar{Y}_{p_i}\right)\left(\bar{Y}_{p_i}^s + \bar{Y}_{p_i}\right),\ i = 1, \dots, I_m,$

$W = \sum_s W^s,$

$\bar{Y} = \sum_s \frac{W^s}{W} \bar{Y}^s,$

$V = \sum_s \frac{W^s}{W} V^s + \sum_s \frac{W^s}{W} (\bar{Y}^s - \bar{Y})(\bar{Y}^s + \bar{Y}).$

The statistics above will be computed with $\aleph(q, r) = \ell^s(q, r)$, and $\aleph(q, r) = \mathscr{g}^s(q, r)$ if necessary.

Moreover, if the condition of in-memory building is satisfied, we will get

$\ell(q, r) = \bigcup_s \ell^s(q, r),$

$\mathscr{g}(q, r) = \bigcup_s \mathscr{g}^s(q, r).$

A.3. Splitting Points and Statistics

For a continuous predictor X_m, the set of splitting points Ω_m consists of $p_1, p_2, \dots, p_{I_m}$ (in ascending order), which are determined by the tiling method, i.e. equal-frequency binning.

For node $t_{q,r}$ and each splitting point p_i, if the target is categorical, we have

$$N_{w,j} = \sum_{k \in \mathcal{H}_q} w_k f_k^q \mathrm{I}(y_k = j),$$

$$N_{w,j}(t_{q,r}) = W_{\cdot j},$$

$$N_{w,j}(t_L) = W_{p_i,j},$$

$$N_{w,j}(t_R) = W_{\cdot j} - W_{p_i,j},$$

where t_L and t_R denote the left child and the right child split by point p_i, respectively.

While if the target is continuous, we have

$$N_w(t_{q,r}) = W,$$

$$N_w(t_L) = W_{p_i},$$

$$N_w(t_R) = W - W_{p_i},$$

$$\bar{Y}(t_L) = \bar{Y}_{p_i},$$

$$\bar{Y}(t_R) = \frac{W\bar{Y} - W_{p_i}\bar{Y}_{p_i}}{W - W_{p_i}},$$

$$V\left(t_{q,r}\right) = V,$$

$$V(t_L) = V_{p_i},$$

$$V(t_R) = \frac{WV - N_w(t_L)V(t_L) - N_w(t_L)(\bar{Y}(t_L) - \bar{Y})(\bar{Y}(t_L) + \bar{Y}) - N_w(t_R)(\bar{Y}(t_R) - \bar{Y})(\bar{Y}(t_R) + \bar{Y})}{N_w(t_R)}.$$

For an ordinal categorical predictor X_m with I_m categories, splitting points just fall between two consecutive categories. While for a nominal categorical predictor X_m with I_m categories, the set of splitting points Ω_m is the power set of the I_m categories. Suppose that one of the splitting points p corresponds to a set of predictor categories C_p. Then if the target is categorical, we have

$$N_{w,j} = \sum_{k \in \mathcal{H}_q} w_k f_k^q \mathrm{I}(y_k = j),$$

$$N_{w,j}(t_{q,r}) = \sum_i W_{i,j},$$

$N_{w,j}(t_L) = \sum_{i \in C_p} W_{i,j}$,

$N_{w,j}(t_R) = N_{w,j}(t_{q,r}) - N_{w,j}(t_L)$.

While if the target is continuous, we have

$N_w(t_{q,r}) = \sum_i W_i$,

$N_w(t_L) = \sum_{i \in C_p} W_i$,

$N_w(t_R) = N_w(t_{q,r}) - N_w(t_L)$,

$\bar{Y} = \sum_i \frac{W_i}{N_w(t)} \bar{Y}_i$,

$\bar{Y}(t_L) = \sum_{i \in C_p} \frac{W_i}{N_w(t_L)} \bar{Y}_i$,

$\bar{Y}(t_R) = \sum_{i \notin C_p} \frac{W_i}{N_w(t_R)} \bar{Y}_i$,

$V\left(t_{q,r}\right) = \sum_i \frac{W_i}{N_w(t_{q,r})} V_i + \sum_i \frac{W_i}{N_w(t_{q,r})} (\bar{Y}_i - \bar{Y})(\bar{Y}_i + \bar{Y})$,

$V(t_L) = \sum_{i \in C_p} \frac{W_i}{N_w(t_L)} V_i + \sum_i \frac{W_i}{N_w(t_L)} \left(\bar{Y}_i - \bar{Y}(t_L)\right)\left(\bar{Y}_i + \bar{Y}(t_L)\right)$,

$V(t_R) = \sum_{i \notin C_p} \frac{W_i}{N_w(t_R)} V_i + \sum_i \frac{W_i}{N_w(t_R)} \left(\bar{Y}_i - \bar{Y}(t_R)\right)\left(\bar{Y}_i + \bar{Y}(t_R)\right)$.

Appendix B. Optimal Partitioning

Chou (1991) proposed a K-means like clustering algorithm that uses a generalization of Kullback's information divergence as its distance measure. It has been demonstrated to be very efficient to find the best splitting point for a predictor with a large number of categories.

Let X be a nominal predictor with a category set $A = \{c_1, \dots, c_N\}$. The partitioning problem is to find a binary partition A_0, A_1 of A that minimizes the average impurity,

$I(A_0, A_1|t) = p(t_0|t)i(t_0) + p(t_1|t)i(t_1)$,

where t_0 and t_1 are child nodes determined by the partition, $p(t_0|t) = p(t_0)/p(t)$, and $p(t_1|t) = p(t_1)/p(t)$.

Firstly, we introduce the notion of divergence. This is the key to formulating the partitioning algorithm as an iterative descent.

For a continuous target, the centroid of node t is

$u(t) = E[Y|t] \approx \bar{Y}(t)$.

Let $\hat{y}$ be an approximation of the centroid. Then the divergence of $\hat{y}$ from $u(t)$ is given by

$d(t, \hat{y}) = (u(t) - \hat{y})^2$.

For a categorical target with J categories, the centroid of node t is a J-dimensional class probability vector of $p(j|t)$. Let Ψ be a real nonnegative definite $J \times J$ matrix, where it has the element $\varphi_{ij} = 1 - C(i|j)$. Then the divergence of $\hat{y}$ from $u(t)$ is given by

$d(t, \hat{y}) = (u(t) - \hat{y})'\Psi(u(t) - \hat{y})$.

Let $\alpha: A \to \{0,1\}$ be the function that assigns each category in A to one of the two bins A_0, or A_1, and let $\beta(\cdot)$ be the function on $\{0,1\}$ that assigns a centroid to each bin. That is, for each $c \in A$, let

$$\alpha(c) = \begin{cases} 0 & \text{if } c \in A_0 \\ 1 & \text{if } c \in A_1 \end{cases}'$$

and for $k = 0,1$, let

$\beta(k) = u(t_k)$.

The partitioning algorithm is as follows.

1. Let A_0, A_1 be an initial random partition of A, and compute $\beta(k)$ for $k = 0,1$.
2. Update α to α' for fixed β, by reassigning each c to its nearest neighbor in the divergence sense. That is, let $\alpha'(c) = \text{arg}min_k d\big(c, \beta(k)\big)$, breaking ties arbitrary if $p(c|t) = 0$ or if the divergences are equal.
3. Update β to β' for fixed α', by recomputing the centroid of each bin.
4. Iterative steps 2 and 3 until the average impurity $I(A_0, A_1|t)$ is not reduced, or it reaches the maximal number of iterations (default 20).

References

[1] Jing Xu. Descriptives - ADD - Map Reduce Algorithms for Bivariate Statistics. *IBM SPSS internal design document*. https://w3-connections.ibm.com/files/form/anonymous/api/library/3e2f9545-db12-46b9-8787-59d138d415d9/document/e71b3790-6316-4009-87db-c2c2477de27e/media/Descriptives%20-%20ADD%20-%20Map%20Reduce%20Algorithms%20for%20Bivariate%20Statistics.docx.

[2] H. T. Deng, G. Runger, and E. Tuv (2011). Bias of importance measures for multi-valued attributes and solutions. *Proceedings of the 21st international conference on Artificial neural networks - Volume Part II*. Pages 293-300.

[3] L. Breiman, J. H. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees*. Wadsworth and Brooks.

[4] P. A. Chou (1991). Optimal Partitioning for Classification and Regression Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, No. 4. Pages 340-354.

[5] V. E. Kuz'min, P. G. Polishchuk, A. G. Artemenko, and S. A. Andronati (2011). Interpretation of QSAR Models Based on Random Forests Methods. *Molecular Informatics*, 30(6-7), Pages 593-603.

[6] A. Palczewska, J. Palczewski, R. M. Robinson, and D. Neagu (2013). Interpreting random forest models using a feature contribution method. *IEEE IRI 2013*, San Francisco, California, USA, Pages 112-119.

SNA - Diffusion Analysis Algorithms

1. Introduction

A diffusion process starts with the construction of a call graph and a seed. The call graph is a directed graph in which each node corresponds to a subscriber in the network and the weight on each directed edge reflects the strength of connection between the caller (head of edge) and callee (tail of edge). The weight associated with each edge is based on call data, such as the total number of calls or the total duration of calls over a period of time. The seed is a list of subscribers that are known to have churned during a predefined period of time, typically a subset of the time period that was used to construct the graph (e.g., the same period, the last two weeks, etc.).

Each such churner is assigned with an initial positive energy and all other subscribers are assigned with zero energy. Finally, a diffusion-like process is initiated in the graph, where at each iteration nodes transfer a fraction of their energy to their outgoing neighbors in the graph. The exact value depends linearly on the weight associated with the edge and on a spreading coefficient $d \in (0,1)$, which determines the fraction of energy that can be given away. After the stopping condition is met, each subscriber is associated with a certain amount of energy, where higher values are considered higher probability candidates for churning. A diffusion process like the one described here mimics a word of mouth scenario where the information spreads among people.

The Diffusion Analysis (DA) component implements a certain type of diffusion process. For each node, the DA component computes the amount of energy at the end of the process described above, as well as additional features related to this graph. These features or key performance indicators (KPIs) can be used to build a churn prediction model for the telecommunications industry. It should be noted that diffusion processes such as the one described in this document can be used for additional targets such as customer retention or viral marketing.

In this document we overview the different stages of the diffusion algorithm.

2. Notations

The following notation will be used in each part of the algorithm unless stated otherwise:

W_{adj}	Adjacency weights matrix, which can be the social graph representation as a connectivity matrix. The matrix has $m \times m$ elements
$w_{i,j}$	The (i,j) entry of W_{adj} representing the weight (i.e., connection strength) from node i to node j
T_{adj}	W_{adj} normalized according to lines: $T_{adj}(i,j) = W_{adj}(i,j) / \sum_k w_{ik}$. For each line, $T_{adj}(i,j)$

	represents the proportional strength of the edge (i, j), compared to all other outgoing edges (i, k)
$C_T(n)$	Total energy vector at the end of iteration n. This is a row vector whose dimensions are $1 \times m$
$c_T^k(n)$	k th element of $C_T(n)$, representing the total energy of the k th node at the end of iteration n
$C_F(n)$	Fresh energy vector at the end of iteration n. This is a row vector, whose dimensions are $1 \times m$
$c_F^k(n)$	k th element of $C_F(n)$ - fresh energy of the k th node at the end of iteration n
$C_G(n)$	Given away energy vector at the end of iteration n. This is a row vector, whose dimensions are $1 \times m$
$c_G^k(n)$	k th element of $C_G(n)$ - given away energy of the k th node at the end of iteration n
d	Spreading coefficient. Fraction of fresh energy that is given away in every iteration
m	Number of nodes in the graph
N_i	The i th node
ε	A small constant value. Used for indicating convergence of the process.
θ_{ITER}	Maximal allowed number of iterations

3. Creating the Adjacency Matrix

The adjacency matrix W_{adj} , which corresponds to the calls matrix, is a matrix whose entries represent the strength of connection between two nodes. The matrix will be sparse by definition since each caller only calls a small fraction of the available callees. Technically, the adjacency matrix is created by the loader component. The weights of the matrix are computed according to the settings given to the loader component. While these exact settings are described in the relevant document, we now provide two examples of such computation.

3.1 Counting the Number of Calls as Weight

The simplest option to weight the strength of the connection between caller i and callee j is by counting the number of calls from i to j. In this case,

$w_{i,j} = \#calls(i \to j)$.

Note: The matrix does not have to be symmetric and the weights defined here are integers.

3.2 Counting the Total Duration as Weight

Another option to weight the strength of the connection between caller i and callee j is the summation over the total duration of calls from i to j. In this case,

$w_{i,j} = \sum_{calls(i \to j)} duration\_call(i \to j)$. As in the previous case, the matrix does not have to be symmetric and the weights defined here are also integer (duration of calls is measured in seconds).

4. Description of the Diffusion Algorithm

4.1 Initialization

We initialize the total energy, free energy, and given away energy vectors as follows:

- $C_T(n) = e$ where e_i equals 1 if i is in the list of churners and 0 otherwise:

$$c_T^i(0) = \begin{cases} 1 \; if \;\; i \; is \; a \; churner \\ 0 \; else \end{cases}$$

- $C_F(n) = e$ where e_i equals 1 if i is in the list of churners and 0 otherwise:

$$c_F^i(0) = \begin{cases} 1 \; if \;\; i \; is \; a \; churner \\ 0 \; else \end{cases}$$

- $C_G(n) = 0$, all zeros vector

4.2 Normalization of the Adjacency Matrix

The adjacency matrix is normalized as follows:

$$T_{adj}(i, j) = W_{adj}(i, j) / \sum_k w_{ik}$$

Where W_{adj} is the original adjacency matrix.

4.3 Diffusion Update Equations

The diffusion process is updated according to the following equations:

1. $C_F(n+1) = d \cdot C_F(n) \cdot T_{adj}$. This equation corresponds to the fresh energy received by the node from all its incoming neighbors. The neighbors give away a fraction governed by the spreading coefficient; this fraction is relative to the normalized connection strength.
   A slightly different view of this equation follows. Each node that has outgoing edges, distributes a d fraction of its fresh energy to its outgoing neighbors. Each neighbor gets an amount that is proportional to its relative weight. Therefore, each node obtains the sum over the energies obtained from the incoming neighbors.
2. $C_G(n) = C_F(n) \cdot diag(c_{ii})$ where $diag(c_{ii}) = \begin{pmatrix} c_{11}, 0, 0, ..., 0 \\ 0, c_{22}, 0, ..., 0 \\ ... \\ 0, 0, 0, ..., c_{mm} \end{pmatrix}$ and

   $$c_{ii} = \begin{cases} d \;\; if \;\; out \deg(N_i) > 0 \\ 0 \; else \end{cases}$$

   which can be rewritten as $C_G(n) = d \cdot C_F(n) \cdot \Delta$ where $\Delta = diag(out \deg(N_i) > 0)$ - a diagonal matrix whose diagonal elements are 1 if the corresponding node has at least 1 outgoing edge, and 0 otherwise.

This equation corresponds to the given away energy of each node. This energy is only positive if there is at least 1 outgoing edge. Each node only gives away a d fraction of its fresh energy.

3. $C_T(n+1) = C_T(n) + C_F(n+1) - C_G(n+1)$. This equation represents the update of the total energy. It is the sum of the total energy from the last iteration plus the free received energy, minus the given away energy.

4.4 Simplification of the Update Equations

The third update equation can be rewritten as follows:

$$\begin{aligned} C_T(n+1) &= C_T(n) + C_F(n+1) - C_G(n+1) \\ &= C_T(n) + d \cdot C_F(n) \cdot T_{adj} - d \cdot C_F(n) \cdot \Delta \\ &= C_T(n) + d \cdot C_F(n) \cdot (T_{adj} - \Delta) \end{aligned}$$

Where as before, $\Delta = diag(out\deg(N_i) > 0)$.

This leaves us with only two update equations: one for the total energy and one for the fresh energy:

$$C_F(n+1) = d \cdot C_F(n) \cdot T_{adj}$$

$$C_T(n+1) = C_T(n) + d \cdot C_F(n) \cdot (T_{adj} - \Delta)$$

4.5 Convergence Criterion

The following criterion is used for stopping the diffusion process (indication of convergence):

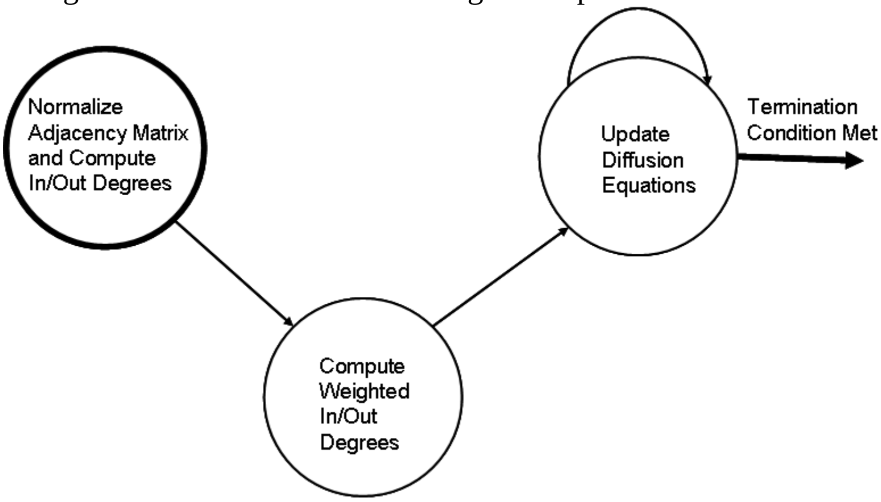
$$\forall i, \left| C_T^i(n+1) - C_T^i(n) \right| < \varepsilon$$

The process stops at the smallest n for which this criterion holds.

5. Implementation Issues

5.1 Algorithm State Machine

The following state machine describes the algorithm presented above.


Normalize Adjacency Matrix and Compute In/Out Degrees
Compute Weighted In/Out Degrees
Update Diffusion Equations
Termination Condition Met

Circles represent states and edges represent the transitions between states. The process begins with the normalizing state (bold circle on the left hand side). During normalization in and out degrees are also calculated. The process continues with the computation of the weighted in and out degrees. The next step of updating the diffusion equations is recurrent and repeats until the termination condition is met.

5.2 Saved Data Structures

As seen in the previous section, the update equations can be rewritten in a manner that allows for the updating and storing of $C_F(n), C_T(n)$ only. The algorithm implementation follows this description.

The following vectors are also stored: in and out degree, weighted in and out degree.

5.3 Termination of the Computation

The natural termination condition of the algorithm is when the convergence criterion is met, as explained in the previous section. However, the diffusion process can be very slow, either due to an incorrect choice of ε or as a result of the problem structure and initial conditions. To allow forced termination, the code implements a hard stop: if the number of update equations has reached a predefined value, the computation stops. Therefore, the stopping condition is as follows:

$$\forall i, \left| C_T^i(n+1) - C_T^i(n) \right| < \varepsilon$$

Or

Number of iterations > θ_{ITER}

The user can control θ_{ITER} (see DA Input Settings.doc).

5.4 Parallelization Scheme

The parallelization schema of the Group Analysis (GA) algorithm is based on the PML architecture [2]. For this section, it is assumed that the reader is familiar with the basic PML event flow model.

In general, each worker is responsible for $\frac{1}{n}$ lines of the adjacency matrix (namely, $\frac{1}{n}$ of the callers). The various stages of the DA algorithm are parallelized as follows:

- Normalization of the adjacency matrix – each worker performs parallelization of $\frac{1}{n}$ lines
- Update equations – each worker updates the values of $\frac{1}{n}$ callers (that correspond to the $\frac{1}{n}$ lines provided to the worker). After the computation, the master collects all the sub vectors and assigns them to the full vectors. The new (updates) ones are then transferred to the workers for the next iteration.
- Convergence check – performed by the master after each iteration.

6. References

[1] K. Dasgupta, R. Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. A. Nanavati, and A. Joshi, “Social ties and their relevance to churn in mobile telecom networks,” in EDBT ’08: Proceedings of the 11th international conference on Extending database technology. ACM, 2008, pp. 668–677

[2] The PML User Guide with CGA and SNA

SNA - Group Analysis Algorithms

1. Introduction

This document describes the group analysis (GA) algorithm of TABI. A more comprehensive description of the scientific ideas behind the algorithm can be found in [1]. The algorithm gets as input records of interaction between pairs of individuals. For the sake of this document we will assume that these are Call Details Records (CDRs). The algorithm then outputs the following objects:

(1) a graph of individuals in which each edge denotes an alleged strong relation between a pair of individuals. This graph is the core of the social network that the algorithm outputs;
(2) a partition of the social network into disjoined reference groups.
(3) A set of basic key performance indices (KPIs) per each group.
(4) A set of basic key performance indices per each individual.

By this the algorithm extracts social relationships, social structures, and social features of groups and individuals. The algorithm is composed of several phases. We will describe the logic behind them as well as the various parameters that governs each one of these steps. Most of the phases of the algorithm are computed in parallel over the Parallel Machine Learning toolbox (PML) [2]. We also briefly describe the parallelization schema as well.

2. Input, Output, and Parameters

In general, the input for TABI is the output of the loader [3] which can be comprised from a single or multiple files. The output is composed of two comma separated files including the basic KPIs of the groups and individuals in the network. Another file containing the kernel relations can be output if the appropriate parameter in the kernel section is turned on (see Section 3.1).

2.1 Running the Algorithm from a Command Line

Running the algorithm from the command line can be done via one the following commands:

Single core, single partition

“Pmlexec” <param-file> <loader-outfile> <model.xml>

Multiple cores

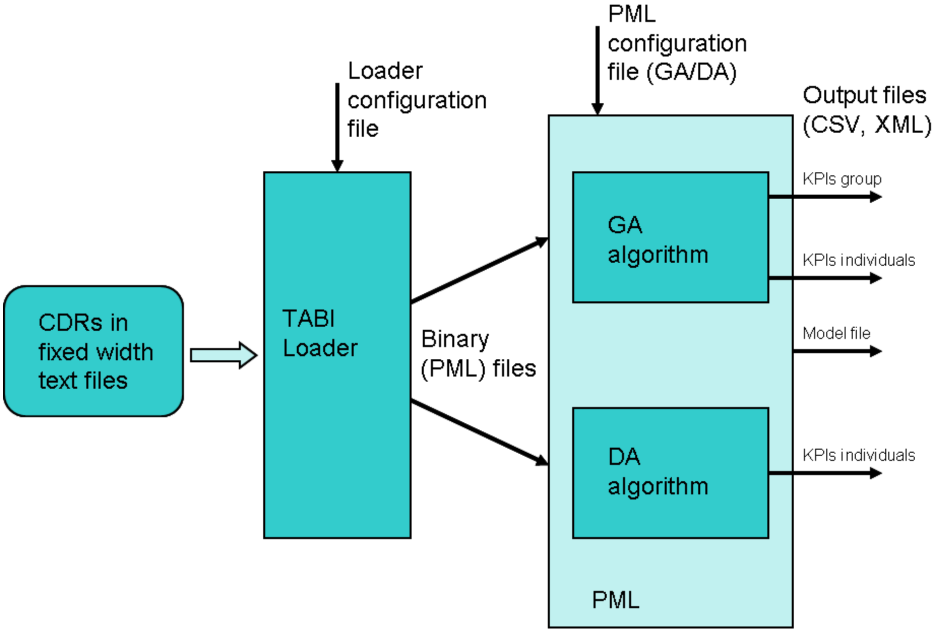
mpiexec -n <ncpus> <pmlexec> <param-file> <loader-outfile> <model.xml>

Multiple partitions

- Suppose the loader output file name is foo.out
  - It will generate output files called foo.out.0 etc. and well as foo.out.0.info etc.
- Copy foo.out.0.info to foo.out.info
- Run the GA using foo.out
  - The multiple training file option in the parameter file should be on

2.2 Basic information flow

The information flow of TABI is depicted in the figure below. The CDRs are collected and presented either in a directory or in a single file of CDRs. They are processed by the TABI loader into either a single binary file in which the most recent calls are kept for each caller, or into a collection of such files, such that at each file about *1/n* fraction of the callers are represented where *n* denotes the number of loader partitions. The loader's output then serves as an input for the GA algorithm that produces the group and individual KPIs. These can then be used by various applications.



2.3 Sample Parameter File

The file below is an example of a complete parameter file for the GA algorithm. The file adheres to an XML format, contains a general section, and an inner section is referring to the kernel parameters.

```
<PMLInput>
        <AlgorithmName>CGA</AlgorithmName>
        <MinClusterSize>1</MinClusterSize>
        <MaxClusterSize>100</MaxClusterSize>
        <OutputFileName>my_out.xml</OutputFileName>
        <NumIterationsConnectingNodes>1</NumIterationsConnectingNodes>
        <VerbosityLevel> 20 </VerbosityLevel>
     <OutputFormat> CSV </OutputFormat>
     <MultipleTrainFiles> True </MultipleTrainFiles>
       <KernelParams>
          <MultipleTrainFiles> True </MultipleTrainFiles>
               <SparsityLevel>0.80</SparsityLevel>
               <KernelType>Friends</KernelType>
               <NormalizingFactor>50000000</NormalizingFactor>
        </KernelParams>
 </PMLInput>
```

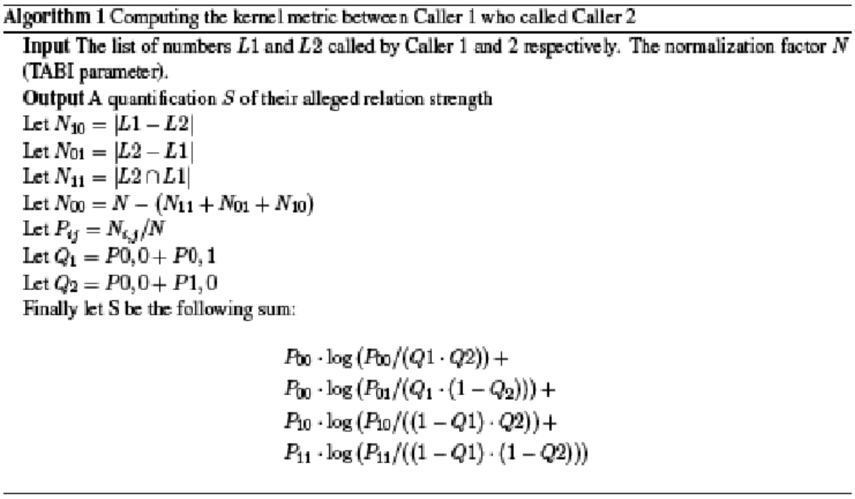
3. The Various Stages of the Algorithm

3.1 Building the Kernel Graph

The goal of this stage is to build a graph of individuals in which an edge denotes an alleged strong relationship between pairs of individuals. In a nutshell, the algorithm defines two individuals as related if (1) They have interacted; and (2) They interacted with similar people. The actual kernel building is done using the following process:

1. Quantifying the relationships between every pair of individuals who have interacted.
2. Constructing the kernel graph from only the strong relations.

In order to quantify the relations, we define the following metric. We assign to each caller ***i***, a vector of its recent called numbers. The length of this vector is governed by a TABI loader [4] parameter called *CyclicTableSize*. For each pair *i* and *j* that interacted at least once, we define a probability space that is based on four events: both called the same person k, both did not call k, *i* called k and *j* did not, and vice versa. The mutual information is then measured on that space. A more elaborate description of this metric is available [1]. In various experiments it was found superior to more direct approaches. The actual computation of the TABI metric is depicted in the figure below. The code can be found in the method: IDMPBKernelFunctionFriends::ComputeKernel(.)



Algorithm 1 Computing the kernel metric between Caller 1 who called Caller 2

```
Input The list of numbers $L1$ and $L2$ called by Caller 1 and 2 respectively. The normalization factor $N$ (TABI parameter).
Output A quantification $S$ of their alleged relation strength
Let $N_{10} = |L1 - L2|$
Let $N_{01} = |L2 - L1|$
Let $N_{11} = |L2 \cap L1|$
Let $N_{00} = N - (N_{11} + N_{01} + N_{10})$
Let $P_{ij} = N_{i,j}/N$
Let $Q_1 = P0,0 + P0,1$
Let $Q_2 = P0,0 + P1,0$
Finally let S be the following sum:
        $P_{00} \cdot \log(P_{00}/(Q1 \cdot Q2)) +$
        $P_{00} \cdot \log(P_{01}/(Q_1 \cdot (1 - Q_2))) +$
        $P_{10} \cdot \log(P_{10}/((1 - Q1) \cdot Q2)) +$
        $P_{11} \cdot \log(P_{11}/((1 - Q1) \cdot (1 - Q2)))$
```

Given the *SparsityLevel* parameter *s*, let *p = 1-s*. Next, the goal is to construct a graph containing as edges only the highest p-fraction of the edges in terms of the quantification above. Due to the distributed nature of the algorithm, each processor (worker) holds its own part of the computed relations and we do not want it to communicate this large amount of data further. Therefore, the graph construction is a twofold process. First, each worker samples its own edges, computes a threshold value *t* such that only *p* fraction of the edge weights are above this threshold (*p*-percentile). These values are then averaged by the master node to compute a common threshold *t**. Next, each worker broadcasts back to the master all the edges in the graph which are above *t**. It should be noted that the resulted kernel graph in undirected.

Kernel Parameters

The parameters below belong to the kernel section of the parameter file.

Num	Parameter name	Description	Data type	Default value	Data range	Restriction
1	MultipleTrainFiles	True if multiple partitions were used by the loader	int	0	0,1	
2	KernelVerbosity	Controls the amount of kernel printouts	int	1	1 - 100	Should be a small number (EndConditionValue << 1)
3	KernelType	Type of the kernel metric	string	Friends		Must be Friends
4	OutputFileName	If on, The algorithm will output the kernel graph to that file, otherwise, the graph will not be output	string	-	-	
5	NumberOfComputeIterations	If greater than 1, the kernel computation will be done in several iterations	int	1	>0	Must be 1
6	NormalizingFactor	Used in the computation of the metric. Recommended to be at the same ballpark as the size of the population	double	50,000,000	> 0	

Due to historical or future reasons, the other parameters are fixed. These include: IsSparseKernel, isSparseData, IndicesData, EmptyMarker, and FilterSimilarities.

3.2 Building the Core Groups

Once the kernel graph is computed, the next step is to partition it into groups. These are called the core groups or initial clusters. The algorithm partitions the kernel graph using a BFS like process (with some high degree preference heuristic) for finding connected components. A parameter called *MaxClusterSize* that governs the maximum size of a cluster. The process stops adding to a group once its size limit has been reached. The underlying assumption is that groups that are too small or too large are not informative for various applications [1].

Relevant Parameters

Num	Parameter name	Description	Data type	Default value	Data range	Restriction
1	MinClusterSize	Miminal cluster size	int	1	>0	
2	MaxClusterSize	Maximal cluster size	int	100	> MinClusterSize	

3.3 Building the Final Groups: Linking Non Core Nodes

In order to increase the coverage of the social network we are building, the next step that TABI takes is to add individuals who are not linked to any core group via the following heuristic: For each such individual we go again over the **call** graph (not the kernel). If a caller *i* called members of core clusters cluster *c* we will add it as a non-core user to the cluster it called the most. This process will be done iteratively if the parameter *NumIterationsConnectingNodes* is greater than 1. In this case, at each phase j the heuristic above will be applied to the groups of iteration j-1. The whole process is done in parallel so each worker responsible only to its own fraction of the callers. Note that not all the callers end up in the social network. A caller who did not communicate with any core group will be left out. There is an essential tradeoff between the strength of the relations controlled by the sparsity threshold, the number of connecting nodes iteration, and the coverage. As a rule of thumb, at least for churn prediction, shooting for coverage of 50% – 75% is often desirable. The heuristics for linking non core nodes is depicted below.

Linking a non-core member

```
For each caller i not already in the core
    Let $S_1$, $S_2$, … $S_l$ be the core groups to whom i called
    If (l > 0)
        Let $k_1$, … ,$k_l$ denote the number of times I called each group
        Let j = argmax $k_j$
        Add caller i to Sj as a non-core group member
```

Relevant Parameters

Num	Parameter name	Description	Data type	Default value	Data range	Restriction
1	NumIterationsConnectingNodes	The number of iterations in which non-cluster nodes will be connected using the above hueristic	int	1	>0	
2	MaxClusterSize	Maximal cluster size	int	100	> MinClusterSize	

3.4 Analyzing Social Influence in the Final Groups

The next stage is to perform a basic analysis on each group in order to extract basic KPIs of groups and individuals. The analysis is done in parallel and on the **call graph**.
The main analysis done at this stage is importance analysis on the call graph projected into each group. This is essentially done via a random walk, once in the direction of the calls to analyze authority leadership and once in the opposite direction to analyze information spreading roles. A more elaborate description can be found in [1].

3.5 Computing the Final KPIs

In the final stage of the algorithm, we go over all the clusters and individuals in the social network and compute the final KPIs for them. Note that TABI outputs only basic social KPIs. These can be enhanced using various techniques. More specifically, individual demographics, usage, and other data can be aggregated (e.g., averaged) at the group level.

This stage is done serially on the master processor. The KPIs are then written into two files, one for the clusters and one for the individual nodes. Note that only those who appear in the final groups will have KPIs.

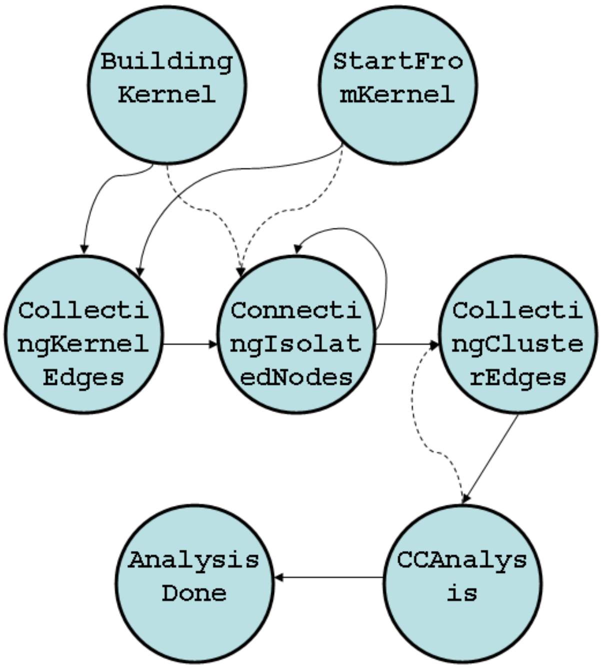
Relevant Parameters

Num	Parameter name	Description	Data type	Default value	Data range	Restriction
1	OutputFileName	The name prefix of the output file. The right suffix will be appended for it	string			
2	OutputFormat	Maximal cluster size	string		XML/CSV/ Normal / Compact	Fixed to CSV

4. Implementation Issues

4.1 Algorithm State Machine

The following state machine describes the algorithm presented above.



Circles represent states and edges represent the transitions between states. Dashed edges represent transitions which are possible under some parameter combinations, but are not common. The actual logic of the algorithm is complicated and should be learned from the code. In the current implementation of TABI, the algorithm always starts from the building kernel state.

4.2 Parallelization Model

The parallelization schema of the GA algorithm is based on the PML architecture [2]. For this section, it is assumed that the reader is familiar with the basic PML event flow model.

The various stages of the GA algorithm are parallelized as follows:

Kernel computation

- Each worker is responsible to 1/n of the callers.
- Threshold computation: Each worker independently computes all the relations of the form S(i,j) such that both i and j belong to its partition
- Kernel metric; Each worker computes all the relations of the form S(i,j) such that I belongs to the worker's partition and I called j. Note that in order to accomplish that the worker must get all the calls matrix

Building the core groups
Done by the master. Not parallelized

Linking non core members
Each worker is responsible to about 1/n of the callers

Group analysis
At the beginning of this stage all workers have the same data and no data is communicated via the PML. The parallelization is obtained via partitioning of the group ids. Each worker *i* analyzes all the groups *k* such that *(k modulo n)* == *i* so about *1/n* fraction of the groups.

General note
The method getWorkerDataRequirements() of the IDMPBCGAData object defines to the PML whether the worker needs to see all the data, just its own partition (1/n of the callers), or no data. A similar method exists in the kernel object.

4.3 A Note on Time and Space Complexity

The most time and space consuming phase in the algorithm is the kernel building. Let *d* denote the cyclic table size parameter of the loader. Roughly speaking, each caller in the loader output graph contributes up to *d* computations of the kernel metric. Each such computation involves going over to lists of length up to *d.* Thus, the overall space complexity is about O(nd). TABI makes a heavy use of STL containers which typically have a logarithmic complexity. Thus, the overall time is about $O(nd^2 \log(nd))$. The actual time is heavily governed by the amount of **page swapping** of the underlying machine. Thus, it is strongly recommended that each core will be able to hold its own partition in the memory. As a rule of thumb, the recommended architectural guidelines are as follows:

- At least one core per each 1M callers
- At least 2GBytes of RAM per each core

These guidelines refer to d = 100.

4.4 The GA Code

The main class that implements the GA algorithm is called *IDMPBCGAData*.
The computation of the kernel is delegated to a class called *IDMPBKernelData* with some specific kernel functions that reside in a file called *idmpb_kernel_functions.hpp*.
The code of these classes can be found in a subdirectory named *src/lib/simple_algorithm*.

5. References

[1] Yossi Richter, Elad Yom-Tov, Noam Slonim, *Predicting customer churn in mobile networks through analysis of social groups",* The Tenth SIAM International Conference on Data Mining, SDM 2010

[2] The PML User Guide with CGA and SNA

[3] TABI Loader User Manual

Spatial Temporal Prediction Algorithms

1. Introduction

Spatio-temporal statistical analysis has many applications. For example, energy management for buildings or facilities, performance analysis and forecasting for service branches, or public transport planning. In these applications, measurements such as energy usage are often taken over space and time. The key questions here are what factors will affect future observations, what can we do to effect a desired change, or to better manage the system. In order to address these questions, we need to develop statistical techniques which can forecast future values at different locations, and can explicitly model adjustable factors to perform what-if analyses.

However, these analytical needs are not the focus of traditional spatio-temporal statistical research. In traditional statistical research, spatio-temporal analysis is treated just as an extension of spatial analysis and focuses more on looking for patterns in past data rather than forecasting future values. The traditional spatio-temporal research targets different application areas such as environmental research. There are, however, different types of spatio-temporal problems in which time is the key component. We therefore need to treat spatio-temporal analysis as a unique type of problem itself, not an extension to spatial analysis. Moreover, we need to explicitly model these factors to allow for what-if analysis. Although these kinds of problems could be addressed by traditional methods, the emphasis is quite different.

This algorithm assumes a fixed set of spatial locations (either point location or center of an area) and equally spaced time stamps common across locations. It can issue predicted or interpolated values at locations with no response measurements (but with available covariates). We call our model spatio-temporal prediction (STP).

The goal of the STP algorithm is to address the needs for solving the spatio-temporal problems. STP can generate predictions at any location within a 3D space for any future time. It also explicitly models the external factors so we can perform what-if analysis.

1.1 Handling of missing data

The algorithm is designed to accommodate missing values in the response variable, as well as in the predictors. We consider an observation at a given time point and location ‘complete’ if all predictors and the response are observed at that time and location. To allow for model fitting in spite of missing data, all of the following conditions must be met:

1. At each location, observations need to be complete for at least one sequence of at least $L + 2$ *consecutive* time points.
2. At each location s_i, for any pair of locations $s_i, s_j,\ s_j \neq s_i$, observations must be complete at *both* locations simultaneously for at least two sequences of $L + 2$ consecutive time points.

3. Overall, at least L sequences of at least $L+2$ consecutive time points must be present in the data (to allow for estimation of α).
4. The total number of complete samples must be at least equal to $D+L+2$, where D is the number of predictors, including the intercept, and L the user-specified lag.
5. After removing locations according to the rules above, no more than 5% of the remaining records should be incomplete. As an example, if after removing locations, n locations and m time stamps remain, no more than $n \times m \times .05$ records should be incomplete.

The above conditions should be verified in the following order:

Step 1. Remove locations that do not meet condition 1.

Step 2. Remove locations that violate condition 2 in the following order:

(a) Let $\mathcal{J}$ be the set of points that violate condition 2.

(b) Eliminate from the data set the observation(s) that violate condition 2 for the greatest number of pairs. In case of a tie, remove all observations that are tied.

(c) Update $\mathcal{J}$ by removing any observations that now no longer violate Condition 2. That is, remove observation that only violated the condition 2 in a pair with the observations that were removed in Step 2b.

(d) Iterate steps 2b and 2c until $\mathcal{J}$ is empty.

Step 3. If after Steps 1 and 2, conditions 3-5 are violated, the model cannot be fit.

2 Model

2.1 Notation

The following notation is used for the model inputs:

Name	Symbol	Type	Dimensions
Number of time stamps	$m > L$	integer	1
Number of measurement locations	$n \geq 3$	integer	1
Number of prediction grid points	N	integer	1
Number of predictors (including intercept)	D	integer	1
Index of time stamps	$t \in \{1, \dots, m\}$	integer	1
Spatial coordinates	$s \in \{s_1, \dots, s_n\}; s_j = (u_j, v_j, w_j)'$	vector	3×1
Targets observed at location s and time t	$Y_t(s)$	scalar	1
Targets observed at location s	$Y(s)$	vector	$m \times 1$
Targets observed at time t	Y_t	vector	$n \times 1$
Predictors observed at location s and time t	$X_t(s) = (X_{t,1}(s), \dots, X_{t,D}(s))'$	vector	$D \times 1$
Predictors observed at location s	$X(s) = (X_1(s), \dots, X_m(s))'$	matrix	$m \times D$
Predictors observed at time t	$X_t = (X_t(s_1), \dots, X_t(s_n))'$	matrix	$n \times D$
Maximum autoregressive time lag	$L > 0$	integer	1
Length of prediction steps	$H > 0$	integer	1

Notes

i. For a predictor that does not vary over space, $X_{t,d}(s_1) = X_{t,d}(s_2) = \cdots = X_{t,d}(s_n)$;

ii. For a predictor that does not evolve over time, $X_{1,d}(s) = X_{2,d}(s) = \cdots = X_{m,d}(s)$.

The following notation is used for model definition and computation:

Name	Symbol	Type	Dimension
Coefficient vector for linear model	$\boldsymbol{\beta} = (\beta_1, \dots, \beta_D)$	vector	D
Coefficient vector for AR model	$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_L)$	vector	L
Vector of 1's	$1 = (1, \dots, 1)'$	vector	variable
Kronecker product	$\otimes$	operator	NA

2.1 Model structure

$$Y_t(s) = \sum_{d=1}^{D} \beta_d \, X_{t,d}(s) + Z_t(s) \tag{1}$$

where $Z_t(s)$ is mean-zero space-time correlated random process. Users can specify whether an "intercept" term needs to be included in the model. The inference algorithm works with general "continuous" variables, and with or without intercept.

- Autoregressive model, AR(L) for time autocorrelation (Brockwell and Davis, 2002):

$$Z_t(s) = \sum_{l=1}^{L} \alpha_l \, Z_{t-l}(s) + \epsilon_t(s) \tag{2}$$

Note that users need to specify the maximum AR lag L.

Let $\epsilon_t = (\epsilon_t(s_1), \dots, \epsilon_t(s_n))'$ be the AR residual vector at time t. Since the time autocorrelation effect has already been removed, $\epsilon_{L+1}, \dots, \epsilon_m$ are independent.

- Parametric or nonparametric covariance model for spatial dependence:

$$V(\epsilon_t) = \Sigma_S, t = L + 1, \dots, m \tag{3}$$

where $\Sigma_S = \{R(s_i, s_j)\}_{i,j=1,\dots,n}$ is a $n \times n$ covariance matrix of spatial covariance functions $R(s, s') = Cov(Y_t(s), Y_t(s'))$ at observed locations. Two alternative ways of modeling the spatial covariance function $R(s_i, s_j)$ are implemented - a *variogram-based parametric* model (Cressie, 1993) and a *Empirical Orthogonal Functions (EOF)-based nonparametric* model (Cohen and Johnes, 1969; Creutin and Obled, 1982).

Note that users can specify which covariance model to be used.

- If a "parametric model" is chosen, the algorithm will automatically test for the goodness-of-fit. If the test suggests a parametric model is not adequate, the algorithm switch to EOF model fitting and issue prediction based on EOF model.
- If a EOF model is chosen, the switching test part will be skipped, and both model fitting and prediction will follow EOF-based algorithm.

Under this model decomposition, the covariance structure for the spatio-temporal process $Y = (Y'_{L+1}, \dots, Y'_m)'$ is of separable form

$$V(Y) = V(Z) = \Sigma = \Sigma_T \otimes \Sigma_S \tag{4}$$

where $\Sigma_T = \{\gamma_T(t - t')\}_{t=L+1,\dots,m;t'=L+1,\dots,m}$ is the $(m - L) \times (m - L)$ AR(L) covariance matrix with the autocovariance function.

3 Estimation algorithm

This section provides details on the multi-step procedure to fit the STP model (see Figure 1) when the user specifies a "parametric model". If an "empirical model" is specified, the switching test part will be skipped, and both model fitting and prediction follows EOF-based algorithm.

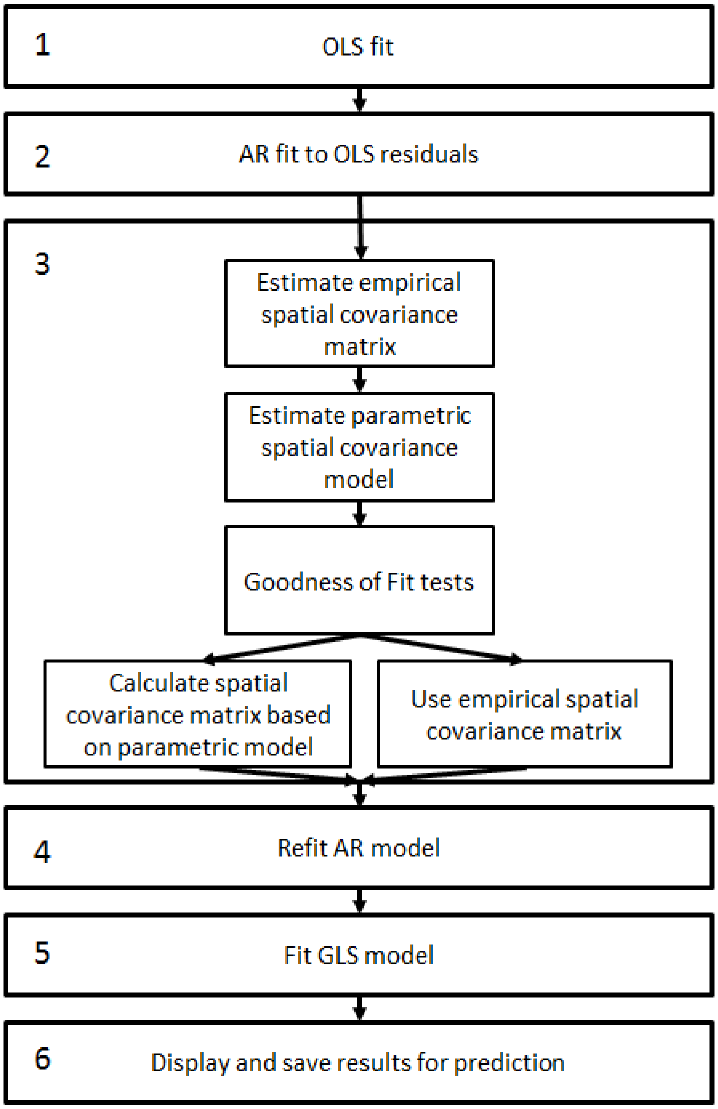

1
OLS fit
2
AR fit to OLS residuals
3
Estimate empirical spatial covariance matrix
Estimate parametric spatial covariance model
Goodness of Fit tests
Calculate spatial covariance matrix based on parametric model
Use empirical spatial covariance matrix
4
Refit AR model
5
Fit GLS model
6
Display and save results for prediction

Figure 1. Flowchart of algorithm steps for model fitting when a "parametric model" is specified.

Step 1: Fit regression model by ordinary least squares (OLS) regression using only observations that have no missing values (see Section 3.1).

We first ignore the spatio-temporal dependence in the data and simply estimate the fixed regression part by OLS and obtain the regression residuals $Z_t(s)$.

Step 2: Fit autoregressive model using only data without missing values (see Section 3.2).

Ignoring spatial dependence in OLS residuals $Z_t(s)$, we estimate autoregressive coefficients by fitting the regression model (2) and obtain the AR residuals $\epsilon_t(s)$.

Step 3: Fit spatial covariance model and test for goodness of fit on data without missing values (see Section 3.3).

We fit a parametric spatial covariance model. We perform two Goodness of Fit tests to decide whether to continue with the parametric covariance model or the empirical covariance matrix.

Step 4: Refit autoregressive model using augmented data (see Section 3.4).

We refit autoregressive model accounting for spatial dependence by generalized least squares (GLS) and obtain improved AR coefficients α.

Step 5: Refit Regression model using augmented data (see Section 3.5).

We obtain improved regression coefficients β by GLS to account for spatio-temporal correlation in the data.

Step 6: Save the results for use in output and prediction.

3.1 Fit regression model

We first ignore the spatio-temporal dependence in the data and simply estimate the fixed regression part by OLS. Assume that out of nm location-time combinations, q samples have missing values in either X or Y. Let $Y = (Y_1', \dots, Y_m')'$, a $(nm - q) \times 1$-vector and $X = (X_1', \dots, X_m')'$, a $(mn - q) \times D$ matrix, such that X and Y contain only complete observations, i.e., observations without any missing values. The OLS estimates of the regression coefficients are:

$$\widehat{\boldsymbol{\beta}} = (X'X)^{-1}X'Y \tag{5}$$

The residuals are:

$$\hat{Z} = Y - X\widehat{\boldsymbol{\beta}}. \tag{6}$$

3.2 Fit autoregressive model

We estimate autoregressive coefficients by OLS assuming no spatial correlation and AR(L) as model for time-series autocorrelation,

$$\hat{Z}_t = \alpha_1 \hat{Z}_{t-1} + \cdots + \alpha_L \hat{Z}_{t-L} + \boldsymbol{\epsilon}_t, \tag{7}$$

where $\hat{Z}_t$ is a $n_t \times 1$ vector. Note that due to the existence of missing values, the number of locations n_t varies among different time points. Moreover, for each time points t, only locations with no missing values at $L + 1$ consecutive time points, i.e., $(t, t-1, \dots, t-L)$ can be used for model fitting, therefore, $\sum_{t=L+1}^{m} n_t \leq [n(m-L) - q]$.

Step 1: Construct $n_t \times L$ time lag matrix

$$\hat{Z}_{t-lag} = \left(\hat{Z}_{t-1}, \hat{Z}_{t-2}, \dots, \hat{Z}_{t-L}\right), t = L+1, \dots, m \tag{8}$$

Step 2: Let $\hat{Z}_{lag} = \left(\hat{Z}'_{L+1-lag}, \dots, \hat{Z}'_{m-lag}\right)'$ and $\hat{Z}^* = \left(\hat{Z}'_{L+1}, \dots, \hat{Z}'_m\right)'$. Solve the linear system

$$\left(\hat{Z}'_{lag}\hat{Z}_{lag}\right)\boldsymbol{\alpha} = \hat{Z}'_{lag}\hat{Z}^* \tag{9}$$

which is equivalent to solving

$$\left(\sum_{t=L+1}^{m} \hat{Z}'_{t-lag}\hat{Z}_{t-lag}\right)\boldsymbol{\alpha} = \sum_{t=L+1}^{m} \hat{Z}'_{t-lag}\hat{Z}_{t} \qquad (10)$$

using the sweep operation to find estimate $\hat{\alpha}$.

Step 3: Compute the de-autocorrelated AR(L) residuals

$$\hat{\epsilon}_t = \widehat{Z}_t - \widehat{\alpha}_1\widehat{Z}_{t-1} - \cdots - \widehat{\alpha}_L\widehat{Z}_{t-L}, t = L+1, \dots, m \qquad (11)$$

3.3 Fit model and check goodness of fit for spatial covariance structure

We explicitly model the spatial covariance structure among locations, rather than using variogram estimation.

Under the assumption of the model (stationarity, AR-relationship removed), the mean of the residuals is 0 at all locations. We therefore estimate the unadjusted empirical covariances s_{ij} and correlations r_{ij} assuming mean 0, i.e.,

$$\boldsymbol{S} = [s_{ij}]_{i,j=1,\dots,n}, s_{ij} = \frac{1}{t_{ij}}\sum_{t} \hat{\epsilon}_t(s_i)\hat{\epsilon}_t(s_j) \qquad (12)$$

where t_{ij} is the number of complete residual pairs between locations s_i and s_j, and t indexes these pairs, i.e., the time points for which both $\hat{\epsilon}_t(s_i)$ and $\hat{\epsilon}_t(j)$ are non-missing.

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}} \qquad (13)$$

To determine whether to model the spatial covariance structure parametrically or to use the nonparametric EOF model, we perform the following two tests sequentially:

1. Fit parametric model to covariances using the parameter vector $\boldsymbol{\psi} = (\sigma^2, \theta, \tau^2)$ (Cressie 1993)

$$Cov(\epsilon_t(s_i), \epsilon_t(s_j); \hat{\psi}) = \begin{cases} \hat{\sigma}^2 exp\left(-\left(h_{ij}/\hat{\theta}\right)^p\right), & if\ h_{ij} > 0; \\ \hat{\sigma}^2 + \hat{\tau}^2, & otherwise. \end{cases} \qquad (14)$$

   where $h_{ij} = \left\|s_i - s_j\right\|_2$ is the Euclidean distance between locations s_i and s_j. Users need to specify the values for the order parameter p.

   $p \in [1, 2]$ is a user-defined parameter that determines the class of covariance models to be fit. $p = 1$ corresponds to an exponential covariance model, $p = 2$ results in a Gaussian covariance model and $p \in (1, 2)$ belongs to the powered exponential family.

   Next, determine if there is a significant decay over space by testing $H_0: -1/\theta^p \geq 0$. If we fail to reject H_0, we conclude that the decay over space is not significant, and EOF estimation will be used. If EOF estimation is used, there is not need to calculate θ, σ or τ, as we have concluded that they are invalid descriptions of the covariance matrix. In fact, there may not be valid solutions for these parameters, therefore they should not be estimated.

2. If the previous test rejects H_0, test for homogeneity of variances among locations: if homogeneity of variances is rejected, EOF estimation will be used. Otherwise, the parametric covariance model will be used.

3.3.1 Fit and test parametric model

a) Enforce a minimum correlation of +.01: if $r_{ij} < .01$, set $s_{ij} = .01\sqrt{s_{ii}s_{jj}}$ and $r_{ij} = .01$.

b) Let $\boldsymbol{s}$ be the vectorized lower triangular of the covariance matrix (excluding the diagonal, i.e., excluding variances), $\boldsymbol{r}$ be the vectorized lower triangular of the correlation matrix (excl. diagonal), and $\boldsymbol{h}$ the corresponding vector of pairwise distances between the n locations. $\boldsymbol{s}, \boldsymbol{r}$ and $\boldsymbol{h}$ are each vectors of length $n(n-1)/2$.

Define $\varphi = -1/\theta^p$. Fit the linear model $\ln \boldsymbol{s} = \ln \sigma^2 + \varphi \boldsymbol{h}^p$ using a GLS fit:

$$\boldsymbol{A} = [1, \boldsymbol{h}^p] \tag{15}$$

$$\boldsymbol{V}^{-1} = \frac{1}{2}\boldsymbol{T}(\boldsymbol{B}^{-1} - cbb')\boldsymbol{T} \tag{16}$$

where$\boldsymbol{b} = 2\boldsymbol{r}^2/(1-\boldsymbol{r}^2)$, $\boldsymbol{r}^2$ is obtained by squaring each element of vector $\boldsymbol{r}$, $\boldsymbol{B}^{-1} =$ diag$(\boldsymbol{b})$, and scalar $c = 1/(1 + \boldsymbol{1}'\boldsymbol{B}^{-1}\boldsymbol{1})$. Also, let $\boldsymbol{T}$ = diag$\left[\sqrt{t_k}\right], k = 1, \dots, n(n-1)/2$, where t_k is the number of pairs of de-autocorrelated residuals in the calculation of the corresponding element r_k in $\boldsymbol{r}$, i.e., the number of observations pairs that went into calculating r_k, which may be different for each entry of the covariance matrix, depending on missing values. Note that t_k corresponds to the vectorized lower triangular of $\left[t_{ij}\right]_{i,j=1,\dots,n}$, where t_{ij} are as defined in (12).

Let $\boldsymbol{\eta} = (\ln \sigma^2, \varphi)$, the GLS estimator can be calculated as

$$\widehat{\boldsymbol{\eta}} = \left(\boldsymbol{A}'\boldsymbol{V}^{-\boldsymbol{1}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}'\boldsymbol{V}^{-1}\ln \boldsymbol{s}$$

The standard error for $\widehat{\boldsymbol{\eta}}$ will be $se(\widehat{\boldsymbol{\eta}}) = \sqrt{\text{diag}[(\boldsymbol{A}'\boldsymbol{V}^{-\boldsymbol{1}}\boldsymbol{A})^{-1}]}$.

Calculate the test statistic $z_1 = \frac{\widehat{\varphi}}{se(\widehat{\varphi})}$. If $z_1 \geq z_{.05}$, where $z_{.05}$ is the .05 quantile of the standard normal distribution (or critical value for selected level of significance γ_1), then all following calculations will be performed using the empirical spatial covariance matrix, i.e., $\Sigma_{\boldsymbol{S}} = \boldsymbol{S}$, and the nonparametric EOF model will be used for prediction. Equivalently, a p-value p_1 can be calculated by evaluating the standard Normal cumulative distribution function (CDF) at z_1 (i.e., $p_1 = P(Z < z_1)$). If $p_1 \geq$ level of significance γ_1, then all following calculations will be performed using the empirical covariance matrix.

c) If the previous test does reject H_0 (i.e., we have not yet decided to continue with the empirical covariance matrix), continue to perform the following test: Let $v = (s_{11}, s_{22}, \dots, s_{nn})'$ be the $(n \times 1)$-vector of location-specific variances. Calculate the weighted mean variance $\bar{v}$

$$\bar{v} = 1'\boldsymbol{W}^{-1}v/(1'\boldsymbol{W}^{-1}1) = 1'\boldsymbol{W}^{-1}v\Big/\sum_{i,j} w_{ij}^* \tag{17}$$

where $\boldsymbol{W} = [w_{ij}] = [s_{ij}^2/t_{ij}]_{i,j=1,\dots,n}$ is an $n \times n$ matrix, where t_{ij} is defined as in (12), and $\boldsymbol{W}^{-1} = [w_{ij}^*]_{i,j=1,\dots,n}$.

Calculate the test statistic $z_2 = (\boldsymbol{v} - \overline{v})'\boldsymbol{W}^{-1}(\boldsymbol{v} - \overline{v})$. If $z_2 \geq \chi^2_{n-1,.95}$ (or critical value for $[1 -$ selected level of significance $\gamma_2])$, all following calculations will be performed using the empirical spatial covariance matrix, i.e., $\Sigma_{\boldsymbol{S}} = \boldsymbol{S}$, and the nonparametric EOF model will be used for prediction. Equivalently, one may compute a p-value p_2 by evaluating 1 minus the $\chi^2_{n-1} - \mathrm{CDF}$: $p_2 = P(\chi^2_{n-1} > z_2)$. If $p_2 <$ level of significance γ_2, then all following calculations will be performed using the empirical spatial covariance matrix.

d) If the two tests in b) and c) do not indicate a switch to the EOF model, all following calculations will be performed using the parametric covariance model, i.e., the spatial covariance matrix Σ_S is constructed according to (14). Recall that $\eta = (ln\,\sigma^2, -1/\theta^p)$. The missing parameter τ^2 is derived as $\widehat{\tau^2} = max\left\{0, \frac{1}{n}\sum_{i=1,\dots,n} s_{ii} - exp\left[\widehat{ln\,\sigma^2}\right]\right\}$.

3.4 Re-fit autoregressive model

We refit the autoregressive model accounting for spatial dependence using GLS with augmented data:

Step 1: Compute the Cholesky factorization $\boldsymbol{\Sigma}_S = \boldsymbol{H}_S\boldsymbol{H}_S'$ and the inverse matrix $\boldsymbol{H}_{\boldsymbol{S}}'$.

Step 2: Substitute 0 for missing values such that $\widehat{\boldsymbol{Z}}_{t-lag,impute}$ is an $n \times L$ matrix and $\widehat{\boldsymbol{Z}}_{t,impute}$ is a vector of length n.

Step 3: Augment predictor matrix as follows. Let $\widehat{\boldsymbol{Z}}_{lag,impute} = \left(\widehat{\boldsymbol{Z}}'_{L+1-lag,impute}, \dots, \widehat{\boldsymbol{Z}}'_{m-lag,impute}\right)'$ be a $n(m-L) \times L$ matrix and $\widehat{\boldsymbol{Z}}_{impute} = \left(\widehat{\boldsymbol{Z}}'_{L+1,impute}, \dots, \widehat{\boldsymbol{Z}}'_{m,impute}\right)'$ is a vector of length $n(m-L)$, then

$$\widehat{\boldsymbol{Z}}_{lag,aug} = (\widehat{\boldsymbol{Z}}_{lag,impute}, \dots, \mathbf{I}_{Zmiss})$$

where $\mathbf{I}_{Zmiss}$ is a $n(m-L) \times q_Z$ indicator matrix given q_Z the total number of rows with missing values in either $\widehat{\boldsymbol{Z}}^*$ or $\widehat{\boldsymbol{Z}}_{lag}$. If there is a missing value in the *i*th row of either $\widehat{\boldsymbol{Z}}^*$ or $\widehat{\boldsymbol{Z}}_{lag}$, and if this is the *j*th out of all q_Z rows that have missing values, then the *j*th column of $\mathbf{I}_{Zmiss}$ is all 0 except for the *i*th element, which is set to 1.

Step 4: Remove the spatial correlation: $\widetilde{\boldsymbol{Z}}_{t-lag,aug} = \boldsymbol{H}_S^{-1}\widehat{\boldsymbol{Z}}_{t-lag,aug}$ and $\widetilde{\boldsymbol{Z}}_{t,impute} = \boldsymbol{H}_S^{-1}\widehat{\boldsymbol{Z}}_{t,impute}$, where $\widehat{\boldsymbol{Z}}_{t-lag,aug}$ are the submatrices of $\widehat{\boldsymbol{Z}}_{lag,aug}$ that correspond to the rows of the matrices $\widehat{\boldsymbol{Z}}_{t-lag,impute}$.

Step 5: Use the same computational steps as for the linear system in equation (10) to solve the linear system

$$\left(\sum_{t=L+1}^{m} \widetilde{\boldsymbol{Z}}'_{t-lag,aug}\widetilde{\boldsymbol{Z}}_{t-lag,aug}\right)\boldsymbol{\alpha}_{aug} = \sum_{t=L+1}^{m} \widetilde{\boldsymbol{Z}}'_{t-lag,aug}\widetilde{\boldsymbol{Z}}_{t,impute} \qquad (18)$$

where $\boldsymbol{\alpha}_{aug}$ is a vector of length $L + q_Z$, and there are $L^* + q_Z^*$ non-redundant parameters in above linear system. The AR coefficient estimate $\widehat{\boldsymbol{\alpha}}$ is the subvector consisting of the first L elements of $\widehat{\boldsymbol{\alpha}}_{aug}$, there are L^* non-redundant parameters in first D elements of $\widehat{\boldsymbol{\alpha}}_{aug}$, and q_Z^* non-redundant parameters in last q_Z elements of $\widehat{\boldsymbol{\alpha}}_{aug}$.

3.5 Re-fit Regression model

Refit regression model by GLS using augmented data to account for spatio-temporal correlation in the data.

Step 1: Substitute the following for missing values such that $\boldsymbol{X}_{impute}$ is a $nm \times D$ matrix and $\boldsymbol{Y}_{impute}$ is a vector of length nm: at location s_i, use the mean of $\boldsymbol{Y}(s_i)$ and the mean of each predictor in $\boldsymbol{X}(s_i)$.

Step 2: Augment predictor matrix as follows.

$$\boldsymbol{X}_{aug} = \left(\boldsymbol{X}_{impute}, \boldsymbol{I}_{Xmiss}\right)$$

where $\mathbf{I}_{Xmiss}$ is a $nm \times q$ indicator matrix given q the total number of rows with missing values in either $\boldsymbol{X}$ or $\boldsymbol{Y}$. If there is a missing value in *i*th row of either $\boldsymbol{X}$ or $\boldsymbol{Y}$, and if this is the *j*th out of all q rows that have missing value, then the *j*th column of $\mathbf{I}_{Xmiss}$ is all 0 except for the *i*th element, which is 1.

Step 3: Remove the spatial correlation: $\widetilde{\boldsymbol{X}}_{t,aug} = \boldsymbol{H}_S^{-1}\boldsymbol{X}_{t,aug}$ and $\widetilde{\boldsymbol{Y}}_{t,impute} = \boldsymbol{H}_S^{-1}\boldsymbol{Y}_{t,impute}$.

Step 4: Remove the autocorrelation:

$$\breve{\boldsymbol{X}}_{t,aug} = \widetilde{\boldsymbol{X}}_{t,aug} - \widehat{\alpha}_1\widetilde{\boldsymbol{X}}_{t-1,aug} - \cdots - \widehat{\alpha}_L\widetilde{\boldsymbol{X}}_{t-L,aug}, t = L+1, \dots, m \qquad (19)$$

$$\breve{\boldsymbol{Y}}_{t,impute} = \widetilde{\boldsymbol{Y}}_{t,impute} - \widehat{\alpha}_1\widetilde{\boldsymbol{Y}}_{t-1,impute} - \cdots - \widehat{\alpha}_L\widetilde{\boldsymbol{Y}}_{t-L,impute}, t = L+1, \dots, m \qquad (20)$$

Step 5: Solve the linear system

$$\left(\breve{\boldsymbol{X}}'_{aug}\breve{\boldsymbol{X}}_{aug}\right)\boldsymbol{\beta}_{aug} = \breve{\boldsymbol{X}}'_{aug}\breve{\boldsymbol{Y}}_{impute} \qquad (21)$$

where $\breve{\boldsymbol{Y}}_{impute} = \left(\breve{\boldsymbol{Y}}'_{L+1,impute}, \dots, \breve{\boldsymbol{Y}}'_{m,impute}\right)'$, an $n(m-L) \times 1$-vector and $\breve{\boldsymbol{X}}_{aug} = \left(\breve{\boldsymbol{X}}'_{L+1,aug}, \dots, \breve{\boldsymbol{X}}'_{m,aug}\right)'$, a $n(m-L) \times (D+q)$ matrix, $\boldsymbol{\beta}_{aug}$ is a vector of length $D+q$, and there are $D^* + q^*$ non-redundant parameters in above linear system. The regression coefficients estimate $\widehat{\boldsymbol{\beta}}$ is the subvector consisting of first D elements of $\widehat{\boldsymbol{\beta}}_{aug}$, there are D^* non-redundant parameters in first D elements of $\widehat{\boldsymbol{\beta}}_{aug}$, and q^* non-redundant parameters in last q elements of $\widehat{\boldsymbol{\beta}}_{aug}$.

3.6 Statistics to display

3.6.1 Goodness of Fit statistics

We present statistics referring to the three main elements of the model: the mean structure, the spatial covariance structure, and the temporal structure.

1. **Goodness of fit mean structure model $\boldsymbol{X\beta}$:**

Let Q be the set of observations $(Y_t(s), \boldsymbol{X}_t(s))$ that have missing values in either $Y_t(s)$ or $\boldsymbol{X}_t(s)$. Note that q has been defined as the number of observations in Q.

Calculate the mean squared error (MSE) and an R^2 statistic based only on complete observations:

$$\text{MSE} = \sum_{\substack{s \in \{s_1,\ldots,s_n\}; \\ t=1,\ldots,m; \\ Y_t(\boldsymbol{S}) \notin Q}} \left(Y_t(s) - \hat{Y}_t(s)\right)^2 / (nm - q - D^*) \qquad (22)$$

$$R^2 = \begin{cases} 1 - \sum\limits_{\substack{s \in \{s_1,\ldots,s_n\}; \\ t=1,\ldots,m; \\ Y_t(\boldsymbol{S}) \notin Q}} \left(Y_t(s) - \hat{Y}_t(s)\right)^2 \Bigg/ \sum\limits_{\substack{s \in \{s_1,\ldots,s_n\}; \\ t=1,\ldots,m; \\ Y_t(\boldsymbol{S}) \notin Q}} Y_t(s)^2\,, & \text{if there is no intercept} \\ 1 - \sum\limits_{\substack{s \in \{s_1,\ldots,s_n\}; \\ t=1,\ldots,m; \\ Y_t(\boldsymbol{S}) \notin Q}} \left(Y_t(s) - \hat{Y}_t(s)\right)^2 \Bigg/ \sum\limits_{\substack{s \in \{s_1,\ldots,s_n\}; \\ t=1,\ldots,m; \\ Y_t(\boldsymbol{S}) \notin Q}} (Y_t(s) - \bar{Y}_t(s))^2\,, & \text{if there is an intercept} \end{cases} \qquad (23)$$

where $\hat{Y}_t(s) = \boldsymbol{X}'_t(s)\boldsymbol{\beta}$, D^* is the number of non-redundant parameters of re-fitted regression in first D elements of $\widehat{\boldsymbol{\beta}}_{aug}$, and $\bar{Y}_t(s)$ is the mean of Y only on complete observations. Note that for this calculation the original (untransformed) observations $\boldsymbol{Y}$ and covariates $\boldsymbol{X}$ are used. Alternatively, we can calculate the adjusted R^2

$$R^2_{adj} = 1 - \frac{nm - q}{nm - q - D^*}(1 - R^2) \qquad (24)$$

2. **Goodness of fit for AR model:**

   Present *t*-tests for AR parameters based on variance estimates in item 3 in Section 3.6.2.

3. **Goodness of fit of spatial covariance model:**

   Present the test statistics listed in item 5 in Section 3.6.2.

3.6.2 Model and parameter estimates

The following information should be displayed as a summary of the model:

1. Model coefficients $\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\alpha}}$ obtained in Sections 3.4 and 3.5
2. Standard errors of elements of $\boldsymbol{\beta}$ based on $V(\widehat{\boldsymbol{\beta}})$, the covariance matrix of $\widehat{\boldsymbol{\beta}}$, which is the upper $D \times D$ submatrix of $V(\widehat{\boldsymbol{\beta}}_{aug})$:

$$V\left(\widehat{\boldsymbol{\beta}}_{aug}\right) = \frac{SS_e}{df_e} \times \left(\check{\boldsymbol{X}}'_{aug}\check{\boldsymbol{X}}_{aug}\right)^{-1} = \frac{SS_e}{df_e} \times \left(\sum_{t=L+1}^{m} \check{\boldsymbol{X}}'_{t,aug}\check{\boldsymbol{X}}_{t,aug}\right)^{-1} \tag{25}$$

where

- $SS_e = \sum_{i=1}^{N}\left(\check{\boldsymbol{Y}}_{impute} - \left(\check{\boldsymbol{Y}}_{impute}\right)^{*}\right)^{2} = \tilde{r}_{\check{Y}\check{Y}}(n(m-L)-1)V\left(\check{\boldsymbol{Y}}_{impute}\right)$,
  - $\left(\check{\boldsymbol{Y}}_{impute}\right)^{*}$ is the predicted value based on estimated $\widehat{\boldsymbol{\beta}}$,
  - $\tilde{r}_{\check{Y}\check{Y}}$ is corresponding element of $\check{\boldsymbol{Y}}_{impute}$ in the correlation matrix of re-fitted regression after sweep operation,
  - $n(m-L)$ is number of transformed records used in equation (21) for re-fit regression ,
  - and $V\left(\check{\boldsymbol{Y}}_{impute}\right)$ is variance of $\check{\boldsymbol{Y}}_{impute}$.
- $df_e = n(m-L) - p$, and $p = D^{*} + q^{*}$ is the number of non-redundant parameters in re-fitted regression.

Based on these standard errors, t-test statistics and/or p-values may be computed and displayed according to standard definitions and output scheme of linear models (please refer to linear model documentation):

(a) For each element β_j of $\widehat{\boldsymbol{\beta}}$ and the corresponding j-th diagonal element of $V\left(\widehat{\boldsymbol{\beta}}\right)$, $j = 1, \dots, D$, compute the t-statistic $t_j = \beta_j / \sqrt{V\left(\widehat{\boldsymbol{\beta}}\right)_{jj}}$

(b) The p-value corresponding to t_j is $2 \times$ the value of the cumulative distribution function of a t-distribution with $nm - q - D^{*}$ degrees of freedom, i.e., $p_j = 2 \cdot \left(1 - P\left(t_{nm-q-D^{*}} \le |t_j|\right)\right)$.

Note that depending on the implementation of the GLS estimation in Section 3.5, $\left(\check{\boldsymbol{X}}'_{aug}\check{\boldsymbol{X}}_{aug}\right)^{-1}$ may have already been computed, in which case this expression does not need to be recalculated.

3. Standard errors of $\boldsymbol{\alpha}$ based on $V(\widehat{\boldsymbol{\alpha}})$, the covariance matrix of $\widehat{\boldsymbol{\alpha}}$, which is the upper $L \times L$ submatrix of $V\left(\widehat{\boldsymbol{\alpha}}_{aug}\right)$:

$$V\left(\widehat{\boldsymbol{\alpha}}_{aug}\right) = \frac{SS_e^{*}}{df_e^{*}} \times \left(\sum_{t=L+1}^{m} \widetilde{\boldsymbol{Z}}'_{t-lag,aug}\widetilde{\boldsymbol{Z}}_{t-lag,aug}\right)^{-1} \tag{26}$$

where

- $SS_e^{*} = \sum_{i=1}^{N}\left(\tilde{Z}_{t,impute} - \left(\tilde{Z}_{t,impute}\right)^{*}\right)^{2} = \tilde{r}_{\tilde{Z}\tilde{Z}}(n(m-L)-1)V\left(\tilde{Z}_{t,impute}\right)$,
  - $\left(\tilde{Z}_{t,impute}\right)^{*}$ is the predicted value based on estimated $\hat{\alpha}$ and $\tilde{Z}_{t-lag,aug}$
  - $\tilde{r}_{\tilde{Z}\tilde{Z}}$is corresponding element of $\tilde{Z}_{t,impute}$ in the correlation matrix of re-fitted autoregressive model after sweep operation,
  - $n(m-L)$ is number of transformed records used in equation (18) for re-fit autoregressive,

  - and $V(\tilde{\mathrm{Z}}_{\mathrm{t,impute}})$ is variance of $\tilde{Z}_{t,impute}$.
- $df_e^* = n(m - L) - p_{AR}$, and $p_{AR} = L^* + q_Z^*$ is the number of non-redundant parameters in re-fitted autoregressive model.

Based on these standard errors, t-test statistics and/or p-values may be computed and displayed according to standard definitions and output scheme of linear models.

(a) For each element α_j of $\hat{\boldsymbol{\alpha}}$ and the corresponding j-th diagonal element of $V(\hat{\boldsymbol{\alpha}})$, $j = 1, \dots, L$, compute the t-statistic $t_j = \alpha_j / \sqrt{V(\hat{\boldsymbol{\alpha}})_{jj}}$

(b) The p-value corresponding to t_j is $2\ \times$the value of the cumulative distribution function of a t-distribution with $\sum_{t=1}^{m} n_t - L^*$ degrees of freedom, i.e., $p_j = 2 \cdot \left(1 - P\left(t_{\sum_{t=1}^{m} n_t - L^*} \le |t_j|\right)\right)$.

4. Indicator of which method has been automatically chosen to model spatial covariances, either empirical covariance (EOF) or parametric variogram model.

5. Test statistics from goodness of fit tests for parametric model:

- Test statistic z_1, p-value p_1, level of significance γ_1 used for automated test for fit of slope parameter
- Test statistic z_2, p-value p_2, level of significance γ_2 used for testing homogeneity of variances

6. Parametric covariance parameters $\hat{\boldsymbol{\psi}}$ if parametric model has been chosen

3.6.3 Tests of effects in Mean Structure Model (Type III)

For each effect specified in the model, type III test matrix L is constructed and $H_0: L_i\beta = 0$ is tested. Construction of type III matrix *L* as well as generating estimable function (GEF) is based on the generating matrix H, which is the upper $D \times D$ submatrix of $\left(\check{X}_{aug}'\check{X}_{aug}\right)^{-1}\check{X}_{aug}'\check{X}_{aug}$, such that $L_i\beta$ is estimable. It involves parameters only for the given effect. For type III analysis, *L* does not depend on the order of effects specified in the model. If such a matrix cannot be constructed, the effect is not testable.

Then the *L* matrix is then used to construct the test statistic

$$F = \frac{\hat{\beta}'L'(L\Sigma L')^{-1}L\hat{\beta}}{r_c}$$

where

- $\hat{\beta}$ is the subvector of the first *D* elements of $\hat{\beta}_{aug}$ obtained in Step 5 of Section 3.5,
- $r_c = rank(L\Sigma L')$,
- Σ is the covariance matrix of $\hat{\beta}$, which is the upper $D \times D$ submatrix of $V(\hat{\beta}_{aug})$ defined in equation (25).

The statistic has an approximate *F* distribution. The numerator degrees of freedom $df1$ is r_c and the denominator degrees of freedom $df2$ is $nm - q - D^*$, where D^* is the number of non-redundant parameters in the first D parameters of refitted regression model obtained in Section 3.5. Then the p-values can be calculated accordingly.

An additional test also should be computed, which is similar to “corrected model” if there is an intercept or “model” if there is no intercept in ANOVA table in linear regression. Essentially, the null hypothesis is regression parameters (except intercept if there is on) are zeros. The test statistic would be the same as the above F statistic except the L matrix is from GEF. If there is no intercept, the L matrix is the whole GEF. If there is an intercept, the L matrix is GEF without the first row which corresponds to the intercept.

Statistics saved for Test of effects in Mean Structure Model (including corrected model or model):

- F statistics
- $df1$
- $df2$
- p-value

3.6.4 Location clustering for spatial structure visualization

Large spatial covariance matrix or correlation matrix are not suitable to demonstrate the relation among the locations. Grouping method, also called community detection or position analysis (Wasserman, 1994), can be used to identify some representative location clusters. To simplify the implementation, hierarchical clustering (Johnson, 1967) is used to detect clusters among locations based on STP model spatial statistics.

Please note location clustering is only supported when empirical nonparametric covariance model is used.

Given a set of *n* locations $\{s_1, \dots, s_n\}$ in STP to be clustered, and their corresponding spatial correlation matrix R, a *n*n* matrix, as the similarity matrix

$$R = \left[r_{ij}\right]_{i,j=1,\dots,n}$$

Given similarity threshold α with default value 0.2, and N_C with default value 10, the process of location clustering is described in following steps, which is based on the basic process of hierarchical clustering.

Step 1. Initialize the clusters and similarities:

- Assign each location s_i to a cluster C_i $(i = 1, \dots, n)$. So that for *n* locations, the total number of clusters $n_C = n$ at the beginning, and each cluster has just one location,
- Define the set of clusters: C,
- Define similarity matrix
  $$R^C = \left[r_{ij}^C\right]_{i,j=1,\dots,n}$$
  where the similarity r_{ij}^C between the clusters C_i and C_j is the similarity r_{ij} between location s_i and s_j.

Step 2. Find 2 clusters C_i and C_j in C with largest similarity $max\left(r_{ij}^C\right)$,

If $max\left(r_{ij}^C\right) > \alpha$:

- Merge C_i and C_j into a new cluster $C_{\langle i,j\rangle}$ to include all locations in C_i and C_j,
- Compute similarities between the new cluster $C_{\langle i,j\rangle}$ and other clusters C_k , $k \neq i\ and\ j$
$$r^C_{\langle i,j\rangle,k} = min\left(r^C_{ik}, r^C_{jk}\right)$$
- Update C by adding $C_{\langle i,j\rangle}$, discarding C_j and C_i. So $n_C = n_C - 1$.
- Update similarity matrix R^C by adding $r^C_{\langle i,j\rangle,k}$, discarding r^C_{ik} and r^C_{jk}, go to step 3.

If $max\left(r^C_{ij}\right) \leq \alpha$, go to step 4.

Step 3. Repeat step 2.

Step 4. For all the detected clusters with more than 1 location, compute following statistics:

- Cluster size: n_{C_i} is the number of locations in C_i,
- Closeness:
$$d_i = \frac{1}{n_{C_i}\left(n_{C_i} - 1\right)/2} \sum r_{kl}\,, \forall\ s_k, s_l \in C_i, and\ k \neq l.$$

Step 5. Define clusters for interactive visualization:

- $C_{closeness}$: The first N_C clusters sorted by descending closeness d_i,
- C_{size}: The first N_C clusters sorted by descending cluster size n_{C_i}.

Step 6. Output the union for location cluster visualization:

$$C^* = C_{closeness} \cup C_{size}$$

Statistics saved for spatial structure visualization including:

1. Number of excluded locations during handling of missing data
2. Spatial correlation matrix $\boldsymbol{R} = \left[r_{ij}\right]_{i,j=1,\dots,n}$
3. Statistics of each output location cluster in C^*:
    - Closeness d_i
    - Cluster size n_{C_i}
    - Coordinates of locations in this cluster

3.7 Results saved for prediction

1. Model coefficients $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\alpha}}$ and the covariance estimate $V\left(\widehat{\boldsymbol{\beta}}\right)$ as defined in (25).
2. Transformed regression residuals and predictors of L most recent observations for prediction:

$$\breve{\boldsymbol{Z}}_{m-l+1} = \boldsymbol{H'}^{-1}_S \boldsymbol{H}^{-1}_S \left(\boldsymbol{Y}_{m-l+1,impute} - \boldsymbol{X}_{m-l+1,aug} \widehat{\boldsymbol{\beta}}_{aug}\right), l = 1, \dots, L \qquad (27)$$

$$\breve{\boldsymbol{X}}_{m-l+1,impute} = \boldsymbol{H'}^{-1}_S \boldsymbol{H}^{-1}_S \boldsymbol{X}_{m-l+1,impute}, l = 1, \dots, L \qquad (28)$$

3. Indicator of which method has been chosen to model spatial covariances, either empirical

covariance (EOF) or parametric variogram model.

4. Parametric covariance parameters $\hat{\psi}$ if parametric model has been chosen.
5. Coordinates of locations s.
6. Number of unique time points used for model build, m.
7. Number of records with missing values in the data set used in model building, q.
8. Spatial covariance matrix Σ_S.
9. H_S^{-1}, inverse of Cholesky factor of spatial covariance matrix.

4 Prediction

We perform the following procedure to issue predictions for future time $m+1,\dots,m+H$ at prediction locations $\boldsymbol{G}=(\boldsymbol{g}_1,\dots,\boldsymbol{g}_N)$ using the results saved in the output file (see Figure 2). The input data set format should include location $\boldsymbol{G}$, predictors $\boldsymbol{X}$ for $t=m+1,\dots,m+H$.

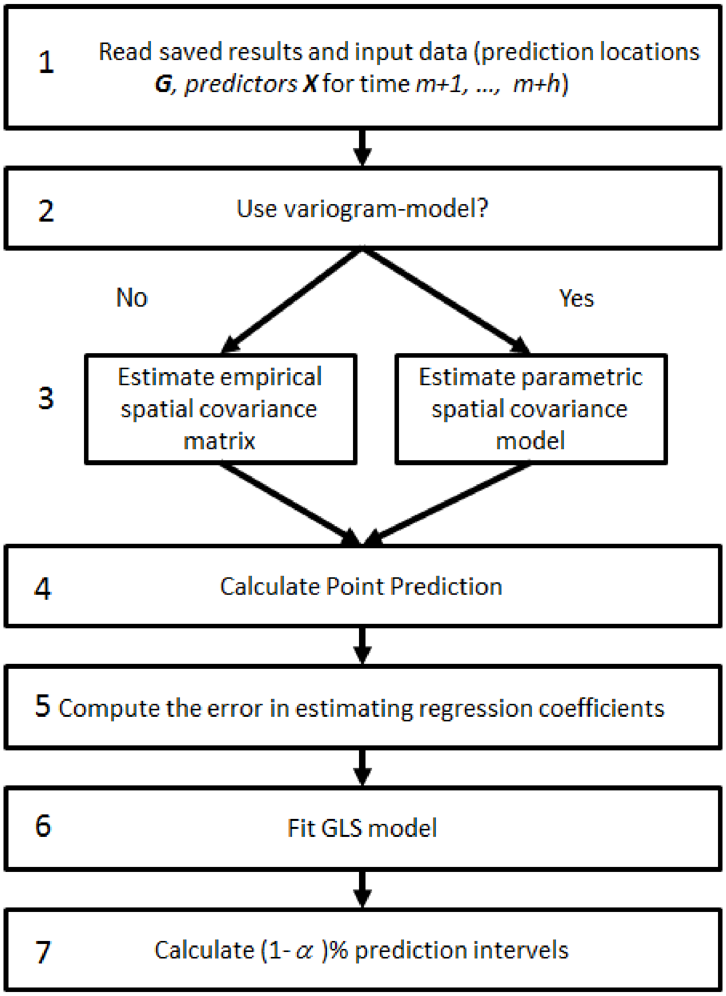

1
Read saved results and input data (prediction locations *G, predictors X* for time *m+1, …, m+h*)
2
Use variogram-model?
No
Yes
3
Estimate empirical spatial covariance matrix
Estimate parametric spatial covariance model
4
Calculate Point Prediction
5 Compute the error in estimating regression coefficients
6
Fit GLS model
7
Calculate (1- α)% prediction intervels

Figure 2. Flowchart of algorithm steps for model prediction

4.1 Point prediction

Step 1: Construct the $N \times n$ spatial covariance matrix to capture the spatial dependence between prediction grids $\boldsymbol{g} \in \boldsymbol{G}$ and original sample locations $\boldsymbol{s}$.

- If variogram-based spatial covariance matrix

$$V_S(\boldsymbol{g}) = V\big(\epsilon_t(\boldsymbol{g})\big) = \sigma^2 + \tau^2 \tag{29}$$

and

$$\boldsymbol{C}_S(\boldsymbol{G}) = \left\{Cov\big(\epsilon_t(\boldsymbol{g}_i), \epsilon_t(\boldsymbol{s}_j); \hat{\psi}\big)\right\}_{i=1,\dots,N;j=1,\dots,n} \tag{30}$$

according to (14) for all locations $\boldsymbol{g}$ (whether locations were included in the model build or not).

- If EOF-based spatial covariance function is used:

For locations $\mathrm{g_i}$ that are included in the original sample locations s, $Cov_{EOF}\big(\epsilon_t(g_i), \epsilon_t(s)\big)$ is equal to the row corresponding to location g_i in the empirical covariance matrix Σ_S and $V_S(g_i)$ is equal to the empirical variance at that location, i.e., the diagonal element of Σ_S corresponding to that location.

For locations g_i that were not included in the model build, calculate the spatial covariance in the following way:

(a) Perform eigendecomposition on the empirical covariance matrix

$$\boldsymbol{S} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}'$$

where $\boldsymbol{\Phi} = (\phi_1, \dots, \phi_n)$ with $\Phi_k = \big(\phi_k(s_1), \dots, \phi_k(s_n)\big)'$ is the $n \times n$ matrix of eigenvectors and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ is the $n \times n$ matrix of eigenvalues.

(b) Apply inverse distance weighting (IDW) (Shepard 1968) to interpolate eigenvectors to locations with no observations.

$$\phi_k(\boldsymbol{g}) = \sum_{i=1}^{n} \frac{w_i(\boldsymbol{g})\phi_k(\boldsymbol{s}_i)}{\sum_{j=1}^{n} w_j(\boldsymbol{g})}, k = 1, \dots, n$$

where

$$w_i(\boldsymbol{g}) = \frac{1}{\mathrm{dist}(\boldsymbol{g}, \boldsymbol{s}_i)^{\rho}}$$

is an Inverse Distance Weighting (IDW) function with $\rho \leq d$ for d-dimensional space and $\mathrm{dist}(\boldsymbol{g}, \boldsymbol{s}_i)$ may be any distance function. As a default value, use Euclidean distance with $\rho = 2$ and $\mathrm{dist}(\boldsymbol{g}, \boldsymbol{s}_i)^2 = (\boldsymbol{g} - \boldsymbol{s}_i)'(\boldsymbol{g} - \boldsymbol{s}_i)$.

(c) The EOF-based spatial variance-covariance functions are

$$V_S(\boldsymbol{g}) = V\big(\epsilon_t(\boldsymbol{g})\big) = \sum_{k=1}^{n} \lambda_n \phi_k^2(\boldsymbol{g}) \tag{31}$$

and

$$Cov\left(\epsilon_t(\boldsymbol{g}_i), \epsilon_t(\boldsymbol{s}_j)\right) = \sum_{k=1}^{n} \lambda_n \phi_k(\boldsymbol{g}_i)\phi_k(\boldsymbol{s}_j) \qquad (32)$$

and the corresponding $N \times n$ spatial covariance matrix

$$\boldsymbol{C}_S(\boldsymbol{G}) = \left\{Cov_{EOF}\left(\epsilon_t(\boldsymbol{g}_i), \epsilon_t(\boldsymbol{s}_j)\right)\right\}_{i=1,\dots,N;j=1,\dots,n} \qquad (33)$$

Note that under the EOF model, we allow for space-varying variances.

Step 2: Spatial interpolation to prediction locations g for the most recent L time units, $Z_{m-L+1}, \dots, Z_m$

$$\widehat{\boldsymbol{Z}}_{m-l+1}(\boldsymbol{G}) = \boldsymbol{C}_S(\boldsymbol{G})\boldsymbol{\Sigma}_{\boldsymbol{S}}^{-1}\boldsymbol{Z}_{m-l+1} = \boldsymbol{C}_S(\boldsymbol{G})\breve{\boldsymbol{Z}}_{m-l+1}, l = 1, \dots, L \qquad (34)$$

where $\widehat{\boldsymbol{Z}}_{m-l+1}(\boldsymbol{G})$ is a vector of length N.

Step 3: Iteratively forecast for future time $\mathrm{m} + 1, \dots, \mathrm{m} + \mathrm{H}$ at prediction locations $\boldsymbol{G}$.

$$\widehat{\boldsymbol{Z}}_{m+1}(\boldsymbol{G}) = \hat{\alpha}_1\widehat{\boldsymbol{Z}}_m(\boldsymbol{G}) + \cdots + \hat{\alpha}_L\widehat{\boldsymbol{Z}}_{m-L+1}(\boldsymbol{G}) \qquad (35)$$

$$\widehat{\boldsymbol{Z}}_{m+2}(\boldsymbol{G}) = \hat{\alpha}_1\widehat{\boldsymbol{Z}}_{m+1}(\boldsymbol{G}) + \cdots + \hat{\alpha}_L\widehat{\boldsymbol{Z}}_{m-L+2}(\boldsymbol{G}) \qquad (36)$$

$$\widehat{\boldsymbol{Z}}_{m+H}(\boldsymbol{G}) = \hat{\alpha}_1\widehat{\boldsymbol{Z}}_{m+H-1}(\boldsymbol{G}) + \cdots + \hat{\alpha}_L\widehat{\boldsymbol{Z}}_{m+H-L}(\boldsymbol{G}) \qquad (37)$$

where $\widehat{\boldsymbol{Z}}_{m+H}(\boldsymbol{G}), h = 1, \dots, H$ are vectors of length N.

Step 4: Incorporate predicted systematic effect

$$\widehat{\boldsymbol{Y}}_{m+H}(\boldsymbol{G}) = \widehat{\boldsymbol{Z}}_{m+H}(\boldsymbol{G}) + X_{m+h}(\boldsymbol{G})\widehat{\boldsymbol{\beta}}, \qquad h = 1, \dots, H \qquad (38)$$

where $\widehat{\boldsymbol{Y}}_{m+H}(\boldsymbol{G})$, $h = 1, \dots, H$ are vectors of length N.

4.2 Prediction intervals

Under the assumption of Gaussian Process and known variance components, the prediction error $\widehat{Y}_{m+H}(\boldsymbol{g}_i) - Y_{m+h}(\boldsymbol{g}_i)$ comes from two sources:

- The prediction error that would be incurred even if regression coefficients $\boldsymbol{\beta}$ were known.
- The error in estimating regression coefficients $\boldsymbol{\beta}$

  The variance of prediction error is thus

$$V\left[\widehat{Y}_{m+H}(\boldsymbol{g}_i) - Y_{m+h}(\boldsymbol{g}_i)\right]$$

$$= \left(\boldsymbol{X}'_{m+h}(\boldsymbol{g}_i) - \boldsymbol{C}'_{m+h}(\boldsymbol{g}_i)\boldsymbol{\Sigma}^{-1}\boldsymbol{X}_{impute}\right)\boldsymbol{V}(\widehat{\boldsymbol{\beta}})\left(\boldsymbol{X}'_{m+h}(\boldsymbol{g}_i) - \boldsymbol{C}'_{m+h}(\boldsymbol{g}_i)\boldsymbol{\Sigma}^{-1}\boldsymbol{X}_{impute}\right)' \qquad (39)$$

$$+\boldsymbol{V}_{m+h}(\boldsymbol{g}_i) - \boldsymbol{C}'_{m+h}(\boldsymbol{g}_i)\boldsymbol{\Sigma}^{-1}\boldsymbol{C}_{m+h}(\boldsymbol{g}_i) \qquad (40)$$

Expression (39) arises from the variance expression for universal kriging, while (40) is the variance of a predicted random effect with known variance of the random effects

(McCulloch et al. 2008, p.171).

- $C_{m+h}(g_i) = C_T(m+h) \otimes C_S(g_i)$ is the covariance vector of length nm between the prediction $Y_{m+h}(g_i)$ and measurements $Y_1(s), \dots, Y_m(s)$. Note that $C_T(m+h) = \{\gamma_T(m+h-t)\}_{t=1,\dots,m}$ is the AR(L) covariance vector of length m and $C_S(g_i) = \left\{Cov\left(Y_t(g_i), Y_t(s_j)\right)\right\}_{j=1,\dots,n}$ is the spatial covariance vector of length n.
- The $\mathrm{nm} \times \mathrm{nm}$ covariance matrix Σ is defined as to $\Sigma = \Sigma_T \otimes \Sigma_S$ and $\Sigma_T = \{\gamma_T|t-t'|\}_{t,t'=1,\dots,m}$. Note that Σ_S is a quantity stored after the model build step.
- $V_{m+h}(g_i) = V\left(Y_{m+h}(g_i)\right) = \gamma_T(0)V_S(g_i)$ is the variance of $Y_{m+h}(g_i)$.

- Note that expressions (39) and (40) are not computed explicitly, but instead are implemented as described in the following.

Computational process:

Step 1: Compute the error in estimating regression coefficients β in (39).

For $l = 1, \dots, L$, interpolate $\boldsymbol{X}$ to prediction locations $\boldsymbol{g}$ for the most recent L time units

$$\boldsymbol{P}_{m+1-l}(\boldsymbol{g}_i) = \boldsymbol{X'}_{m+1-l,impute}\boldsymbol{\Sigma}_{\boldsymbol{S}}^{-1}\boldsymbol{C}_S(\boldsymbol{g}_i) = \breve{\boldsymbol{X}}'_{m+1-l,impute}\boldsymbol{C}_S(\boldsymbol{g}_i) \qquad (41)$$

where $\boldsymbol{P}_{m+1-l}(\boldsymbol{g}_i)$ is a vector of dimension $D \times 1$. Define

$$\widehat{\boldsymbol{X}}_{m+h-l}(\boldsymbol{g}_i) = \begin{cases} \boldsymbol{P}_{m+h-l}(\boldsymbol{g}_i), & if\ h-l \leq 0; \\ \boldsymbol{X}_{m+h-l}(\boldsymbol{g}_i), & \text{otherwise.} \end{cases} \qquad (42)$$

For $t = m-L+1, \dots, m\ (h \leq l)$, we only have X at sample locations s, so $\hat{X}_t(g_i) = P_t(g_i)$, the interpolated values from $X_t(s)$; for $t > m$ (or $h > l$), we already input X at prediction locations g, so there is no need to interpolate and $\hat{X}_t(g_i) = X_t(g_i)$.

Then, for $h = 1, \dots, H$, recursively compute the $D \times 1$ vectors $W_{m+h}(g_i)$

$$W_{m+h}(g_i) = X_{m+h}(g_i) + \sum_{l=1}^{L} \hat{\alpha}_l \left(\widehat{W}_{m+h-l}(g_i) - \hat{X}_{m+h-l}(g_i)\right) \qquad (43)$$

where

$$\widehat{W}_{m+h-l}(g_i) = \begin{cases} 0, & \text{if } h-l \leq 0; (7) \\ W_{m+h-l}(g_i), & \text{otherwise.} \end{cases} \qquad (44)$$

The prediction error in estimating β, that is, expression (39) is thus

$$W'_{m+h}(g_i)V(\hat{\beta})W_{m+h}(g_i) \qquad (45)$$

where $V(\hat{\beta})$ is computed in (25).

Step 2: Compute the prediction error that would be incurred if regression coefficients β were known, i.e., equation (40).

- Compute $C_T(m+h)$ by AR(L) autocovariance function $\gamma_T(k)$ (McLeod 1975).

First, compute $\gamma_T(0), \dots, \gamma_T(L)$ by solving a linear system $AX = b$,

$$\begin{pmatrix} 1 & -\hat{\alpha}_1 & -\hat{\alpha}_2 & \dots & -\hat{\alpha}_{L-1} & -\hat{\alpha}_L \\ -\hat{\alpha}_1 & 1-\hat{\alpha}_2 & -\hat{\alpha}_3 & \dots & -\hat{\alpha}_L & 0 \\ -\hat{\alpha}_2 & -(\hat{\alpha}_1+\hat{\alpha}_3) & 1-\hat{\alpha}_4 & \dots & 0 & 0 \\ -\hat{\alpha}_3 & -(\hat{\alpha}_2+\hat{\alpha}_4) & -(\hat{\alpha}_1+\hat{\alpha}_5) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ -\hat{\alpha}_{L-2} & -(\hat{\alpha}_{L-3}+\hat{\alpha}_{L-1}) & -(\hat{\alpha}_{L-4}+\hat{\alpha}_L) & \dots & 0 & 0 \\ -\hat{\alpha}_{L-1} & -(\hat{\alpha}_{L-2}+\hat{\alpha}_L) & -\hat{\alpha}_{L-3} & \dots & 1 & 0 \\ -\hat{\alpha}_L & -\hat{\alpha}_{L-1} & -\hat{\alpha}_{L-2} & \dots & -\hat{\alpha}_1 & 1 \end{pmatrix} \begin{pmatrix} \gamma_T(0) \\ \gamma_T(1) \\ \gamma_T(2) \\ \gamma_T(3) \\ \vdots \\ \gamma_T(L-2) \\ \gamma_T(L-1) \\ \gamma_T(L) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{46}$$

Note that the first element of the vector on the right hand side (the variance of the measurement error) is fixed to be one, to account for the normalization through the spatial variance-covariance structure.

For $k = L+1, \dots, m+H-1$, recursively compute

$$\gamma_T(k) = \hat{\alpha}_1 \gamma_T(k-1) + \cdots + \hat{\alpha}_L \gamma_T(k-L) \tag{47}$$

Remark: To construct the $(L+1) \times (L+1)$ matrix A,

$$A_{ij} = \begin{cases} -[\alpha_{i-1}], & j = 1; i = 1, \dots, L+1 \\ -[\alpha_{i-j}] - [\alpha_{i+j-2}], & j = 2, \dots, L+1; i = 1, \dots, L+1. \end{cases} \tag{48}$$

where

$$[\alpha_k] = \begin{cases} -1, & k = 0; \\ 0, & k < 0 \text{ or } k > L; \\ \hat{\alpha}_k, & 0 < k \le L. \end{cases} \tag{49}$$

- Compute the approximated factorization of Σ_T^{-1} such that $R'R \approx \Sigma_T^{-1}$, where R is a $(m-L) \times m$ matrix (follows from Cholesky or Gram-Schmidt orthogonalization, see for example Fuller 1975):

$$R = \begin{pmatrix} -\hat{\alpha}_L & \dots & -\hat{\alpha}_1 & 1 & 0 & 0 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \dots & 0 & -\hat{\alpha}_L & \dots & -\hat{\alpha}_1 & 1 & 0 & 0 \\ \dots & \dots & 0 & -\hat{\alpha}_L & \dots & -\hat{\alpha}_1 & 1 & 0 \\ \dots & \dots & \dots & 0 & -\hat{\alpha}_L & \dots & -\hat{\alpha}_1 & 1 \end{pmatrix} \tag{50}$$

- Compute the value of expression (40):

$$\gamma_T(0) V_S(g_i) - (C'_T(m+h) \otimes C'_S(g_i))(R'R \otimes H_S^{-1'} H_S^{-1})(C_T(m+h) \otimes C_S(g_i)) \tag{51}$$

where $C'_S(g_i)$ is a the row of $C_S(G)$ corresponding to location g_i.

Step 3: The $(1 - \alpha\%)$ prediction interval is

$$\hat{Y}_{m+h}(g_i) \pm t_{nm-q-D^*,\alpha/2}\sqrt{V[\hat{Y}_{m+h}(g_i) - Y_{m+h}(g_i)]} \qquad (55)$$

where $V[\hat{Y}_{m+h}(g_i) - Y_{m+h}(g_i)]$ is the sum of equations (39) and (40) as computed in expressions (45) and (51), respectively. $t_{nm-q-D,\alpha/2}$ is defined as $P(X \leq t_{nm-q-D^*,\alpha/2}) = 1 - \alpha/2$ where X follows t-distribution with degree freedom $nm - q - D^*$. The default value for α is 0.05.

As final output from the prediction step, point prediction, variances of point predictions and prediction interval (lower and upper bounds) are issued for each specified (location, time).

We remark that to perform what-if-analysis, a set of $\boldsymbol{X}$ variables under the new settings need to be provided. Then we re-run the prediction algorithm described in Section 4 to obtain prediction results under adjusted settings.

References

[1] Brockwell, P., Davis, R.A. (2002), *Introduction to Time Series and Forecasting*, Second Edition, New York: Springer.

[2] Cohen, A., Johnes, R. (1969), “Regression on a Random Field”, *Journal of the American Statistical Association*, 64 (328), 1172-1182.

[3] Cressie, N. (1993), *Statistics for Spatial Data*, Revised Edition, Wiley-Interscience.

[4] Creutin, J.D., Obled, C. (1982), “Objective Analyses and Mapping Techniques for Rainfall Fields: an Objective Comparison”, *Water Resources Research*, 18(2), 413-431.

[5] Fuller, W.A. (1975), *Introduction to Statistical Time Series*, John Wiley & Sonse, New York, New York.

[6] Johnson S. (1967), “Hierarchical Clustering Schemes”, *Psychometrika*, 32(3), 241-254.

[7] McCulloch, C.E., Searle, S.R., Neuhaus, J.M. (2008), *Generalized, Linear and Mixed Models*, Second Edition, John Wiley & Sons, Hoboken, New Jersey.

[8] McLeod, I. (1975), “Derivation of the Theoretical Autocovariance Function of Autoregressive-Moving Average Time Series”, *Applied Statistics*, 24(2), 255-256.

[9] Shepard, D. (1968), “A two-dimensional interpolation function for irregularly-spaced data”, *Proceedings of the 1968 ACM National Conference*, 517-524.

[10] Wasserman S. (1994), *Social network analysis: Methods and applications*. Cambridge university press.

Temporal Causal Modeling Algorithms

1. Introduction

Forecasting and prediction are important tasks in real world applications that involve decision making. In such applications, it is important to go beyond discovering statistical correlations and unravel the key variables that influence the behaviors of other variables using an algebraic approach. Many real world data, such as stock price data, are temporal in nature; that is, the values of a set of variables depend on the values of another set of variables at several time points in the past. Temporal causal modeling, or TCM, refers to a suite of methods that attempt to discover key temporal relationships in time series data. This chapter describes a particular method to discover temporal relationships using a combination of Granger causality and regression algorithms for variable selection. Although this treatment strives to be self-contained, a minimal set of papers describing the design principles behind the method can be found in [Lozano et al., 2011, Lozano et al., 2009, Arnold et al., 2007][1].

The rest of the chapter is organized as follows. Section 2 lays the groundwork for the TCM algorithm (notation and brief history) and explains the greedy orthogonal matching pursuit (GOMP) [Lozano et al., 2011] algorithm that is used. Section 3 describes the techniques used to fit and forecast time series and compute approximated forecasting intervals. Section 4 describes scenario analysis, which refers to a capability of the TCM product to "play-out" the repercussions of artificially setting the value of a time series. Section 5 describes the detection of outliers, and Section 6 discusses how potential causes for outliers can be established using root cause analysis.

2. Model

Introduced by Clive Granger [Granger, 1980], Granger causality in time series is based on the intuition that a cause should necessarily precede its effect, and that if time series a causally affects time series b, then the past values of a should be useful in predicting the future values of a. More specifically, time series a is said to "Granger cause" time series b if the accuracy of regressing for b in terms of past values of both a and b is statistically significantly better than regressing just with past values of b. If the time series have T time points and are denoted by $\{a_t\}_{t=1}^{T}$and $\{b_t\}_{t=1}^{T}$, then the following regressions are performed:

$$b_t \approx \sum_{j=1}^{L} \alpha_j \, a_{t-j} + \sum_{j=1}^{L} \beta_j \, b_{t-j} \tag{1}$$

$$b_t \approx \sum_{j=1}^{L} \beta_j \, b_{t-j} \tag{2}$$

Here L is the number of lags; that is, the value of b at time t can only be determined by values of other time series at times $\{t-1, t-2, \cdots, t-L\}$. If Equation (1) is statistically more significant (using some test for significance) than Equation (2), then a is deemed to Granger cause b.

[1] The methods described in this chapter are particularly useful for under-determined systems, where the number of time series (n) far exceeds the number of samples (m); that is $n \gg m$. Although these methods function for both over-determined ($m \gg n$) and fully-determined ($n == m$) systems, there are other approaches to pursue for such systems.

2.1 Graphical Granger Modeling

The classical definition of Granger causality is defined for a pair of time series. In the real world, we are interested in finding not one, but *all* the significant time series that influence the target time series. In order to accomplish this, we use group greedy (ℓ_0) regression algorithms with variable selection (see Section 2.3). An important feature of our TCM algorithm is that it groups influencer/predictor variables; that is, we are interested in predicting whether time series a as a whole – $\{a_{t-1}, a_{t-2}, \cdots, a_{t-L}\}$ – has influence over time series b. Such grouping is a more natural interpretation of causality and also helps sparsify the solution set. For example, without such grouping we may select the time-lagged series a_{t-2} to model b_t but not select any other value of a, which increases the number of choices for variable selection L-fold, where L is the number of lags that is allowed.

2.2 Notation

The following notation is used throughout this chapter unless otherwise stated:

Table 1: Notation

Notation	Type	Description
$\mathcal{N}$	—	Set of natural numbers
$\mathcal{R}$	—	Set of real numbers
$\backslash$	—	Regression solve operator
$\lvert . \rvert$	$\mathcal{N}$	Size operator
$\Vert . \Vert_2$	$\mathcal{R}$	ℓ_2 norm of a vector, i.e., $\Vert \boldsymbol{z} \Vert_2 = \sqrt{\sum_i z_i^2}$
m	$\mathcal{N}$	Number of time points
n	$\mathcal{N}$	Number of time series
L	$\mathcal{N}$	Number of lags for each target, $L < m$
$\boldsymbol{X}$	$\mathcal{R}^{m\times n}$	Design matrix of input series
$\boldsymbol{y}$	$\mathcal{R}^{m\times 1}$	Target series vector
G	$G: \mathcal{R}^{m\times n} \times J \times L \to \mathcal{R}^{(m-L)\times\lvert J\rvert L}$ $J = \{j_1, j_2, \cdots\}, 1 \le j_k \le n$	Computes lag matrix for the set of column indices in J
M	$M: \mathcal{R}^{m\times k} \to \mathcal{R}^{k\times 1}$	Computes means for k series
S	$S: \mathcal{R}^{m\times k} \to \mathcal{R}^{k\times 1}$	Computes standard deviations for k series
ϵ	$\mathcal{R}$	Tolerance value for stopping criterion
K^*	$\mathcal{N}$	Max number of predictors selected or maximum number of iterations
K	$\mathcal{N}$	Actual number of predictors selected for a target series $\boldsymbol{y}$
$\widehat{\boldsymbol{\beta}}^*$	$\mathcal{R}^{k\times 1}, 0 \le k \le KL$	Estimated coefficients for predictors on the transformed scale

In this section, we introduce the algorithm that is used to construct the temporal causal model. The list of symbols used in the rest of this chapter is summarized in Table 1. Most of the symbols are self-explanatory; however, the function G, which stands for grouping, requires some additional explanation. G is a function that takes a matrix ($\mathcal{R}^{m\times n}$), a set of column indices J, and a lag value L and constructs a lag matrix that has $(m-L)$ rows and $(\lvert J\rvert L)$ columns. Basically, for every column index $j \in J$, G constructs a $(m-L)\times L$ lag matrix by carefully unrolling the jth column of the input matrix. An example of G's action is shown below:

$$G\left(\boldsymbol{X}=\begin{bmatrix} a_1 & b_1 & c_1 & d_1\\ a_2 & b_2 & c_2 & d_2\\ a_3 & b_3 & c_3 & d_3\\ a_4 & b_4 & c_4 & d_4\\ a_5 & b_5 & c_5 & d_5 \end{bmatrix}, J=\{1\}, L=2\right)\rightarrow\begin{bmatrix} a_2 & a_1\\ a_3 & a_2\\ a_4 & a_3 \end{bmatrix}$$

In this example, the input matrix $\boldsymbol{X}\in\mathcal{R}^{5\times 4}$ has 4 time series ($n=4$) and five time points per time series ($m=5$). The lag matrix associated with the time series in column 1, when L (lag) is 2, is produced by invoking $G(\boldsymbol{X},\{1\},2)$. Note that the lag matrix consists of the lag-1 vector of $\boldsymbol{X}$ as the first column, the lag-2 vector as the second column, up to the lag- L vector as the L^{th} column. Similarly, the functions (M,S) accept any input matrix and compute the mean and the standard deviation, respectively, of the matrix's columns. For purposes of numerical stability, and to increase interpretability during modeling, columns of the lagged matrix are both centered by the column means and scaled by the column standard deviations [2]. On the other hand, the target $\boldsymbol{y}$ is *only centered*. An example of mean centering and scaling for the lagged matrices is shown below:

$$\left(\begin{bmatrix} a_1 & b_1\\ a_2 & b_2\\ a_3 & b_3 \end{bmatrix}\right)\rightarrow\begin{bmatrix} \dfrac{a_1-a_\mu}{a_\sigma} & \dfrac{b_1-b_\mu}{b_\sigma}\\ \dfrac{a_2-a_\mu}{a_\sigma} & \dfrac{b_2-b_\mu}{b_\sigma}\\ \dfrac{a_3-a_\mu}{a_\sigma} & \dfrac{b_3-b_\mu}{b_\sigma} \end{bmatrix}$$

Here, (a_μ , a_σ) and (b_μ , b_σ) are the means and standard deviations of the first and the second columns, (a,b) respectively.

2.3 Group Orthogonal Matching Pursuit (GOMP)

Algorithm 1: GOMP

```
Input: $\boldsymbol{X},\boldsymbol{y},G,M,S,L,\epsilon,K^*,J_{sel}^0,J_{sel}^{\sim}$.
Output: $J_{sel}$, $\widehat{\boldsymbol{\beta}}^*$.
$\boldsymbol{X}_{aug}^0=G(X,J_{sel}^0,L)$;
for $i\in[1,(m-L)]$ do $\boldsymbol{X}_{aug}^0(i,:)=\frac{\boldsymbol{X}_{aug}^0(i,:)-M(\boldsymbol{X}_{aug}^0)^T}{S(\boldsymbol{X}_{aug}^0)^T}$;
$\widehat{\boldsymbol{\beta}}^{*0}=\boldsymbol{X}_{aug}^0\setminus(\boldsymbol{y}-M(\boldsymbol{y}))$;
$\boldsymbol{r}^0=\boldsymbol{y}-M(\boldsymbol{y})-\boldsymbol{X}_{aug}^0\widehat{\boldsymbol{\beta}}^{*0}$;
if any redundant series are found, delete them in $J_{sel}^0$;
if ($|J_{sel}^0|\geq K^*$), then $J_{sel}^0=J_{sel}^0(1:K^*)$, update $\widehat{\boldsymbol{\beta}}^{*0}$ , return $J_{sel}^0$, $\widehat{\boldsymbol{\beta}}^{*0}$ and stop;
otherwise update $\widehat{\boldsymbol{\beta}}^{*0}$ and $\boldsymbol{r}^0$;
for $k\in 1,2,3\ldots(K^*-|J_{sel}^0|)$ do
    $j^k=argmin(\boldsymbol{X},\boldsymbol{r}^{k-1},G,M,S,L,\epsilon,J_{sel}^0,J_{sel}^{\sim})$;
    if $j^k=-1$, return $J_{sel}^{(k-1)}$ and $\widehat{\boldsymbol{\beta}}^{*(k-1)}$ and stop;
    $\boldsymbol{X}_{aug}^k=G(\boldsymbol{X},J_{sel}^{k-1}\cup j^k,L)$;
```

[2] Although each column of the lagged matrix has a different mean and standard deviation, due to the structure of these columns, it is possible to compute the mean and the standard deviation of the time series itself and use those to center and scale the lagged columns.

```
12      for $i \in [1,(m-L)]$ do
13         $\boldsymbol{X}_{aug}^{k}(i,:) = \frac{\boldsymbol{X}_{aug}^{k}(i,:)-M(\boldsymbol{X}_{aug}^{k})^{T}}{S(\boldsymbol{X}_{aug}^{k})^{T}}$;
14      $\widehat{\boldsymbol{\beta}}^{*k} = \boldsymbol{X}_{aug}^{k} \setminus \left(\boldsymbol{y} - M(\boldsymbol{y})\right)$;
15      $\boldsymbol{r}^{k} = \boldsymbol{y} - M(\boldsymbol{y}) - \boldsymbol{X}_{aug}^{k}\widehat{\boldsymbol{\beta}}^{*k}$;
16      $J_{sel}^{k} = J_{sel}^{k-1} \cup j^{k}$;
17     if $\left\|\boldsymbol{r}^{k}\right\|_{2} \leq \epsilon$, break;
18   return $J_{sel}^{k}$, $\widehat{\boldsymbol{\beta}}^{*k}$.
```

We begin by describing Algorithm 1: GOMP, which will be used to establish causality of time-series data. This algorithm receives the variables $\boldsymbol{X}, \boldsymbol{y}, G, M, S, L, \epsilon, K^{*}$ (described in Table 1) as input. Briefly, $\boldsymbol{y} \in \mathcal{R}^{(m-L)\times 1}$ is a target vector for which we want to establish the Granger causality (note that we have excluded the first L values of $\boldsymbol{y}$**)**. In contrast, $\boldsymbol{X} \in \mathcal{R}^{m\times n}$ is the input *unlagged* time series data. L is the number of lags for each predictor in each target series, K^{*} is the maximum number of predictors to be selected per-target, and ϵ determines whether a new predictor needs to be added. In addition, G, M and S are grouping, centering, and scaling functions which have been described in Section 2.2. J_{sel}^{0} is the set of pre-selected predictor indices for $\boldsymbol{y}$, and always contains the lagged $\boldsymbol{y}$. $J_{sel}^{\sim}$ is the set of forbidden predictors, if any, for $\boldsymbol{y}$. If there are no forbidden predictors, then $J_{sel}^{\sim} = \emptyset$. Given these, the goal is to greedily find predictors that solve the system $\boldsymbol{X\beta} = \boldsymbol{y}$ subject to sparsity constraints.

The greedy algorithm approximates an ℓ_0–sparse solution by itertively choosing the best predictor for addition at each iteration. We use superscripts to denote the iteration number in Algorithm 1. For example, J_{sel}^{0} represents the initial values of J_{sel} at the 0*th* iteration (before the actual iteration starts). The first part of the algorithm (lines 1 – 4) constructs and solves a linear system consisting of the predictors in J_{sel}^{0} to obtain $\boldsymbol{\beta}^{*0}$, the coefficient vector for predictors on the transformed scale. At the end of this first part, we have $\boldsymbol{r}^{0}$, the initial residual. Then check whether there are redundant predictor series in J_{sel}^{0}. If yes, then delete them. If the number of predictor series in the (updated) J_{sel}^{0} is equal to or larger than the maximum number of iterations (i.e., $|J_{sel}^{0}| \geq K^{*}$) then keep the first K^{*} predictor series in J_{sel}^{0}, update $\boldsymbol{\beta}^{*0}$, return J_{sel}^{0} and $\boldsymbol{\beta}^{*0}$, and stop the process (line 6); otherwise (i.e., $|J_{sel}^{0}| < K^{*}$), update $\boldsymbol{\beta}^{*0}$ and $\boldsymbol{r}^{0}$ (line 7) if any redundant predictor series were deleted. Then start the iterative process to add one predictor series at a time (line 8). The first step in predictor selection (line 9) consists of an **argmin** function that systematically goes over each eligible predictor and evaluates its goodness (see Algorithm 2). This step is the performance critical portion of the algorithm and can be searched in parallel. At the end of the step, j^{k}, the index corresponding to the best predictor is available. However, if no suitable predictor is found in the **argmin** function (i.e., $j^{k} = -1$), then return $J_{sel}^{k-1}$and $\boldsymbol{\beta}^{*(k-1)}$ and stop (line 10). The next part (lines 11 – 14) re-estimates the model coefficients by adding j^{k} to the model. Line 15 updates the residual, $\boldsymbol{r}^{k}$, for this model and line 16 adds j^{k} to the model. Finally, if the ℓ_2 norm of the current residuals is equal to or smaller than the tolerance value (i.e., ($\|\boldsymbol{r}^{k}\|_2 \leq \epsilon$)), then the iterative process is terminated.

Note that if the tolerance ϵ is achieved by adding j^{k}, then no new iterations are required and the iterative process is terminated. Thus the actual number of predictors selected, K, can be less than the maximum number of iterations, (i.e., $K \leq K^{*}$). However, if the tolerance ϵ is set very small, then it is highly unlikely that such a situation will happen.

Algorithm 2: argmin

Input: $\boldsymbol{X}, \boldsymbol{r}, G, M, S, L, \epsilon_2, K^{*}, J_{sel}^{0}, J_{sel}^{\sim}$.

```
Output: $j_{sel}$: Selected group index.
1   $cost = \|\boldsymbol{r}\|_2^2, j_{sel} = -1$;
2   for $j \in 1, 2, 3 \dots n$ do
3       if $j \in J_{sel} \parallel j \in J_{sel}^{\sim}$ continue;
4       $\boldsymbol{X}_{G_j} = G(\boldsymbol{X}, j, L)$;
5       for $i \in [1, (m - L)]$ do $\boldsymbol{X}_{G_j}(i,:) = \frac{\boldsymbol{X}_{G_j}(i,:) - M(\boldsymbol{X}_{G_j})^T}{S(\boldsymbol{X}_{G_j})^T}$;
6       $\widehat{\boldsymbol{\beta}}_j = \boldsymbol{X}_{G_j} \setminus \boldsymbol{r}$;
7       $\boldsymbol{r}_j = \boldsymbol{r} - \left(\boldsymbol{X}_{G_j} \widehat{\boldsymbol{\beta}}_j\right)_{G_j}$;
8       if $\|\boldsymbol{r}_j\|_2^2 < (cost - \epsilon_2)$, then $(cost, j_{sel}) = \left(\|\boldsymbol{r}_j\|_2^2, j\right)$;
9   return $j_{sel}$.
```

The implementation of the **argmin** function (line 8, Algorithm 1) is shown in Algorithm 2. The algorithm first assigns the initial cost to be the square of the ℓ_2 norm of the current residuals, and the selected group index to be -1 (line 1). Then it loops over each series group, first checking if the time series being considered for addition (j) has already been added to the solution J_{sel} or if it is a forbidden predictor (line 3). If the current group (j) is not yet selected, the lagged transformed matrix corresponding to this time series ($\boldsymbol{X}_{G_j}$) is constructed using the G, M and S functions (lines 4 and 5). After grouping and transforming X_{G_j}, the residual $(\boldsymbol{r}_j)$ corresponding to the candidate time series *j* is computed by first regressing $\boldsymbol{r}$ on $\boldsymbol{X}_{G_j}$ (line 6), and then computing the residual (line 7). Finally, the current time series is selected as the leading candidate if the square of the ℓ_2 norm of its residual $(\boldsymbol{r}_j)$ is lower than the previous estimate minus a threshold value, ϵ_2. Including such a threshold value prevents selecting an (almost) identical series.

The loop in Algorithm 2 (line 2) can be thought of as iterating over all candidate series. For each candidate series, the following computations are carried out: (1) a filter is applied in line 3 to ensure that it is a valid candidate; (2) lines 4 and 5 map the current candidate to the transformed matrix ($\boldsymbol{X}_{G_j}$) that represents the lag matrix to be used; (3) lines 6 and 7 evaluate the goodness of the current candidate by first solving a dense linear system and then computing the residual; (4) line 8 applies a predicate to check if the current candidate series is better than previously evaluated candidates. Notice that the predicate (line 8) is associative and commutative; therefore, Algorithm 2 can be parallelized by dividing the iteration space ([1,*n*]) into chunks and executing each chunk in parallel. To get the globally best group, it is sufficient to *reduce* the groups that were selected by each parallel instance in a tree-like fashion by applying the predicate in line 8.

2.4 Selecting L

Both Algorithms 1 and 2 accept L as an input parameter which can be specified by user. If L is not explicitly specified then the following heuristic approach can be used to determine L based on m (# of time points) and s (periodicity or seasonal length):

(1) If $s > 1$ and $m \geq 4s$, then $L = \min(s, 20)$.

(2) If $s = 1$ or $m < 4s$, then $L = 5$.

2.5 AR(L) Model

Out of the n series in the data, some series may be used as predictors only, so no TCM models are built for them. However, if they are selected as predictors for some target series, then simple models need to be built for them in order to do forecasting. For example, suppose that time series 1 is a selected predictor for time series 2, but there is no model built for time series 1. While a model for time series 1 is not needed in order to forecast time series 2 at time $(t+1)$ (where t is the latest time in the data), forecasts for time (t + 2) require values of time series 1 for time $(t+1)$, which then requires a model for time series 1.

Hence, for each predictor-only series, a simple auto-regressive (AR) model is built using the same lag, L, as used for the target series. This model, called an AR(L) model, can be constructed using Algorithm 1 by specifying J_{sel}^0 to be the target itself and setting the maximum number of predictors to be 1.

2.6 Post-estimation steps

Algorithm 1 selects the best predictors (time series) to model a target series $\boldsymbol{y}$. Without loss of generality, we assume that the model for $\boldsymbol{y}$ is $\boldsymbol{y} = \overline{\boldsymbol{y}} + \boldsymbol{X}_G^* \widehat{\boldsymbol{\beta}}^* + \boldsymbol{r} = \widehat{\boldsymbol{y}} + \boldsymbol{r}$, where $\boldsymbol{X}_G^*$ is the selected predictor series matrix with the lagged terms on the transformed scale, $\widehat{\boldsymbol{\beta}}^*$ is the estimated standardized coefficient vector, and $\boldsymbol{r} = \boldsymbol{y} - \widehat{\boldsymbol{y}}$ is the residual vector**.**

However, this is not the end of modeling. Several post processing steps are needed in order to complete the modeling process for $\boldsymbol{y}$. The steps include three parts: (1) coefficients and statistics inference; (2) tests of model effects; (3) model quality measures.

2.6.1 Coefficients and statistical inference

The results of Algorithm 1 include $\widehat{\boldsymbol{\beta}}^*$ and $(\boldsymbol{X}^{*\mathrm{T}}\boldsymbol{X}^*)^-$ (by solving the linear system from Cholesky decomposition), where superscript T means the transpose of a matrix or vector, and $(\boldsymbol{z})^-$ is a generalized inverse of the $\boldsymbol{z}$ matrix. Based on these quantities, the first step is to compute coefficient estimates, their standard errors, and statistical inference on the original scale.

Table 2: Additional notation

Notation	Description
K	Actual number of predictors selected (including target itself) for $\boldsymbol{y}$, i.e., $K = \lvert J_{sel} \rvert$.
p	Number of coefficient estimates in $\widehat{\boldsymbol{\beta}}^*$, i.e., $p = K \times L$
$\boldsymbol{p}^c$	Number of non-redundant coefficient estimates in $\widehat{\boldsymbol{\beta}}^*$, $\boldsymbol{p}^{\boldsymbol{c}} \le p$
$\boldsymbol{X}_G^*$	Selected predictor series matrix with lagged terms on the transformed scale. This is an $(m-L) \times p$ matrix as $\boldsymbol{X}_G^* = [\boldsymbol{X}_{G_1}^*, \dots, \boldsymbol{X}_{G_K}^*]$ with $\boldsymbol{X}_{G_j}^* = G(\boldsymbol{X}^*, j, L) = [\boldsymbol{X}_{G_{j1}}^*, \dots, \boldsymbol{X}_{G_{jL}}^*]$ (an $(m-L) \times L$ matrix).
$\boldsymbol{X}_G$	Selected predictor series matrix on the original scale. This is an $(m-L) \times (p+1)$ matrix as $\boldsymbol{X}_G = [\mathbf{1}, \boldsymbol{X}_{G_1}, \dots, \boldsymbol{X}_{G_K}] = [\mathbf{1}, \boldsymbol{X}_{G_{11}}, \dots, \boldsymbol{X}_{G_{1L}}, \cdots, \boldsymbol{X}_{G_{K1}}, \dots, \boldsymbol{X}_{G_{KL}}]$, where $\mathbf{1}$ is a column vector of 1's corresponding to an intercept.
$\widehat{\boldsymbol{\beta}}$	Unstandardized coefficient estimates vector (corresponding to $\boldsymbol{X}_G$), which is a $(p+1) \times$

	1 vector. The first element, $\hat{\beta}_0$, is the intercept estimate.
$\hat{\sigma}^2$	Estimated variance of the model based on residuals.
$\boldsymbol{\Sigma}^*$	Covariance matrix of standardized coefficient estimates on the transformed scale, i.e., $\boldsymbol{\Sigma}^* = \hat{\sigma}^2(\boldsymbol{X}_G^{*\mathrm{T}}\boldsymbol{X}_G^*)^-$. The j^{th} diagonal element is $\hat{\sigma}^2_{\hat{\beta}_j^*}$ and its square root, $\hat{\sigma}_{\hat{\beta}_j^*}$, is the standard error of the j^{th} standardized coefficent estimate.
$\boldsymbol{\Sigma}$	Covariance matrix of unstandardized coefficient estimates on the original scale. The j^{th} diagonal element is $\hat{\sigma}^2_{\hat{\beta}_j}$ and its square root, $\hat{\sigma}_{\hat{\beta}_j}$, is the standard error of the j^{th} unstandardized coefficent estimate.
$\mathbf{M}$	Centering vector of $\boldsymbol{X}$, i.e., $\mathbf{M} = \left[\mathrm{M}_1, \dots, \mathrm{M}_p\right]^{\mathrm{T}}$, where $\mathrm{M}_j = M(\boldsymbol{X}_j)$ is the mean of $\boldsymbol{X}_j$.
$\mathbf{S}$	Scaling matrix of $\boldsymbol{X}$, i.e., $\mathbf{S} = \mathrm{diag}\left[\mathrm{S}_1, \dots, \mathrm{S}_p\right]$, where $\mathrm{S}_j = S(\boldsymbol{X}_j)$ is the standard deviation of $\boldsymbol{X}_j$.
$\boldsymbol{A}$	Transformation matrix of $\boldsymbol{X}$ to $\boldsymbol{X}^*$, i.e., $\boldsymbol{A} = \begin{bmatrix} -\mathbf{M}^{\mathrm{T}}\mathbf{S}^{-1} \\ \mathbf{S}^{-1} \end{bmatrix}$, which is a $(p+1) \times p$ vector. Note that $\boldsymbol{X}_G^* = \boldsymbol{X}_G\boldsymbol{A}$.

The relationship between $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\beta}}^*$ is $\hat{\boldsymbol{\beta}} = \boldsymbol{A}\hat{\boldsymbol{\beta}}^* + [\bar{y}, 0, \dots, 0]^{\mathrm{T}}$ and the relationship between $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^*$ is $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{\Sigma}^*\boldsymbol{A}^{\mathrm{T}}$. The relevant statistics are computed as follows:

- **Unstandardized coefficient estimates**

$$\hat{\beta}_j = \mathrm{S}_j^{-1}\hat{\beta}_j^*, \quad j = 1, \dots, p \tag{3}$$

$$\hat{\beta}_0 = \bar{\boldsymbol{y}} - \mathbf{M}^{\mathrm{T}}\mathbf{S}^{-1}\hat{\boldsymbol{\beta}}^* \tag{4}$$

- **Standard errors of unstandardized coefficient estimates**

$$\hat{\sigma}_{\hat{\beta}_j} = \frac{\hat{\sigma}_{\hat{\beta}_j^*}}{\mathrm{S}_j}, \quad j = 1, \dots, p \tag{5}$$

$$\hat{\sigma}_{\hat{\beta}_0} = sqrt\left(\left[\frac{\mathrm{M}_1}{\mathrm{S}_1}, \dots, \frac{\mathrm{M}_p}{\mathrm{S}_p}\right]\boldsymbol{\Sigma}^*\begin{bmatrix} \frac{\mathrm{M}_1}{\mathrm{S}_1} \\ \vdots \\ \frac{\mathrm{M}_p}{\mathrm{S}_p} \end{bmatrix}\right) \tag{6}$$

where $\boldsymbol{\Sigma}^* = \hat{\sigma}^2(\boldsymbol{X}_G^{*\mathrm{T}}\boldsymbol{X}_G^*)^-$ and $\hat{\sigma}^2 = SS_e/df_e$ with $SS_e = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \sum_{t=1}^{t=(m-L)}(y_t - \hat{y}_t)^2$ and $df_e = m - L - p^c - 1$.

- **t-statistics for coefficient estimates**

$$t_j = \frac{\hat{\beta}_j}{\hat{\sigma}_{\hat{\beta}_j}}, \quad j = 0, 1, \dots p, \tag{7}$$

which follows an asymptotic *t* distribution with df_edegrees of freedom. Then the *p*-value is computed as

$$p_{t_j} = 2 \times \left(1 - prob\left(t_{df_e} \leq |t_j|\right)\right) \tag{8}$$

- **$\mathbf{100(1-\alpha)\%}$ confidence internals**

$$\hat{\beta}_j \pm \hat{\sigma}_{\hat{\beta}_j} \times t_{\alpha/2,df_e} \tag{9}$$

where α is the significance level and $t_{\alpha/2,df_e}$ is the $100(1-\alpha/2)^{\text{th}}$ percentile of the t distribution with df_e degrees of freedom.

2.6.2 Tests of model effects

For each selected predictor series for $\boldsymbol{y}$, there are L lagged columns associated with it. The columns can be grouped together, considered as an effect, and tested with a null hypothesis of zero for all coefficients. This is similar to the test of a categorical effect with all dummy variables in a (generalized) linear model setting. Only type III tests are conducted here. For each selected predictor series $\boldsymbol{X}_{G,i}$, the type III test matrix $\boldsymbol{L}_i$ is constructed and $H_0 : \boldsymbol{L}_i\boldsymbol{\beta} = \mathbf{0}$ is tested based on an F-statistic.

- **F-statistics for effects**

$$F_i = \frac{\hat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{L}_i^{\mathrm{T}}\left(\boldsymbol{L}_i\boldsymbol{\Sigma}\boldsymbol{L}_i^{\mathrm{T}}\right)^{-1}\boldsymbol{L}_i\hat{\boldsymbol{\beta}}}{r_i} \tag{10}$$

where $r_i = rank\left(\boldsymbol{L}_i\boldsymbol{\Sigma}\boldsymbol{L}_i^{\mathrm{T}}\right)$. The statistic follows an approximate F distribution with the numerator degrees of freedom r_i and the denominator degrees of freedom df_e. Then the p-value is computed as follows:

$$p_{F_i} = 1 - prob\left(F_{r_i,df_e} \leq |F_i|\right) \tag{11}$$

2.6.3 Model quality measures

In addition to statistical inferences, the goodness of the model can be evaluated. The following model quality measures are provided:

- **Root Mean Squared Error (RMSE)**

$$RMSE = \sqrt{MSE} = \sqrt{\frac{SS_e}{df_e}} \tag{12}$$

Note that $RMSE = \hat{\sigma}$.

- **Root Mean Squared Percentage Error (RMSPE)**

$$RMSPE = \sqrt{MSPE} = \sqrt{\frac{\sum_{t=L+1}^{m}\left(\frac{y_t - \hat{y}_t}{y_t}\right)^2}{(m-L)}} \tag{13}$$

- **R squared**

$$R^2 = 1 - \frac{\sum_{t=1}^{t=(m-L)}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{t=(m-L)}(y_t - \bar{y})^2} = 1 - \frac{SS_e}{SS_t} \qquad (14)$$

- **Bayesian Information Criterion (BIC)**

$$BIC = (m-L)ln\left(\frac{SS_e}{(m-L)}\right) + \left((p^c+1)ln(m-L)\right) \qquad (15)$$

- **Akaike Information Criterion (AIC)**

$$AIC = (m-L)ln\left(\frac{SS_e}{(m-L)}\right) + 2(p^c+1) \qquad (15’)$$

3. Scoring

Once the models $(\hat{\boldsymbol{\beta}}, J_{sel})$ for all the required targets $(\boldsymbol{y})$ are built and post-estimation statistics are computed, the next task is to use these models to do scoring. There are two types of scoring: (1) fit: in-sample prediction for the past and current values of the target series; (2) forecast: out-of-sample prediction for future values of the target series.

3.1 Fit

Without loss of generality, we assume $\boldsymbol{X}$ and $\boldsymbol{X}_G$ are the selected predictor series matrices without lagged terms and with lagged terms, respectively; and $\hat{\boldsymbol{\beta}}$ is the coefficient estimates vector for the target $\boldsymbol{y}$, so $\boldsymbol{X} = [\boldsymbol{X}_1, \dots, \boldsymbol{X}_K]$, $\boldsymbol{X}_G = \left[\boldsymbol{1}, \boldsymbol{X}_{G_{11}}, \dots, \boldsymbol{X}_{G_{1L}}, \cdots, \boldsymbol{X}_{G_{K1}}, \dots, \boldsymbol{X}_{G_{KL}}\right]$ and $\hat{\boldsymbol{\beta}} = \left[\hat{\beta}_0, \hat{\beta}_{11}, \dots, \hat{\beta}_{1L}, \cdots, \hat{\beta}_{K1}, \dots, \hat{\beta}_{KL}\right]^{\mathrm{T}}$. Given that all series have m time points, in-sample prediction of $\boldsymbol{y}$ is one-step ahead prediction and can be written as

$$\hat{y}_t = \boldsymbol{X}_{G,t}\hat{\boldsymbol{\beta}} = \hat{\beta}_0 + \sum_{j \in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}_{j,\ell} \cdot X_{G_{j\ell},t} \qquad (16)$$

$$= \hat{\beta}_0 + \sum_{j \in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}_{j,\ell} \cdot X_{j,t-\ell}; \quad t = L+1, \dots, m. \qquad (17)$$

The corresponding $100(1-\alpha)\%$ confidence interval of $\boldsymbol{y}$ is

$$\left[\hat{y}_t - t_{\alpha/2,df_e} \times \hat{\sigma},\ \hat{y}_t - t_{\alpha/2,df_e} \times \hat{\sigma}\right];\ \ t = L+1, \dots, m. \qquad (18)$$

3.2 Forecast

Given that data is available up to time interval m, the one-step ahead forecast for $\boldsymbol{y}$ is

$$\hat{y}_m(1) = \hat{\beta}_0 + \sum_{j \in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}_{j,\ell} \cdot X_{j,m+1-\ell} \qquad (19)$$

The h-step ahead forecast for $\boldsymbol{y}$ is

$$\hat{y}_m(h) = \hat{\beta}_0 + \sum_{j \in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}_{j,\ell} \cdot \hat{X}_{j,m+h-\ell} \qquad (20)$$

where

$$\hat{X}_{j,m+h-\ell} = \begin{cases} X_{j,m+h-\ell}, & h \le \ell \\ \hat{X}_{j,m}(h-\ell), & h > \ell \end{cases}$$

Thus, forecasting the value of y_{m+2} requires us to first forecast the values of all the predictors up to time $(m+1)$. Forecasting the values of all the predictors up to time $(m+1)$ requires us to use Equation (19) on all the predictors $j \in J_{sel}$. Similarly, to predict the value of y_{m+3}, we need to forecast the values of predictors $j \in J_{sel}$ at time $(m+2)$ by using Equation (20). This task poses a bigger problem; to forecast the values of $j \in J_{sel}$ at time $(m+2)$, we first need to forecast the values of the predictors of $j \in J_{sel}$ at time $(m+1)$. That is, as we increasingly look into the future, we need to forecast more and more values to determine the value of y_{m+h}.

3.3 Approximated forecasting variances and intervals

In this subsection, we outline how forecasting variances and intervals can be computed for TCM models. We start by using the following representation for the linear model built by TCM for target y_{m+h}:

$$y_{m+h} = \hat{\beta}_0 + \sum_{j\in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}_{j,\ell} \cdot X_{j,m+h-\ell} + \varepsilon_{m+h} \tag{21}$$

where $\varepsilon_{m+h} \sim N(0,\sigma^2)$ and σ^2 is estimated as $\hat{\sigma}^2$ (computed in Section 2.6.1). Please note that we don't include parameter estimation error when defining forecasting error in TCM.

The forecasting error at $m+1$ is defined as the difference between y_{m+1} and $\hat{y}_m(1)$, which can be written as

$$e_{y,m}(1) = y_{m+1} - \hat{y}_m(1) = \varepsilon_{m+1} \tag{22}$$

The forecasting variance for one-step ahead forecasts is computed as $\hat{\sigma}^2$. For multi-step ahead forecasts, the forecasting error at $m+h$ is

$$e_{y,m}(h) = y_{m+h} - \hat{y}_m(h) = \sum_{j\in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}_{j,\ell} \cdot e_{X_j,m}(h-\ell) + \varepsilon_{m+h} \tag{23}$$

where $e_{X_j,m}(h-\ell) = X_{j,m+h-\ell} - \hat{X}_{j,m}(h-\ell)$ and $e_{X_j,m}(h-\ell) = 0$ if $h \le \ell$.

In general, $e_{X_j,m}(1), \cdots, e_{X_j,m}(h-\ell)$ are not independent of each other. The larger the h is, the more complex the dependence is. In addition, $e_{X_j,m}(h-\ell)$ and $e_{X_i,m}(h-\ell)$ might not be independent for $j, i \in J_{sel}$. In order to fully consider the dependence, we need to write all time series in vector autoregressive (VAR) format. Since we assume the number of series n is usually large, the parameter matrix, which is an $n \times n$ matrix, might be too large to handle in computation of the forecasting variances. Therefore, we make the assumption that all forecasting error terms in Equation (23), $e_{X_j,m}(h-\ell), j \in J_{sel}, \ell = 1, \dots, L$, are independent, so it is easier to compute the forecasting variances.

Based on the above independence assumption, the approximated variance of the forecasting error, $e_{y,m}(h)$, is

$$\hat{\sigma}^2_{e_{y,m,h}} = \sum_{j\in J_{sel}} \sum_{\ell=1}^{L} \hat{\beta}^2_{j,\ell} \hat{\sigma}^2_{e_{X_j,m,h-\ell}} + \hat{\sigma}^2 \tag{24}$$

where $\hat{\sigma}^2_{e_{X_{j,m,h-\ell}}}$ is the variance of the forecasting error in the series X_j at $m+h-\ell$.

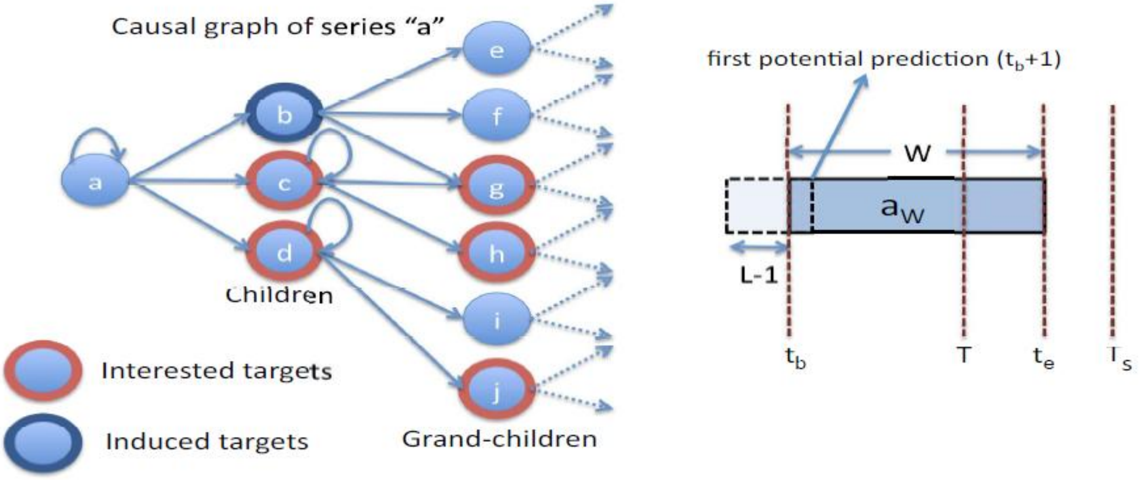
Then the corresponding $100(1-\alpha)\%$ approximated forecasting interval of y_{m+h} can be expressed as

$$\left[\hat{y}_m(h) - t_{\alpha/2,df_e} \times \hat{\sigma}_{e_{y,m,h}}, \hat{y}_m(h) + t_{\alpha/2,df_e} \times \hat{\sigma}_{e_{y,m,h}}\right] \tag{25}$$

4. Scenario analysis

Scenario analysis refers to a capability of TCM to "play-out" the repercussions of artificially setting the value of a time series. A scenario is the set of forecasts that are generated by substituting the values of a *root* time series by a vector of substitute values, as illustrated in Figure 1.

Figure 1: Causal graph of a root time series and the specification of the vector of substitute values

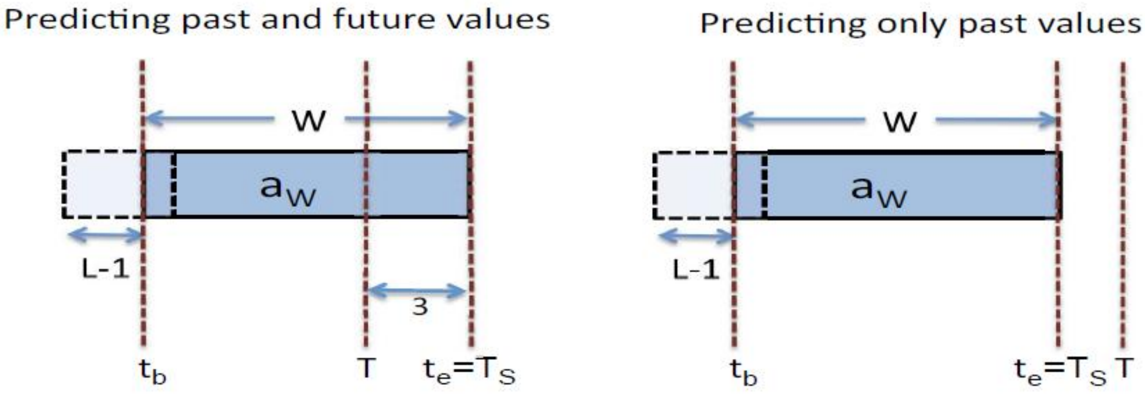


During scenario analysis, we specify the targets that we want to analyze as a response to changes in the values of the root series ("a" in Figure 1), along with the time window. In Figure 1, we are interested in the behavior of time series "c", "d", "g", "h", and "j" *only*. The rest of the time series are ignored. The figure also depicts the vector $\mathbf{a}_W$ of values for "a" that should be used instead of the observed or predicted values of "a". The values $(\boldsymbol{t_b}, \boldsymbol{t_e}, \boldsymbol{T}, \boldsymbol{T_s})$ specify the beginning and end of the replacement values for the root series, the current time, and the farthest time for analysis, respectively.

The partial Granger causal graph of time series "a" is shown in Figure 1. That is, "a" is the parent of itself, "b", "c", and "d". Similarly, it is the grand-parent of "e", "f", "g", "h", "i", and "j". Further descendents are possible, but only two generations suffice for the sake of explanation. Figure 1 also displays the specification of the vector $\mathbf{a}_W$, of length W, that contains the replacement values of the root series. In the example shown in the figure, $\mathbf{a}_W$ starts at time $t_b < T$, where T is the current time, and ends at $t_e > T$, which is in the future. We are also given T_s, the last time point $(t_e \leq T_s)$ for which we want to perform scenario analysis on the target variables. Finally, we are given a set of time series for which the scenario predictions are carried out. In the figure, these are "c", "d", "g", "h", and "j", which are marked with a thick red border. Since "b" is required to model "g", "b" is marked with a thick blue border to signify that it is an induced target. Given this information, the goal of scenario analysis is to forecast the values of the target time series (“c”, “d”, “g”, “h”, and “j”) up to time T_s, based on the values of the root time series $\mathbf{a}_W$.

Notice that we have to predict values of targets up to time T_s, where T_s can be $> (T+1)$ or $\leq (T+1)$. When $T_s = (T+2)$, we need to compute the values of the predictors of the target time series at time $(T+1)$. Similarly, when $T_s = (T+3)$, we need to compute the values of the predictors’ predictors at time $(T+1)$ and the values of the predictors at time $(T+2)$ before predicting the values of the target time series at time $(T+3)$.

Figure 2: Scenarios with and without predicting future values


Predicting past and future values
W
a_W
L-1
3
t_b
T
t_e=T_S
Predicting only past values
W
a_W
L-1
t_b
t_e=T_S T

The left-hand panel in Figure 2 depicts a scenario where the values of ancestors of targets of interest also have to be predicted. In this particular case, $\boldsymbol{T_s} = (\boldsymbol{T}+\mathbf{3})$ and therefore it is necessary to predict the values of the predictors of the targets at $(\boldsymbol{T}+\mathbf{1})$ and $(\boldsymbol{T}+\mathbf{2})$, and values of the predictors’ predictors at time $(\boldsymbol{T}+\mathbf{1})$. The right-hand panel depicts a scenario where the entire period of prediction is earlier than the current time $\boldsymbol{T}$ (i.e., $\boldsymbol{T_s} < \boldsymbol{T}$). In this case, all the values of the predictors and their ancestors are readily available.

Determining $\mathbf{a}_W$

In the discussion above, we have neglected the issue of $\mathbf{a}_W$, the substitute values for time series “a”, which is the root time series. For purposes of scenario analysis, it is sufficient to consider that $\mathbf{a}_W$ is readily available. In a typical use case for scenario analysis, $\mathbf{a}_W$ will come from the values specified by the user’s direct input, although its values could also come as input from a calling meta-process (as is the case with the use of scenario analysis as a sub-procedure in root cause analysis, as shown in Section 6).

Caveat on scenario analysis

It is possible to carry out scenario analysis for a time period that is entirely in the future; that is $t_b > T$. However, forecasting errors in the remaining predictors may make such scenario analysis inherently low-precision. That is, if $\theta = t_b - T$ and $t_b > T$, then the precision of scenario analysis decreases with an increase in θ.

4.1 *SA*, the scenario analysis algorithm

Input:

The inputs to *SA* are: (1) $\boldsymbol{r}$: the root time series; (2) $\boldsymbol{r}_W$: the vector of replacement values for time series $\boldsymbol{r}$; (3) (t_b, t_e, T, T_s): the beginning and end time for the modified values of $\boldsymbol{r}$, the current time, and the last time point for which target values need to be predicted, respectively; (4) D: a set of descendant target time series of interest along with their relation to $\boldsymbol{r}$ (which may be input as the Granger causal graph, G). Notice that the length of $\boldsymbol{r}_W$ is $t_e - t_b + 1$ and $t_e \leq T_s$. Furthermore, it is erroneous to have a target $\boldsymbol{d} \in D$, where $\boldsymbol{r}$ is not an ancestor of $\boldsymbol{d}$.

Output:

For each $\boldsymbol{d}$ in D, we output a vector $\boldsymbol{d}_{sa}$ containing values that pertain to the scenario analysis of these time series and the corresponding confidence intervals (when $T_s \leq T$) or apprxomiated forecasting intervals (when $T_s > T$). Please note that the time period for the children series in D is $[t_b + 1, T_s]$, for the grand-children series is $[t_b + 2, T_s]$, etc.

Preparation:

To prepare for *SA*, we first calculate the closure on the set of targets D^* that need to be predicted, which is determined by the relationship between $\boldsymbol{r}$ and each of the targets in D. Essentially, D^* is computed by iteratively looking at the path from each $\boldsymbol{d} \in D$ and adding all those intermediate nodes that are ancestors of $\boldsymbol{d}$ and are also descendents of $\boldsymbol{r}$. In the example shown in Figure 1, the time series “b” is itself not of primary interest, but since it is a parent of “g”, which is of interest, “b” is also added as a target of interest to the set {“c”, “d”, “g”, “h”, “j”}.

Next, we compute M, the set of models that need to be included in order to perform scenario analysis on D^*. Obviously, M contains the models for each of the series in D^*, i.e., $D^* \subset M$; however, depending on the time span of the scenario analysis, additional models of some time series might have to be brought in (see Figure 2). Basically, depending on how far ahead T_s is from T, we may need to compute the values of the ancestors (other than $\boldsymbol{r}$) of the targets of interest at time points $(T+1), \ldots, (T_s - 1)$. That is, the set $\{M - D^*\}$ (which may be $\emptyset$) contains all series that are needed for scenario analysis and are not descendants of $\boldsymbol{r}$.

At the end of the preparation phase we have D^* and M, which allows us to predict all the time series of interest.

Computation:

The computation in scenario analysis is exactly that of scoring the values of a set of time series (see Section 3). For each target in D^*, we have a range of time points for which we need to fit/forecast values. For example, for immediate children of the root (“c”, “d”, and the induced child “b” in Figure 1), this range is $[t_b + 1, T_s]$. Similarly, for grand-children (“g”, “h”, and “j” in Figure 1), this range is $[t_b + 2, T_s]$. Using the models in M and substituted values $\boldsymbol{r}_W$ for $\boldsymbol{r}$, this task can be carried out.

5. Outlier detection

One of the advantages of building TCM models is the ability to detect model-based outliers. Outliers can be defined in several ways. For now, we shall define an outlier in a time series to be a value that strays too far from its expected (fitted) value based on the TCM models. The detection process is based on the normal distribution assumption for series $\boldsymbol{y}$. Consider the value of a time series $\boldsymbol{y}$ at time t. Let y_t and $\hat{y}_t$ be the observed and expected values of $\boldsymbol{y}$ at time t, respectively; and $\hat{\sigma}^2$ be the variance of $\boldsymbol{y}$ from the TCM model (based on residuals). Given these inputs, we call y_t an outlier if the likelihood of y_t when modeled as a normal random variable with mean $\hat{y}_t$ and variance $\hat{\sigma}^2$ is below a particular threshold.

Input:

The inputs to OD (outlier detection) are: (1) $y_t, \forall\, t$; (2) $\hat{y}_t, \forall\, t$; (3) $\hat{\sigma}^2$; (4) the outlier threshold value $\kappa \in (0,1]$ (the default is 0.95).

Computation:

a) Under the assumption that the observed value $y_t$is a normal random variable with mean $\hat{y}_t$ and variance $\hat{\sigma}^2$, compute the square score at time t as

$$s_{sqr,t} = \frac{(y_t - \hat{y}_t)^2}{\hat{\sigma}^2} \qquad (26)$$

b) Compute the outlier probability as

$$p_{sqr,t} = \ prob\left(\chi_1^2 \leq s_{sqr,t}\right) \qquad (27)$$

where χ_1^2 is a random variable with a chi-squared distribution with 1 degree of freedom.

c) Flag y_t as an outlier if $p_{sqr,t} \geq \kappa$.

Output:

The output to OD for series $\boldsymbol{y}$ is a set of time points with their corresponding outlier probabilities.

6. Outlier root cause analysis

In Section 5, we saw how to detect outliers. The next logical step is to find the likely causes for a time series whose value has been flagged as an outlier. Outlier root cause analysis refers to the capability to explore the Granger causal graph in order to analyse the key/root values that resulted in the outlier under question. To formalize this notion, consider a time series $\boldsymbol{y}$, whose observed value at time t (that is, y_t) has been flagged as an outlier due to its abnormal deviation from its expected value $\hat{y}_t$. The goal of outlier root cause analysis (*ORCA*) is to output the set of time series $\mathcal{A}$ that can be considered as root causes of the anomalous value of y_t. The idea is that setting the values of time series in the predictor set $\boldsymbol{X}$ to their normal/expected values, instead of their observed values, will bring the outlying y_t back to normal. The normal value of y_t is unknown so we specify it with the expected value of $\boldsymbol{y}$ at time t as predicted by $\boldsymbol{y}$'s univariate model, which is an AR(L) model, and denoted as $\tilde{y}_t$.

The result of *ORCA* has the following objective function with a constraint as follows:

$$\arg\max_{x \in \mathcal{A}_y} |\hat{y}_t - \tilde{y}_t| - |\hat{y}_{t|x=\hat{x}} - \tilde{y}_t| \qquad (28)$$

$$\text{s.t.}\, |\hat{y}_t - \tilde{y}_t| \geq |\hat{y}_{t|x=\hat{x}} - \tilde{y}_t|$$

where $\mathcal{A}_y$ corresponds to the set of ancestors of $\boldsymbol{y}$ according to the Granger causal graph G. The quantity $\hat{y}_{t|x=\hat{x}}$ should be interpreted as the likely predicted value of y at time t had the value of its ancestor x been set to its expected value of $\hat{x}$. We see that Equation (28) is made up of two parts: (1) the portion $|\hat{y}_t - \tilde{y}_t|$, which is the degree of "outlier-ness" of y at t as predicted by the "Granger model", where the outlier-ness is judged based on what is expected from the history of $\boldsymbol{y}$; (2) the portion $|\hat{y}_{t|x=\hat{x}} - \tilde{y}_t|$, which is the degree of "outlier-ness" of y at t as predicted by the "Granger model", if x was corrected. In other words, Equation (28) amounts to replacing the observed value y_t by its "expected" value, given by a simpler, univariate model. Therefore Equation (28) expresses the reduction in the degree of outlier-ness in y_t brought about by correcting x.

6.1 *ORCA*, the outlier root cause analysis algorithm

Input:

The inputs to *ORCA* are: (1) $\boldsymbol{y}$, the anomalous time series; (2) t, the time at which the anomaly was detected; (3) y_t, the anomalous value; (4) $\hat{y}_t$, the expected value of y_t; (5) k, the oldest generation of ancestors to search based on the Granger causal graph, G.

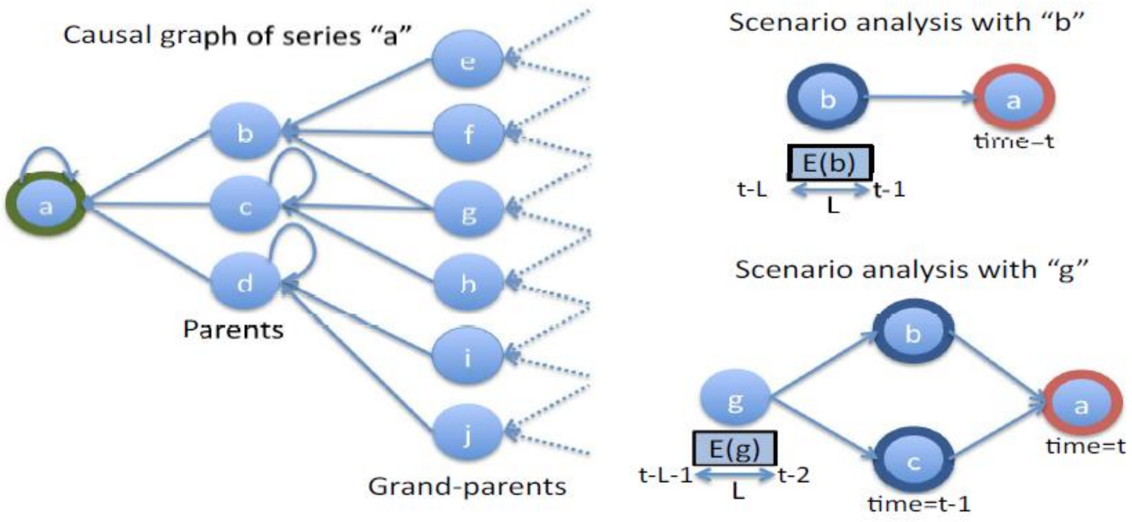
Output:

ORCA outputs the set of root causes $\mathcal{A}$ of the anomaly in y_t, where each $\boldsymbol{x} \in \mathcal{A}$ maximizes the objective function in Equation (28) by the same amount.

Preparation:

To prepare for *ORCA*, we first compute $\mathcal{A}_y$, the set of ancestors that need to be examined as the potential root causes of the anomaly in y_t.

Figure 3: Outlier root cause analysis for a time series



In the example shown in Figure 3, assuming that $\boldsymbol{y}$ ="a" and $k = 2$, then $\mathcal{A}_y$ = { "b", "c", "d", "e", "f", "g", "h", "i", "j"}. $\mathcal{A}_y$ can be computed by performing a reverse breadth-first search from $\boldsymbol{y}$ to k levels.

Second, each potential root cause $\boldsymbol{x} \in \mathcal{A}_y$ is prepped for scenario analysis by computing the vector of substitute values of $\boldsymbol{x}$ to be used during scenario analysis. Note that the length of this substitute vector is L, the lag. For example, consider $\boldsymbol{b}_L$, the substitute for time series “b” in Figure 3. As “b” is a parent of “a”, we need to compute the fits of “b” from $(t - L)$ to $(t - 1)$. On the other hand, as “g” is a grand-parent of “a”, $\boldsymbol{g}_L$ contains the fits for “g” from the time $(t - L - 1)$ to $(t - 2)$ (see Section 3.1 for computation of fits). Please note that this approach assumes that any anomalies are purely in “b” (the parent series) or “g” (the grandparent series). In particular, it is assumed that anomalies in “b” are not caused by values in the grandparent series, including anomalous values in the grandparent series.

Third, for each potential root cause $\boldsymbol{x} \in \mathcal{A}_y$, scenario analysis is carried out (see Section 4) using the substitute values computed in the previous step. For the example in Figure 3, scenario analysis is called for series “b” with the parameters $(\boldsymbol{r} = \boldsymbol{b}, \boldsymbol{r}_W = \boldsymbol{b}_L, t_b = (t - L), t_e = (t - 1), T = t, D = \{\boldsymbol{a}\}, T_s = t)$. And the result of scenario analysis is $\hat{y}_{t|x=\hat{x}}$.

Computation:

The process of *ORCA* is as follows:

- Initiaize $\mathcal{A}$, the set of potential root causes for y_t, to $\emptyset$.
  Initialize obj_{max}, the maximum objective function value, to 0.
- Suppose there are J series in $\mathcal{A}_y$, $\boldsymbol{x}_1, \dots, \boldsymbol{x}_J$.
  For each $\boldsymbol{x}_j, j \in 1, \dots, J$, compute $obj_j = |\hat{y}_t - \tilde{y}_t| - |\hat{y}_{t|x_j=\hat{x}_j} - \tilde{y}_t|$.
  If $obj_j \geq obj_{max}$, set $obj_{max} = obj_j$ and store $\boldsymbol{x}_j$ in $\mathcal{A}$.

References

[1]. Arnold, A., Liu, Y., and Abe, N. (2007). Temporal causal modeling with graphical granger methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 66–75, New York, NY, USA. ACM.

[2]. Darema, F., George, D. A., Norton, V. A., and Pfister, G. F. (1988). A single-program-multiple-data computational model for EPEX/FORTRAN. *Parallel Computing*, 7(1):11–24.

[3]. Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. volume 51.

[4]. Duchi, J., Gould, S., and Koller, D. (2008). Projected subgradient methods for learning sparse gaussians. In *Proceedings of the Twenty-fourth Conference on Uncertainty in AI (UAI)*.

[5]. Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

[6]. Granger, C. W. J. (1980). Testing for causality : A personal viewpoint. *Journal of Economic Dynamics and Control*, 2(1):329–352.

[7]. Hsieh, C.-J., Sustik, M. A., Dhillon, I. S., and Ravikumar, P. (2011). Sparse inverse covariance matrix estimation using quadratic approximation. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 2330–2338. http://nips.cc/.

[8]. Kambadur, P. and Lozano, A. C. (2013). A parallel, block greedy method for sparse inverse covariance estimation for ultra-high dimensions. In *Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[9]. Li, L. and chuan Toh, K. (2010). An inexact interior point method for l1-regularized sparse covariance selection. Technical report, National University Of Singapore.

[10]. Lozano, A. C., Abe, N., Liu, Y., and Rosset, S. (2009). Grouped graphical granger modeling methods for temporal causal modeling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 577–586, New York, NY, USA. ACM.

[11]. Lozano, A. C., Swirszcz, G., and Abe, N. (2011). Group orthogonal matching pursuit for logistic regression. *Journal of Machine Learning Research - Proceedings Track*, 15:452–460.

[12]. MPI Forum (1995). Message Passing Interface. http://www.mpi-forum.org/.

[13]. MPI Forum (1997). Message Passing Interface-2. http://www.mpi-forum.org/.

[14]. O.Banerjee, El Ghaoui, L., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516.

[15]. Scheinberg, K., Ma, S., and Goldfarb, D. (2010). Sparse inverse covariance selection via alternating linearization methods. *CoRR*, abs/1011.0097.

[16]. Scheinberg, K. and Rish, I. (2010). Learning sparse gaussian markov networks using a greedy coordinate ascent approach. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 196–212, Berlin, Heidelberg. Springer-Verlag.

[17]. Strang, G. (1993). *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

Tree-AS (CHAID) Modeling Algorithms

1. Introduction

CHAID stands for Chi-squared Automatic Interaction Detector. It is a highly efficient statistical technique for segmentation, or tree growing, developed by (Kass, 1980). Using the significance of a statistical test and effect size as criteria, CHAID evaluates all of the values of a potential predictor. It merges values that are judged to be statistically homogeneous (similar) with respect to the target variable and maintains all other values that are heterogeneous (dissimilar). It then selects the best predictor to form the first branch in the decision tree, such that each child node is made of a group of homogeneous values of the selected predictor. This process continues recursively until the tree is fully grown.

Exhaustive CHAID is a modification of CHAID developed to address some of the weaknesses of the CHAID method (Biggs, de Ville, and Suen, 1991). In particular, sometimes CHAID may not find the optimal split for a variable, since it stops merging categories as soon as it finds that all remaining categories are statistically different. Exhaustive CHAID remedies this by continuing to merge categories of the predictor variable until only two super categories are left. It then examines the series of merges for the predictor and finds the set of categories that gives the strongest association with the target variable, and computes an adjusted *p*-value for that association. Thus, Exhaustive CHAID can find the best split for each predictor, and then choose which predictor to split on by comparing the adjusted *p*-values.

Although CHAID or Exhaustive CHAID is efficient for data mining, there could be performance issues. For example, the collection of summary statistics required for the tree growth will be expensive when the raw data is distributed and massive. Moreover, the decision of splitting rules will also be heavy when the number of predictors becomes very large since conducting category merge for each predictor is not trivial. In these regard, parallel calculation is necessary in order to improve the performance.

The document is concerned with CHAID and Exhaustive CHAID related algorithms. These algorithms will be implemented in parallel within Analytic Engine (AE), based on the map-reduce framework.

Notice that the document provides technical details for engineers to develop the Tree engine. For a more readable document, please refer to the algorithm document in Statistics or Modeler.

2. Notes

1. To prepare training data, invalid, system missing, and user missing values in predictors will be considered as a single missing category.
2. The Tree engine relies on the Descriptive engine through SmartModeler doing the following transformations:
    a. Zero inflation handling
    Zero inflated cases are imputed with missing values.
    b. Binning continuous variables
    The tiling method is used with δ (default 5) as the number of bins.
    c. Supervised category merging
    If the number of categories in a categorical variable is larger than δ (default 12), supervised category merging will be used.
    d. Feature selection
    If the number of predictors is larger than δ (default 500), feature selection will be applied.
    e. Trim trailing blanks
    f. Date/time handling
    Date/time variables are transformed into continuous ones with Jan 1st, 1970 as default reference data and 00:00:00 as default reference time.

3. Notations

The following notations are used throughout the document unless otherwise stated:

Y	Dependent, or target, variable. If Y is categorical with J categories, its category takes values in $C = \{1, \dots, J\}$
$X_m, m = 1, \dots, M$	Set of all predictor variables. If X_m is categorical with I_m categories, its category takes values in $D = \{1, \dots, I_m\}$
$\hbar = \{x_{m,n}, y_n\}_{n=1}^{N}$	Whole training cases
$\hbar(t)$	Training cases that fall in node t
w_n	Case weight associated with case n
f_n	Frequency weight associated with case n. Non-integral positive value is rounded to its nearest integer

$C(i\|j)$	Cost of miss-classifying a category j case as a category i case, $C(j\|j)$
$\mathrm{I}(a=b)$	Indicator function taking value 1 when $a=b$, 0 otherwise

4. Binning Continuous Predictors

CHAID and Exhaustive CHAID algorithms only accept nominal or ordinal categorical predictors. When predictors are continuous, they are transformed into ordinal predictors before tree growth.

For a given set of break points $a_1, a_2, \dots, a_{I-1}$ (in ascending order), a given x is mapped into category $C(x)$ as follows:

$$C(x) = \begin{cases} 1 & x \le a_1 \\ i+1 & a_i < x \le a_{i+1}, i = 1, \dots, I-2 \\ I & a_{I-1} < x \end{cases}$$

For binning continuous predictors, we use the tiling method which has been implemented by the Descriptive engine. We use 5 as the default number of bins. For algorithm details, please refer to Ref. 3.

The choice of the tiling method is based on some experimental results. Please refer to the document of 'Comparison of binning methods' in Ref. 6.

5. CHAID Algorithm

CHAID tree grows level-by-level from the root node. The general procedure is as follows:

1. Create the root node, and mark it as the initial non-terminal leaf node.
2. Repeat the following steps until no non-terminal leaf nodes exist in the current tree:
   a) Pass the training data, and collect summary statistics for each predictor and non-terminal leaf node.
   b) Merging – For each predictor and non-terminal leaf node, merge predictor categories.
   c) Splitting – For each non-terminal leaf node, select the best predictor to be used to best split the node. If the best predictor is valid for splitting, split the node using this predictor. Else, mark it as a terminal leaf node.
   d) Stopping – For each node that was split in step c), check the child nodes to see which nodes should be marked as terminal leaf nodes.

In the following, we will describe how each step in tree growth can be accomplished.

5.1. Creating Root Node

To grow a tree, the root node should be created in the first step.

CreateRootNode()

```
Inputs:
   –  $N_f$                      // Count of valid training cases
   –  $I_m$, $m = 1, \ldots, M$           // Number of categories of predictor $X_m$
   <Continuous target>
   –  $V_f(Y)$                   // Variance of target variable
   <Categorical target>
   –  $J$                        // Number of target categories

   <Parameter settings>
   –  MinParentCasesABS      // Default 100
   –  NodeSizeRequirement    // {‘absolute’, ‘percentage’}, default ‘absolute’
Outputs:
   –  $T(0)$                     // Initial tree
   –  $t$                        // Root node
Procedure:
1. If ($N_f = 0$),
   or(NodeSizeRequirement=’absolute’ and $N_f$ < MinParentCasesABS),
   or((target is continuous)and($V_f(Y) = 0$)),
   or((target is categorical)and($J = 1$)),
   or($I_m = 1, m = 1, \ldots, M$),
      Return a null tree;
2. Else,{
      Create root node $t$;
      Create tree $T(0)$ which has only the root node;
   }
```

5.2. Collecting Summary Statistics

Summary statistics are collected for each predictor and non-terminal leaf node.

According to the type of target variable, we compute different sets of summary statistics. If the target variable is categorical, summary statistics for a non-terminal leaf node t, predictor X_m (with I_m categories), and categorical target Y (with J categories), consist of the following statistics:

Cell frequency: $n_{i,j}^{<m,t>} = \sum_{n \in \hbar(t)} f_n \mathrm{I}(x_{m,n} = i \cap y_n = j)$,

Cell weighted frequency: $w_{i,j}^{<m,t>} = \sum_{n \in \hbar(t)} w_n f_n \mathrm{I}(x_{m,n} = i \cap y_n = j)$, where $i = 1, \ldots, I_m, j = 1, \ldots, J$.

If the target variable is continuous, summary statistics for a non-terminal leaf node t, predictor X_m (with I_m categories), and continuous target Y consist of the following statistics:

Count: $N_{f,i}^{<m,t>} = \sum_{n \in \hbar(t)} f_n \mathrm{I}(x_{m,n} = i)$,

Mean: $\bar{y}_{f,i}^{<m,t>} = \frac{\sum_{n \in \hbar(t)} f_n y_n \mathrm{I}(x_{m,n}=i)}{N_{f,i}^{<m,t>}}$,

Variance: $V_{f,i}^{<m,t>} = \frac{1}{N_{f,i}^{<m,t>}} \sum_{n \in \hbar(t)} f_n \mathrm{I}(x_{m,n} = i)(y_n - \bar{y}_{f,i}^{<m,t>})^2$,

Weighted count: $N_{w,i}^{<m,t>} = \sum_{n \in \hbar(t)} w_n f_n \mathrm{I}(x_{m,n} = i)$,

Weighted mean: $\bar{y}_{w,i}^{<m,t>} = \frac{\sum_{n \in \hbar(t)} w_n f_n y_n \mathrm{I}(x_{m,n}=i)}{N_{w,i}^{<m,t>}}$,

Weighted variance: $V_{w,i}^{<m,t>} = \frac{1}{N_{w,i}^{<m,t>}} \sum_{n \in \hbar(t)} w_n f_n \mathrm{I}(x_{m,n} = i)(y_n - \bar{y}_{w,i}^{<m,t>})^2$,

where $i = 1, \ldots, I_m$.

5.3. Merging

Based on summary statistics, non-significant categories are merged for each predictor and non-terminal leaf node.

CHAID_Merging()

```
Inputs:
   // Global summary statistics for predictor $X_m$ and node $t$
   <Continuous target>
   –  $N_{f,i}^{<m,t>}$
   –  $\bar{y}_{f,i}^{<m,t>}$
   –  $V_{f,i}^{<m,t>}$
   –  $N_{w,i}^{<m,t>}$
   –  $\bar{y}_{w,i}^{<m,t>}$
   –  $V_{w,i}^{<m,t>}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>}$
   –  $w_{i,j}^{<m,t>}$
   where $i = 1, \dots, I_m$ and $j = 1, \dots, J$

   <Parameter settings>
   –  TreeGrowingMethod       // {‘p-value’, ‘effectsize’}
   –  AlphaMerge              // Default 0.05
   –  AlphaSplitMerge         // Default 0.025
   –  EffectSizeThreshold
   –  BonferroniAdjustment    // {true, false}, default true
   –  ChiSquareType           // {‘pearson’, ‘likelihood’}, default ‘pearson’
   –  Epsilon                 // Default 0.001
   –  MaxIterations           // Default 100
   –  MinChildCasesABS        // Default 50
   –  MinChildCasesPct        // Default 1
   –  NodeSizeRequirement     // {‘absolute’, ‘percentage’}, default ‘absolute’
   –  Scores                  // Vector value, scores for categories of $Y$
   –  SplitMergedCategories   // {true, false}, default false
Outputs:
   –  $\Theta^{<m,t>}$                  // Set of merged categories
   –  $p_{value}^{<m,t>}$                 // P-value, computed for $\Theta^{<m,t>}$
   –  TestStatistic           // Test statistic associated with $p_{value}^{<m,t>}$
   –  FreedomDegrees          // Freedom degrees associated with $p_{value}^{<m,t>}$
   –  $E_s^{<m,t>}$                  // Effect size
   <Continuous target>
   –  $N_{f,i}^{<m,t>}$
   –  $\bar{y}_{f,i}^{<m,t>}$
   –  $V_{f,i}^{<m,t>}$
   –  $N_{w,i}^{<m,t>}$
   –  $\bar{y}_{w,i}^{<m,t>}$
   –  $V_{w,i}^{<m,t>}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>}$
   –  $w_{i,j}^{<m,t>}$
   where $i \in \Theta^{<m,t>}$, $j = 1, \dots, J$
Procedure:
1. If (target is continuous),
      $\Theta^{<m,t>} = \{i | N_{w,i}^{<m,t>} > 0, i = 1, \dots, I_m\}$;
   If (target is categorical),
      $\Theta^{<m,t>} = \{i | \sum_{j=1}^{J} n_{ij}^{<m,t>} > 0, i = 1, \dots, I_m\}$;
   // Notice that if the predictor is ordinal, $\Theta^{<m,t>}$ will not include the
   missing category initially.
2. If ($|\Theta^{<m,t>}| \leq 1$),
      Go to step 6;
3. If (TreeGrowingMethod=‘p-value’),{
      If (predictor is nominal),{
         $pt = -1$;
         For $\forall i \in \Theta^{<m,t>}$,{
            For $\forall j \in \Theta^{<m,t>}$ and $j > i$,{
               Compute p-value $p$ and effect size for category $i$ and $j$;
               If ($p > pt$),{
                  $pt = p$;
                  $cat\_i = i$;
                  $cat\_j = j$;
               }
               Else if $p = pt$, resolve tied maximum p-values;
            }
         }
      }
      If (predictor is ordinal),{
         $pt = -1$;
         For $\forall i \in \Theta^{<m,t>}$,{
            Get category $j$ in $\Theta^{<m,t>}$ which is subsequent to $i$, if exists;
            Compute p-value $p$ and effect size for category $i$ and $j$;
            If ($p > pt$),{
               $pt = p$;
               $cat\_i = i$;
               $cat\_j = j$;
            }
            Else if $p = pt$, resolve tied maximum p-values;
         }
      }
      If ($pt >$ AlphaMerge),{
         Merge $cat\_i$ and $cat\_j$ into a compound category $c$.
         Compute summary statistics for the compound category $c$;
         Update $\Theta^{<m,t>}$;
      }
      Else,
         Go to step 6;
   }
   If (TreeGrowingMethod=‘effectsize’),{
      If (predictor is nominal),{
         $et = 100$;
         For $\forall i \in \Theta^{<m,t>}$,{
            For $\forall j \in \Theta^{<m,t>}$ and $j > i$,{
               Compute effect size $es$ for category $i$ and $j$;
               If ($es < et$),{
                  $et = es$;
                  $cat\_i = i$;
                  $cat\_j = j$;
               }
            }
         }
      }
      If (predictor is ordinal),{
         $et = 100$;
         For $\forall i \in \Theta^{<m,t>}$,{
            Get category $j$ in $\Theta^{<m,t>}$ which is subsequent to $i$, if exists;
            Compute effect size $es$ for category $i$ and $j$;
            If ($es < et$),{
               $et = es$;
               $cat\_i = i$;
               $cat\_j = j$;
            }
         }
      }
      If ($et <$ EffectSizeThreshold),{
         Merge $cat\_i$ and $cat\_j$ into a compound category $c$.
         Compute summary statistics for the compound category $c$;
         Update $\Theta^{<m,t>}$;
      }
      Else,
         Go to step 6;
   }
4. Let A be the set of original categories in the new category $c$;
   If (TreeGrowingMethod='p-value'),
   and(SplitMergedCategories=true),
   and($3 \leq |\mathrm{A}| \leq 15$),{
      If (predictor is nominal),{
         $pt = 2$;   // Any value larger than 1 should be ok
         For ($k = 1{:}\lfloor |\mathrm{A}|/2 \rfloor$),{
            For ($\forall A1$ with $k$ categories belonging to A),{
               Let $A2 = A - A1$;
               Compute p-value $p$ and effect size for category $cat\_A1$ and $cat\_A2$;
               // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
               If ($p < pt$),{
                  $pt = p$;
                  $cat\_i = cat\_A1$;
                  $cat\_j = cat\_A2$;
               }
               Else if $p = pt$, resolve tied minimum p-values;
            }
         }
      }
      If (predictor is ordinal),{   // Set A consists of ordered categories
         $pt = 2$;
         Let $A1$ be the set consisting of the first category in $A$;
         Let $A2 = A - A1$;
         While ($A2$ is not empty),{
            Compute p-value $p$ and effect size for category $cat\_A1$ and $cat\_A2$;
            // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
            If ($p < pt$),{
               $pt = p$;
               $cat\_i = cat\_A1$;
               $cat\_j = cat\_A2$;
            }
            Else if $p = pt$, resolve tied minimum p-values;
            Move the first category in $A2$ into $A1$;
         }
      }
      If ($pt \leq$ AlphaSplitMerge),{
         Split category $c$ into two categories $cat\_i$ and $cat\_j$;
         Compute summary statistics for categories $cat\_i$ and $cat\_j$;
         Update $\Theta^{<m,t>}$;
      }
   }
   If (TreeGrowingMethod='effectsize'),
   and(SplitMergedCategories=true),
   and($3 \le |A| \le 15$),{
      If (predictor is nominal),{
         $et = -1$;
         For ($k = 1{:}\lfloor |A|/2 \rfloor$),{
            For ($\forall A1$ with $k$ categories belonging to A),{
               Let $A2 = A - A1$;
               Compute effect size $es$ for category $cat\_A1$ and $cat\_A2$;
               // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
               If ($es > et$),{
                  $et = es$;
                  $cat\_i = cat\_A1$;
                  $cat\_j = cat\_A2$;
               }
            }
         }
      }
      If (predictor is ordinal),{
         $et = -1$;
         Let $A1$ be the set consisting of the first category in $A$;
         Let $A2 = A - A1$;
         While ($A2$ is not empty),{
            Compute effect size $es$ for category $cat\_A1$ and $cat\_A2$;
            // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
            If ($es > et$),{
               $et = es$;
               $cat\_i = cat\_A1$;
               $cat\_j = cat\_A2$;
            }
            Move the first category in $A2$ into $A1$;
         }
      }
      If ($et \ge$ EffectSizeThreshold),{
         Split category $c$ into two categories $cat\_i$ and $cat\_j$;
         Compute summary statistics for categories $cat\_i$ and $cat\_j$;
         Update $\Theta^{<m,t>}$;
      }
   }
5. Go to step 2;
6. If (TreeGrowingMethod=‘p-value’),
   and(predictor is ordinal),
   and($N_{f,cat\_missing}^{<m,t>} > 0$),{
   // We use subscript ‘cat_missing’ to denote summary statistics for the
   // missing category. Notice that this is only for ordinal predictors and we
   // do not distinguish the missing category with other categories for nominal
   // predictor.
      $\Theta_1^{<m,t>} = \Theta^{<m,t>} \cup$ {cat_missing};
      Compute p-value $p_1$ and effect size $e_1$ for the set of merged categories $\Theta_1^{<m,t>}$;
      $pt = -1$;
      $cat\_i$ =cat_missing;
      For $\forall j \in \Theta^{<m,t>}$,{
         Compute p-value $p$ and effect size for category $j$ and cat_missing;
         If ($p > pt$),{
            $pt = p$;
```

```
               $cat\_j = j$;
           }
           Else if $p = pt$, resolve tied maximum p-values;
       }
       Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
       Compute summary statistics for the compound category $c$;
       Let $\Theta_2^{<m,t>}$ be the new set of categories;
       Compute p-value $p_2$ and effect size $e_2$ for $\Theta_2^{<m,t>}$;
       If $(p_1 \neq p_2)$,{
           $p_{value}^{<m,t>} = (p_1 < p_2)? p_1 : p_2$;
           $\Theta^{<m,t>} = (p_1 < p_2)? \Theta_1^{<m,t>} : \Theta_2^{<m,t>}$;
           $E_s^{<m,t>} = (p_1 < p_2)? e_1 : e_2$;
       }
       Else, resolve tied minimum p-values;
   }
   If (TreeGrowingMethod=‘effectsize’),
   and(predictor is ordinal),
   and($N_{f,cat\_missing}^{<m,t>} > 0$),{
       $et = 100$;
       $cat\_i$ =cat_missing;
       For $\forall j \in \Theta^{<m,t>}$,{
           Compute p-value and effect size $es$ for category $j$ and cat_missing;
           If $(es < et)$,{
               $et = es$;
               $cat\_j = j$;
           }
       }
       Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
       Compute summary statistics for the compound category $c$;
       Let $\Theta_2^{<m,t>}$ be the new set of categories;
       Compute p-value $p_2$ and effect size $e_2$ for $\Theta_2^{<m,t>}$;
       $p_{value}^{<m,t>} = (e_1 > e_2)? p_1 : p_2$;
       $\Theta^{<m,t>} = (e_1 > e_2)? \Theta_1^{<m,t>} : \Theta_2^{<m,t>}$;
       $E_s^{<m,t>} = (e_1 > e_2)? e_1 : e_2$;
   }
7. If (TreeGrowingMethod=‘p-value’),{
       While $\exists i \in \Theta^{<m,t>}$,((NodeSizeRequirement=’absolute’)and(((target is
       continuous)and($N_{f,i}^{<m,t>}$ < MinChildCasesABS))or((target is
       categorical)and($\sum_j n_{i,j}^{<m,t>}$ < MinChildCasesABS)))),
       or((NodeSizeRequirement=’percentage’)and(((target is
       continuous)and($N_{f,i}^{<m,t>}$ < MinChildCasesPct*$N_f$))or((target is
       categorical)and($\sum_j n_{i,j}^{<m,t>}$ < MinChildCasesPct*$N_f$)))),{
           If (predictor is nominal),{
               $pt = -1$;
               $cat\_i = i$;
               For $\forall j \in \Theta^{<m,t>}$ and $j \neq i$,{
                   Compute p-value $p$ and effect size for category $i$ and $j$;
                   If $(p > pt)$,{
                       $pt = p$;
                       $cat\_j = j$;
                   }
```

```
            Else if $p = pt$, resolve tied maximum p-values;
         }
         Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
         Compute summary statistics for the compound category $c$;
         Update $\Theta^{<m,t>}$;
      }
      If (predictor is ordinal),{
         $pt = -1$;
         $cat\_i = i$;
         Get category $j$ in $\Theta^{<m,t>}$ which is antecedent to $i$, if exists;
         Compute p-value $p$ and effect size for category $i$ and $j$;
         If ($p > pt$),{
            $pt = p$;
            $cat\_j = j$;
         }
         Else if $p = pt$, resolve tied maximum p-values;
         Get category $j$ in $\Theta^{<m,t>}$ which is subsequent to $i$, if exists;
         Compute p-value $p$ and effect size for category $i$ and $j$;
         If ($p > pt$),{
            $pt = p$;
            $cat\_j = j$;
         }
         Else if $p = pt$, resolve tied maximum p-values;
         Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
         Compute summary statistics for the compound category $c$;
         Update $\Theta^{<m,t>}$;
      }
   }
}
If (TreeGrowingMethod=‘effectsize’),{
   While $\exists i \in \Theta^{<m,t>}$,((NodeSizeRequirement=’absolute’)and(((target is
   continuous)and($N_{f,i}^{<m,t>}$ < MinChildCasesABS))or((target is
   categorical)and($\sum_j n_{i,j}^{<m,t>}$ < MinChildCasesABS)))),
   or((NodeSizeRequirement=’percentage’)and(((target is
   continuous)and($N_{f,i}^{<m,t>}$ < MinChildCasesPct*$N_f$))or((target is
   categorical)and($\sum_j n_{i,j}^{<m,t>}$ < MinChildCasesPct*$N_f$)))),{
      If (predictor is nominal),{
         $et = 100$;
         $cat\_i = i$;
         For $\forall j \in \Theta^{<m,t>}$ and $j \neq i$,{
            Compute effect size $es$ for category $i$ and $j$;
            If ($es < et$),{
               $et = es$;
               $cat\_j = j$;
            }
         }
         Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
         Compute summary statistics for the compound category $c$;
         Update $\Theta^{<m,t>}$;
      }
      If (predictor is ordinal),{
         $et = 100$;
```

```
            $cat\_i = i$;
            Get category $j$ in $\Theta^{<m,t>}$ which is antecedent to $i$, if exists;
            Compute effect size $es$ for category $i$ and $j$;
            If ($es < et$),{
               $et = es$;
               $cat\_j = j$;
            }
            Get category $j$ in $\Theta^{<m,t>}$ which is subsequent to $i$, if exists;
            Compute effect size $es$ for category $i$ and $j$;
            If ($es < et$),{
               $et = es$;
               $cat\_j = j$;
            }
            Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
            Compute summary statistics for the compound category $c$;
            Update $\Theta^{<m,t>}$;
         }
      }
   }
8. Compute $p_{value}^{<m,t>}$ and effect size $E_s^{<m,t>}$ for the set of merged categories $\Theta^{<m,t>}$;
   // If ($|\Theta^{<m,t>}| = 1$),{
   //    $p_{value}^{<m,t>} = 1$;
   //    $E_s^{<m,t>} = 0$;
   // }
9. If (BonferroniAdjustment=true),{
      Compute adjusted p-value by applying Bonferroni adjustments;
      Let $p_{value}^{<m,t>}$ be the adjusted p-value;
   }
   // Bonferroni adjustments are described in section 5.3.2.
```

The function of `CHAID_Merging()` will be used by each Reducer in the map-reduce environment, see Appendix A for details.

Summary statistics for a compound category can be derived from those for original categories in the compound category. Denote the compound category as c, and the set of original categories in the compound category as Ω, the new summary statistics are calculated as,

$$N_{f,c}^{<m,t>} = \sum_{i\in\Omega} N_{f,i}^{<m,t>},$$

$$\bar{y}_{f,c}^{<m,t>} = \sum_{i\in\Omega} \frac{N_{f,i}^{<m,t>}}{N_{f,c}^{<m,t>}} \bar{y}_{f,i}^{<m,t>},$$

$$V_{f,c}^{<m,t>} = \sum_{i\in\Omega} \frac{N_{f,i}^{<m,t>}}{N_{f,c}^{<m,t>}} V_{f,i}^{<m,t>} + \sum_{i\in\Omega} \frac{N_{f,i}^{<m,t>}}{N_{f,c}^{<m,t>}} \left(\bar{y}_{f,i}^{<m,t>} - \bar{y}_{f,c}^{<m,t>}\right)\left(\bar{y}_{f,i}^{<m,t>} + \bar{y}_{f,c}^{<m,t>}\right),$$

$$N_{w,c}^{<m,t>} = \sum_{i\in\Omega} N_{w,i}^{<m,t>},$$

$$\bar{y}_{w,c}^{<m,t>} = \sum_{i\in\Omega} \frac{N_{w,i}^{<m,t>}}{N_{w,c}^{<m,t>}} \bar{y}_{w,i}^{<m,t>},$$

$$V_{w,c}^{<m,t>} = \sum_{i\in\Omega} \frac{N_{w,i}^{<m,t>}}{N_{w,c}^{<m,t>}} V_{w,i}^{<m,t>} + \sum_{i\in\Omega} \frac{N_{w,i}^{<m,t>}}{N_{w,c}^{<m,t>}} \left(\bar{y}_{w,i}^{<m,t>} - \bar{y}_{w,c}^{<m,t>}\right)\left(\bar{y}_{w,i}^{<m,t>} + \bar{y}_{w,c}^{<m,t>}\right),$$

$$n_{c,j}^{<m,t>} = \sum_{i\in\Omega} n_{i,j}^{<m,t>},$$

$$w_{c,j}^{<m,t>} = \sum_{i\in\Omega} w_{i,j}^{<m,t>}.$$

5.3.1. p-Value and Effect Size Calculations

Calculations of (unadjusted) p-values and effect sizes in the merging step depend on the type of target variable.

The merging step sometimes needs the p-value and effect size for a pair of original/compound categories, and sometimes needs the p-value and effect size for all the original/compound categories. For convenience, we denote the set of nonempty original/compound categories, for which the p-value and effect size are computed, as $\Gamma^{<m,t>}$.

Continuous Target Variable

If the target variable Y is continuous, perform an ANOVA F test that tests if the means of Y for different categories in $\Gamma^{<m,t>}$ are the same. This ANOVA F test calculates the F-statistic as

$$F = \frac{\sum_{i\in\Gamma^{<m,t>}} N_{w,i}^{<m,t>}\left(\bar{y}_{w,i}^{<m,t>} - \bar{y}_{w,\Gamma^{<m,t>}}^{<m,t>}\right)^2/(I-1)}{\sum_{i\in\Gamma^{<m,t>}} N_{w,i}^{<m,t>} V_{w,i}^{<m,t>}/(N_f'-I)},$$

where $N_f' = \sum_{i\in\Gamma^{<m,t>}} N_{f,i}^{<m,t>}$, $I = |\Gamma^{<m,t>}|$.

Accordingly, the p-value is calculated as

$$\text{p-value}=\begin{cases} \text{undefined}, & \text{If both numerator and denominator of F are zero;} \\ 0, & \text{Else if denominator of F is zero;} \\ \text{Prob}\{F(I-1, N_f'-I) > F\}, & \text{Otherwise.} \end{cases}$$

And $F(I-1, N_f'-I)$ is a random variable that follows a F-distribution with degrees of freedom $I-1$ and $N_f'-I$.

The effect size E_s is evaluated by the measure of $EtaSquare$, i.e.

$$E_s = 1 - \frac{\sum_{i\in\Gamma^{<m,t>}} N_{w,i}^{<m,t>} V_{w,i}^{<m,t>}}{\sum_{i\in\Gamma^{<m,t>}} N_{w,i}^{<m,t>} V_{w,\Gamma^{<m,t>}}^{<m,t>}}.$$

Nominal Target Variable

If the target variable Y is nominal categorical, the null hypothesis of independence of predictor X_m and Y is tested. According to the parameter of `ChiSquareType`, the p-value is computed based on either Pearson chi-squared statistic or likelihood ratio statistic.

The Pearson's chi-square statistic and likelihood ratio statistic are, respectively,

$$X^2 = \sum_{i\in\Gamma^{<m,t>}} \sum_{j\in\Delta^{<m,t>}} \frac{\left(n_{i,j}^{<m,t>} - \hat{m}_{i,j}\right)^2}{\hat{m}_{i,j}}$$

$$G^2 = 2 \sum_{i\in\Gamma^{<m,t>}} \sum_{j\in\Delta^{<m,t>}} n_{i,j}^{<m,t>} ln\left(n_{i,j}^{<m,t>} / \hat{m}_{i,j}\right)$$

where $\Delta^{<m,t>}$ denotes the set of nonempty target categories, and $\hat{m}_{i,j}$ is the estimated expected frequency following the independence model. The corresponding p-value is given by Prob$\{\chi_d^2 > X^2\}$ for Pearson's chi-square test or Prob$\{\chi_d^2 > G^2\}$ for likelihood ratio test, where χ_d^2 follows a chi-squared distribution with degrees of freedom $d = (J-1)(I-1)$, herein $J = |\Delta^{<m,t>}|$ and $I = |\Gamma^{<m,t>}|$.If case weight is not specified, the expected frequency is estimated by

$$\hat{m}_{i,j} = \frac{n_{i\cdot} n_{\cdot j}}{n_{\cdot\cdot}}$$

where $n_{i\cdot} = \sum_{j\in\Delta^{<m,t>}} n_{i,j}^{<m,t>}$, $n_{\cdot j} = \sum_{i\in\Gamma^{<m,t>}} n_{i,j}^{<m,t>}$, and $n_{\cdot\cdot} = \sum_{i\in\Gamma^{<m,t>}} \sum_{j\in\Delta^{<m,t>}} n_{i,j}^{<m,t>}$.

Else if case weight is specified, the expected frequency under the null hypothesis of independence is of the form

$$m_{i,j} = \bar{w}_{i,j}^{-1} \alpha_i \beta_j$$

where α_i and β_j are parameters to be estimated, and

$$\bar{w}_{i,j} = \frac{w_{i,j}^{<m,t>}}{n_{i,j}^{<m,t>}}.$$

Parameters estimates $\hat{\alpha}_i$, $\hat{\beta}_j$, and hence $\hat{m}_{i,j}$, are resulted from the following iterative procedure.

1. Initialize $k = 0$, $\alpha_i^{(0)} = \beta_j^{(0)} = 1$, $m_{i,j}^{(0)} = \bar{w}_{i,j}^{-1}$.
2. Compute $\alpha_i^{(k+1)} = \alpha_i^{(k)} \frac{n_{i\cdot}}{\sum_{j\in\Delta<m,t>} m_{i,j}^{(k)}}$.
3. Compute $\beta_j^{(k+1)} = \frac{n_{\cdot j}}{\sum_{i\in\Gamma<m,t>} \bar{w}_{i,j}^{-1}\alpha_i^{(k+1)}}$.
4. Compute $m_{i,j}^{(k+1)} = \bar{w}_{i,j}^{-1}\alpha_i^{(k+1)}\beta_j^{(k+1)}$.
5. If $k + 1$=`MaxIterations` or $\max_{i,j}\left|m_{i,j}^{(k+1)} - m_{i,j}^{(k)}\right| <$ `Epsilon`, stop and output $\alpha_i^{(k+1)}$, $\beta_j^{(k+1)}$, and $m_{i,j}^{(k+1)}$ as the final estimates. Otherwise, $k = k + 1$, go to step 2.

Given the chi-square test statistic χ_d^2, the effect size E_s is computed as

$$E_s = \left(\frac{\chi_d^2}{n_{\cdot\cdot} d_f}\right)^{1/2},$$

where $d_f = \min(I, J) - 1$, $\chi_d^2 = X^2, G^2$, or H^2 in below.

Ordinal Target Variable

If the target variable Y is categorical ordinal, the null hypothesis of independence of predictor X_m and Y is tested against the row effects model, with rows being the categories of X_m and columns the categories of Y, proposed by Goodman (1979). Two sets of expected frequencies $\hat{m}_{i,j}$ (under the hypothesis of independence) and $\hat{\hat{m}}_{i,j}$ (under the hypothesis that the data follow a row effects model), are both estimated. The likelihood ratio statistic is

$$H^2 = 2\sum_{i\in\Gamma<m,t>}\sum_{j\in\Delta<m,t>} \hat{\hat{m}}_{i,j} ln\left(\hat{\hat{m}}_{i,j}/\hat{m}_{i,j}\right).$$

The corresponding p-value is given by $\text{Prob}\{\chi_d^2 > H^2\}$ for likelihood ratio test, where χ_d^2 follows a chi-squared distribution with degrees of freedom $d = I - 1$, herein $I = |\Gamma^{<m,t>}|$.

In the row effects model, `Scores` for categories of Y are needed. By default, the order of a category of Y is used as the category score. Users can specify their own set of scores. Scores are set at the beginning of the tree and kept unchanged afterward. Let s_j be the score for category j of Y, $j \in \Delta^{<m,t>}$. The expected cell frequency under the row effects model is given by

$$m_{i,j} = \bar{w}_{i,j}^{-1}\alpha_i\beta_j\gamma_i^{(s_j-\bar{s})}$$

where

$$\bar{s} = \frac{\sum_{j\in\Delta^{<m,t>}} w_{\cdot j} s_j}{\sum_{j\in\Delta^{<m,t>}} w_{\cdot j}}$$

in which $w_{\cdot j} = \sum_{i\in\Gamma^{<m,t>}} w_{i,j}^{<m,t>}$, α_i, β_j, and γ_i are unknown parameters to be estimated. Parameters estimates $\hat{\hat{\alpha}}_i$, $\hat{\hat{\beta}}_j$, $\hat{\hat{\gamma}}_i$ and hence $\hat{\hat{m}}_{i,j}$, are resulted from the following iterative procedure.

1. Initialize $k = 0$, $\alpha_i^{(0)} = \beta_j^{(0)} = \gamma_i^{(0)} = 1$, $m_{i,j}^{(0)} = \bar{w}_{i,j}^{-1}$.
2. Compute $\alpha_i^{(k+1)} = \alpha_i^{(k)} \frac{n_{i\cdot}}{\sum_{j\in\Delta^{<m,t>}} m_{i,j}^{(k)}}$.
3. Compute $\beta_j^{(k+1)} = \frac{n_{\cdot j}}{\sum_{i\in\Gamma^{<m,t>}} \bar{w}_{i,j}^{-1} \alpha_i^{(k+1)} \left(\gamma_i^{(k)}\right)^{(s_j-\bar{s})}}$.
4. Compute $m_{i,j}^{*} = \bar{w}_{i,j}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)} \left(\gamma_i^{(k)}\right)^{(s_j-\bar{s})}$, $G_i = 1 + \frac{\sum_{j\in\Delta^{<m,t>}} (s_j-\bar{s})(n_{i,j}^{<m,t>} - m_{i,j}^{*})}{\sum_{j\in\Delta^{<m,t>}} (s_j-\bar{s})^2 m_{i,j}^{*}}$.
5. Compute $\gamma_i^{(k+1)} = \begin{cases} \gamma_i^{(k)} G_i, & \text{If } G_i > 0; \\ \gamma_i^{(k)}, & \text{Otherwise.} \end{cases}$
6. Compute $m_{i,j}^{(k+1)} = \bar{w}_{i,j}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)} \left(\gamma_i^{(k+1)}\right)^{(s_j-\bar{s})}$.
7. If $k + 1$=`MaxIterations` or $\max_{i,j}\left|m_{i,j}^{(k+1)} - m_{i,j}^{(k)}\right|$ < `Epsilon`, stop and output $\alpha_i^{(k+1)}$, $\beta_j^{(k+1)}$, $\gamma_i^{(k+1)}$, and $m_{i,j}^{(k+1)}$ as the final estimates. Otherwise, $k = k + 1$, go to step 2.

5.3.2. Bonferroni Adjustments for CHAID

The adjusted p-value is calculated as the p-value times a Bonferroni multiplier. The Bonferroni multiplier adjusts for multiple tests.

Suppose that there are I original categories of predictor X_m, including missing category if exists, in the set of merged categories $\Theta^{<m,t>}$, and it is reduced to l, $l = |\Theta^{<m,t>}|$, categories after the merging step. The Bonferroni multiplier B is the number of possible ways that I categories can be merged into l categories.

For $l = I$, $B = 1$. For $2 \leq l < I$, use the following equation

$$B = \begin{cases} \binom{I-1}{l-1}, & \text{Ordinal predictor;} \\ \sum_{v=0}^{l-1} (-1)^v \frac{(l-v)^I}{v!\,(l-v)!}, & \text{Nominal predictor;} \\ \binom{I-2}{l-2} + l\binom{I-2}{l-1}, & \text{Ordinal with a missing category.} \end{cases}$$

5.4. Splitting

When categories have been merged for all predictors, each predictor is evaluated for its association with the target variable, based on the p-value or effect size of the statistical test of association. The predictor with the strongest association, indicated by the smallest p-value or the largest effect size, is compared to the split threshold, `AlphaSplit` or `EffectSizeThreshold`. If the p-value is less than or equal to `AlphaSplit`, or the effect size is larger than or equal to `EffectSizeThreshold`, that predictor is selected as the split variable for the current node. Each of the merged categories of the split variable defines a child node of the split.

In map-reduce environment, the selection of the smallest p-value or the largest effect size can be performed efficiently in parallel. Firstly, each Reducer finds the locally smallest p-value or the locally largest effect size and passes it to the Controller. Then, the Controller sorts the local ones and gets the globally smallest p-value or the globally largest effect size. The following procedure is used during the process.

FindLocalBest()

```
Inputs:
   –  $p_{value}^{<m,t>}$
   –  $E_s^{<m,t>}$
   where $< m,t > \in \Psi_r$, $\Psi_r$ denotes the set of keys that are allocated to the $r$th
   Reducer

   <Parameter settings>
   –  TreeGrowingMethod       // {‘p-value’, ‘effectsize’}
   –  AlphaSplit              // Default 0.05
   –  EffectSizeThreshold
Outputs:
   –  $\Psi_r^*$
   // Set of keys with locally smallest p-values or largest effect sizes
Procedure:
1. Initially let $\Psi_r^*$ be empty;
2. If (TreeGrowingMethod=‘p-value’),{
      For $\forall \psi_r(t) \subseteq \Psi_r$,{  // Set $\psi_r(t)$ contains all keys in $\Psi_r$ corresponding to node $t$
         $< m,t >^* = arg\min_{<m,t>}\{p_{value}^{<m,t>}, < m,t > \in \psi_r(t)\}$; // Resolve tied minimum p-values
```

```
        If ($p_{value}^{<m,t>^*}$ ≤ AlphaSplit),
            $\Psi_r^* = \Psi_r^* \cup \{<m,t>^*\}$;
        }
    }
    If (TreeGrowingMethod=‘effectsize’),{
        For $\forall\psi_r(t) \subseteq \Psi_r$,{
            $<m,t>^* = arg\max_{<m,t>}\{E_s^{<m,t>}, <m,t> \in \psi_r(t)\}$;
            If ($E_s^{<m,t>^*}$ > EffectSizeThreshold),
                $\Psi_r^* = \Psi_r^* \cup \{<m,t>^*\}$;
        }
    }
3. Return $\Psi_r^*$;
```

FindGlobalBest()

```
Inputs:
    –  $p_{value}^{<m,t>}$
    –  $E_s^{<m,t>}$
    where $<m,t> \in \Psi_r^*$, $r = 1,\dots,R$

    <Parameter settings>
    –  TreeGrowingMethod        // {‘p-value’, ‘effectsize’}
Outputs:
    –  $\Psi^*$
    // Set of keys with globally smallest p-values or largest effect sizes
Procedure:
1. Let $\Psi = \bigcup_{r=1}^{R}\Psi_r^*$, and $\Psi^*$ be empty;
2. If(TreeGrowingMethod=‘p-value’)and($\Psi$ is not empty),{
        For $\forall\psi(t) \subseteq \Psi$,{    // Set $\psi(t)$ contains all keys in $\Psi$ corresponding to node $t$
            $<m,t>^* = arg\min_{<m,t>}\{p_{value}^{<m,t>}, <m,t> \in \psi(t)\}$;
            // Resolve tied minimum p-values
            $\Psi^* = \Psi^* \cup \{<m,t>^*\}$;
        }
    }
    If(TreeGrowingMethod=‘effectsize’)and($\Psi$ is not empty),{
        For $\forall\psi(t) \subseteq \Psi$,{
            $<m,t>^* = arg\max_{<m,t>}\{E_s^{<m,t>}, <m,t> \in \psi(t)\}$;
            $\Psi^* = \Psi^* \cup \{<m,t>^*\}$;
        }
    }
3. Return $\Psi^*$;
```

If the set Ψ^* is not empty, then the Controller will perform the splitting step. That is to split the node using the predictor suggested by each key in Ψ^*.

Splitting()

```
Inputs:
    –  $T(d)$                        // Current tree of depth $d$
    –  $\Theta^{<m,t>}$
    –  $p_{value}^{<m,t>}$                    // P-value, computed for $\Theta^{<m,t>}$
    –  TestStatistic             // Test statistic associated with $p_{value}^{<m,t>}$
```

```
   –  FreedomDegrees          // Freedom degrees associated with $p_{value}^{<m,t>}$
   –  $E_s^{<m,t>}$                  // Effect size
   <Continuous target>
   –  $N_{f,i}^{<m,t>}$
   –  $\bar{y}_{f,i}^{<m,t>}$
   –  $V_{f,i}^{<m,t>}$
   –  $N_{w,i}^{<m,t>}$
   –  $\bar{y}_{w,i}^{<m,t>}$
   –  $V_{w,i}^{<m,t>}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>}$
   –  $w_{i,j}^{<m,t>}$
   where $< m,t >\in \Psi^*$, $\Psi^*$ denotes the set of keys for splitting
Outputs:
   –  $T(d+1)$       // New tree of depth $d+1$
   –  $Q$            // Set of candidate non-terminal leaf nodes
Procedure:
1. Let $Q$ be empty;
2. For $\forall < m,t >\in \Psi^*$,{
      Save the following statistics for node $t$: $p_{value}^{<m,t>}$, TestStatistic,
      FreedomDegrees, and $E_s^{<m,t>}$;
      Split node $t$ using predictor $X_m$ according the set of categories $\Theta^{<m,t>}$;
      Let $q$ be the set of child nodes $t_i$, $i \in \Theta^{<m,t>}$;
      $\mathrm{Q} = \mathrm{Q} \cup q$;
      For $\forall t_i \in q$,{
         // Compute and save the following statistics for child node $t_i$
         // For continuous target
         $N_f(t_i) = N_{f,i}^{<m,t>}$;
         $\bar{y}_f(t_i) = \bar{y}_{f,i}^{<m,t>}$;
         $V_f(t_i) = V_{f,i}^{<m,t>}$;
         $N_w(t_i) = N_{w,i}^{<m,t>}$;
         $\bar{y}_w(t_i) = \bar{y}_{w,i}^{<m,t>}$;
         $V_w(t_i) = V_{w,i}^{<m,t>}$;
         // For categorical target
         $N_{f,j}(t_i) = n_{i,j}^{<m,t>}$, $j = 1, \ldots, J$;
         $N_{w,j}(t_i) = w_{i,j}^{<m,t>}$, $j = 1, \ldots, J$;
      }
   }
3. Denote the new tree as $T(d+1)$;
4. Return $T(d+1)$ and $Q$;
```

5.5. Stopping

After the split is applied to a node, the child nodes are examined to see if they warrant splitting further.

Stopping()

```
Inputs:
   –  $d$                        // Current tree depth
   –  $count$                    // Current number of tree nodes
   –  $Q$                        // Set of candidate non-terminal leaf nodes
   <Continuous target>
   –  $N_f(t)$
   –  $V_f(t)$
   <Categorical target>
   –  $N_{f,j}(t)$
   where $t \in Q$, and $j = 1, \dots, J$

   <Parameter settings>
   –  MaxTreeDepth             // Default 5
   –  MaxNodeNumber            // Default 1,000
   –  MinParentCasesABS        // Default 100
   –  MinParentCasesPct        // Default 2
   –  NodeSizeRequirement      // {'absolute', 'percentage'}, default 'absolute'
Outputs:
   –  $Q$       // Set of non-terminal leaf nodes
Procedure:
1. For $\forall t \in Q$,{
      If ($d$=MaxTreeDepth),
      or($count$=MaxNodeNumber),
      or((target is continuous)and($V_f(t) = 0$)),
      or((target is categorical)and($\exists j, N_{f,j}(t) > 0$ and $N_{f,j}(t) = \sum_{k=1}^{J} N_{f,k}(t)$)),
      or((NodeSizeRequirement='absolute')and(((target is continuous)and($N_f(t) <$
         MinParentCasesABS))or((target is categorical)and($\sum_{k=1}^{J} N_{f,k}(t) <$
         MinParentCasesABS)))),
      or((NodeSizeRequirement='percentage')and(((target is
         continuous)and($N_f(t) <$ MinParentCasesPct*$N_f$))or((target is
         categorical)and($\sum_{k=1}^{J} N_{f,k}(t) <$ MinParentCasesPct*$N_f$)))),
         $Q = Q - \{t\}$;
   }
2. Return $Q$;
```

6. Exhaustive CHAID Algorithm

Exhaustive CHAID differs from CHAID in that different merging strategy and Bonferroni adjustments are used in tree growth.

6.1. Merging

Merging step uses an exhaustive search procedure to merge any similar pair until only a single pair is left.

ExhaustiveCHAID_Merging()

```
Inputs:
   // Global summary statistics for predictor $X_m$ and node $t$
   <Continuous target>
   –  $N_{f,i}^{<m,t>}$
   –  $\bar{y}_{f,i}^{<m,t>}$
   –  $V_{f,i}^{<m,t>}$
   –  $N_{w,i}^{<m,t>}$
   –  $\bar{y}_{w,i}^{<m,t>}$
   –  $V_{w,i}^{<m,t>}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>}$
   –  $w_{i,j}^{<m,t>}$
   where $i = 1, \ldots, I_m$ and $j = 1, \ldots, J$

   <Parameter settings>
   –  TreeGrowingMethod        // {‘p-value’, ‘effectsize’}
   –  EffectSizeThreshold
   –  BonferroniAdjustment     // {true, false}, default true
   –  ChiSquareType            // {‘pearson’, ‘likelihood’}, default ‘pearson’
   –  Epsilon                  // Default 0.001
   –  MaxIterations            // Default 100
   –  MinChildCasesABS         // Default 50
   –  MinChildCasesPct         // Default 1
   –  NodeSizeRequirement      // {‘absolute’, ‘percentage’}, default ‘absolute’
   –  Scores                   // Vector value, scores for categories of $Y$
   –  SplitMergedCategories    // {true, false}, default false
Outputs:
   –  $\Theta^{<m,t>}$                  // The set of merged categories
   –  $p_{value}^{<m,t>}$                 // P-value, computed for $\Theta^{<m,t>}$
   –  TestStatistic            // Test statistic associated with $p_{value}^{<m,t>}$
   –  FreedomDegrees           // Freedom degrees associated with $p_{value}^{<m,t>}$
   –  $E_s^{<m,t>}$                   // Effect size
   <Continuous target>
   –  $N_{f,i}^{<m,t>}$
   –  $\bar{y}_{f,i}^{<m,t>}$
   –  $V_{f,i}^{<m,t>}$
   –  $N_{w,i}^{<m,t>}$
   –  $\bar{y}_{w,i}^{<m,t>}$
   –  $V_{w,i}^{<m,t>}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>}$
   –  $w_{i,j}^{<m,t>}$
   where $i \in \Theta^{<m,t>}$, $j = 1, \ldots, J$
Procedure:
1. If (target is continuous),
      $\Theta^{<m,t>} = \{i | N_{w,i}^{<m,t>} > 0, i = 1, \ldots, I_m\}$;
   If (target is categorical),
      $\Theta^{<m,t>} = \{i | \sum_{j=1}^{J} n_{ij}^{<m,t>} > 0, i = 1, \ldots, I_m\}$;
   // Notice that if the predictor is ordinal, $\Theta^{<m,t>}$ will not include the
   missing category initially.
2. $level = 0$;
   $\Theta_{level}^{<m,t>} = \Theta^{<m,t>}$;
   Compute p-value $p_{level}$ and effect size $E_{level}$ for the set of categories $\Theta_{level}^{<m,t>}$;
3. If ($\left|\Theta_{level}^{<m,t>}\right| \le 1$),
      Go to step 7;
4. If (TreeGrowingMethod=‘p-value’),{
      If (predictor is nominal),{
         $pt = -1$;
         For $\forall i \in \Theta_{level}^{<m,t>}$,{
            For $\forall j \in \Theta_{level}^{<m,t>}$ and $j > i$,{
               Compute p-value $p$ and effect size for category $i$ and $j$;
               If ($p > pt$),{
                  $pt = p$;
                  $cat\_i = i$;
                  $cat\_j = j$;
               }
               Else if $p = pt$, resolve tied maximum p-values;
            }
         }
      }
      If (predictor is ordinal),{
         $pt = -1$;
         For $\forall i \in \Theta_{level}^{<m,t>}$,{
            Get category $j$ in $\Theta_{level}^{<m,t>}$ which is subsequent to $i$, if exists;
            Compute p-value $p$ and effect size for category $i$ and $j$;
            If ($p > pt$),{
               $pt = p$;
               $cat\_i = i$;
               $cat\_j = j$;
            }
            Else if $p = pt$, resolve tied maximum p-values;
         }
      }
   }
   If (TreeGrowingMethod=‘effectsize’),{
      If (predictor is nominal),{
         $et = 100$;
         For $\forall i \in \Theta_{level}^{<m,t>}$,{
            For $\forall j \in \Theta_{level}^{<m,t>}$ and $j > i$,{
               Compute effect size $es$ for category $i$ and $j$;
               If ($es < et$),{
                  $et = es$;
                  $cat\_i = i$;
                  $cat\_j = j$;
               }
            }
         }
      }
      If (predictor is ordinal),{
         $et = 100$;
         For $\forall i \in \Theta_{level}^{<m,t>}$,{
            Get category $j$ in $\Theta_{level}^{<m,t>}$ which is subsequent to $i$, if exists;
            Compute effect size $es$ for category $i$ and $j$;
            If ($es < et$),{
               $et = es$;
               $cat\_i = i$;
               $cat\_j = j$;
            }
         }
      }
   }
   Merge $cat\_i$ and $cat\_j$ into a compound category $c$;
   Compute summary statistics for the compound category $c$;
5. Let A be the set of original categories in the new category $c$;
   If (TreeGrowingMethod=‘p-value’),
   and(SplitMergedCategories=true),
   and($3 \leq |\mathrm{A}| \leq 15$),{
      $p_{merge} = pt$;
      If (predictor is nominal),{
         $pt = 2$;
         For ($k = 1{:}\lfloor|\mathrm{A}|/2\rfloor$),{
            For ($\forall A1$ with $k$ categories belonging to A),{
               Let $A2 = A - A1$;
               Compute p-value $p$ and effect size for category $cat\_A1$ and $cat\_A2$;
               // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
               If ($p < pt$),{
                  $pt = p$;
                  $cat\_i = cat\_A1$;
                  $cat\_j = cat\_A2$;
               }
               Else if $p = pt$, resolve tied minimum p-values;
            }
         }
      }
      If (predictor is ordinal),{   // Set A consists of ordered categories
         $pt = 2$;
         Let $A1$ be the set consisting of the first category in $A$;
         Let $A2 = A - A1$;
         While ($A2$ is not empty),{
            Compute p-value $p$ and effect size for category $cat\_A1$ and $cat\_A2$;
            // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
            If ($p < pt$),{
               $pt = p$;
               $cat\_i = cat\_A1$;
               $cat\_j = cat\_A2$;
            }
            Else if $p = pt$, resolve tied minimum p-values;
            Move the first category in $A2$ into $A1$;
         }
      }
      If ($pt < p_{merge}$),{
         Split category $c$ into two categories $cat\_i$ and $cat\_j$;
         Compute summary statistics for categories $cat\_i$ and $cat\_j$;
      }
   }
   If (TreeGrowingMethod='effectsize'),
   and(SplitMergedCategories=true),
   and($3 \le |\mathrm{A}| \le 15$),{
      $e_{merge} = et$;
      If (predictor is nominal),{
         $et = -1$;
         For ($k = 1{:}\lfloor|\mathrm{A}|/2\rfloor$),{
            For ($\forall A1$ with $k$ categories belonging to A),{
               Let $A2 = A - A1$;
               Compute effect size $es$ for category $cat\_A1$ and $cat\_A2$;
               // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
               If ($es > et$),{
                  $et = es$;
                  $cat\_i = cat\_A1$;
                  $cat\_j = cat\_A2$;
               }
            }
         }
      }
      If (predictor is ordinal),{
         $et = -1$;
         Let $A1$ be the set consisting of the first category in $A$;
         Let $A2 = A - A1$;
         While ($A2$ is not empty),{
            Compute effect size $es$ for category $cat\_A1$ and $cat\_A2$;
            // Category $cat\_A1$ and $cat\_A2$ corresponds to $A1$ and $A2$ respectively
            If ($es > et$),{
               $et = es$;
               $cat\_i = cat\_A1$;
               $cat\_j = cat\_A2$;
            }
            Move the first category in $A2$ into $A1$;
         }
      }
      If ($et > e_{merge}$),{
         Split category $c$ into two categories $cat\_i$ and $cat\_j$;
         Compute summary statistics for categories $cat\_i$ and $cat\_j$;
      }
   }
6. Denote the new set of categories as $\Theta_{level+1}^{<m,t>}$;
   Compute p-value $p_{level+1}$ and effect size $E_{level+1}$ for the set of categories $\Theta_{level+1}^{<m,t>}$;
   $level = level + 1$;
   Go to step 3;
7. If (TreeGrowingMethod='p-value'),
      $level^* = arg\min_{level}\{p_{level}\}$;
   If (TreeGrowingMethod='effectsize'),
      $level^* = arg\max_{level}\{E_{level}\}$;
   $p_{value}^{<m,t>} = p_{level^*}$;
   $\Theta^{<m,t>} = \Theta_{level^*}^{<m,t>}$;
   $E_s^{<m,t>} = E_{level^*}$;
8. Same as Step 6 in the procedure of CHAID_Merging();
9. Same as Step 7 in the procedure of CHAID_Merging();
```

```
10.Same as Step 8 in the procedure of CHAID_Merging();
11.Same as Step 9 in the procedure of CHAID_Merging();
   // Bonferroni adjustments are described in section 6.1.1.
```

The function of `Exhaustive_CHAID_Merging()` will be used by each Reducer in the map-reduce environment, see Appendix A for details.

6.1.1. Bonferroni Adjustments for Exhaustive CHAID

Exhaustive CHAID merges two categories iteratively until two categories left. The Bonferroni multiplier B is the sum of number of possible ways of merging two categories at each iteration.

Suppose that there are I original categories of predictor X_m, including missing category if exists, in the set of merged categories $\Theta^{<m,t>}$, the Bonferroni multiplier B is calculated as

$$B = \begin{cases} \dfrac{I(I-1)}{2}, & \text{Ordinal predictor;} \\ \dfrac{I(I^2-1)}{2}, & \text{Nominal predictor;} \\ \dfrac{I(I-1)}{2}, & \text{Ordinal with a missing category.} \end{cases}$$

7. Assignment and Risk Estimation Algorithms

7.1. Assignment

Once the tree is grown successfully, we compute an assignment (also called action or decision) for each node. To predict the target variable value for an incoming case, we first find in which terminal node it falls, then use the assignment of that terminal node for prediction.

7.1.1. Node Assignment

For any node t, let d_t be the assignment given to node t,

$$d_t = \begin{cases} j^*(t), & Y \text{ is categorical} \\ \bar{y}_w(t), & Y \text{ is continuous} \end{cases}$$

$$j^*(t) = arg\min_i \sum\nolimits_j C(i|j)p(j|t),$$

$$\bar{y}_w(t) = \frac{\sum_{n \in \hbar(t)} w_n f_n y_n}{N_w(t)}$$

where $p(j|t)$ is the weighted probability of a case being in category j given that it is in node t, defined as

$$p(j|t) = \frac{N_{w,j}(t)}{N_w(t)},$$

where $N_{w,j}(t)$ is the weighted number of cases in node t with category j,

$$N_{w,j}(t) = \sum_{n \in \hbar(t)} w_n f_n \mathrm{I}(y_n = j)$$

and $N_w(t)$ is the weighted number of cases in node t,

$$N_w(t) = \sum_{n \in \hbar(t)} w_n f_n.$$

If there is more than one category j that achieves the minimum, choose $j^*(t)$ to be the smallest such j for which $N_{f,j}(t) = \sum_{n \in \hbar(t)} f_n \mathrm{I}(y_n = j)$ is greater than 0, or just the smallest j if $N_{f,j}(t)$ is zero for all of them.

If the target variable is categorical, for each target category in node t, a confidence value is calculated as

$$\frac{N_{f,j}(t)+1}{N_f(t)+J},$$

where $N_f(t) = \sum_{j=1}^{J} N_{f,j}(t)$.

7.1.2. Case Assignment

For a case with predictor vector X, the assignment or prediction $d_T(\mathrm{X})$ for this case by the tree T is

$$d_T(\mathrm{X}) = \begin{cases} j^*(t(\mathrm{X})), & Y \text{ is categorical} \\ \bar{y}(t(\mathrm{X})), & Y \text{ is continuous} \end{cases}$$

where $t(\mathrm{X})$ is the terminal node the case falls in. For categorical target, besides the prediction, the confidence for the predicted category is also available, as computed above.

In classification of new cases, missing values are handled as they are during tree growth, being treated as an additional category (possibly merged with other non-missing categories).

For nodes where there were no missing values in the training data, a missing category will not exist for the split of that node. In that case, cases with a missing value for the split variable are assigned as

$$j^*(t) = arg\max_j p(j|t),$$

where $p(j|t)$ is the weighted probability, as computed above.

7.2. Risk Estimation

Risk estimates describe the risk of error in predicted values for specific nodes of the tree and for the tree as a whole.

Note that case weight is not involved in risk estimation, though it is involved in tree growing process and assignment.

7.2.1. Risk Estimation of a Node

For classification tree, the risk estimate $r(t)$ of node t is computed as

$$r(t) = \frac{1}{N_f(t)} \sum_{j=1}^{J} N_{f,j}(t) C(j^*(t)|j).$$

For regression tree, the risk estimate $r(t)$ of node t is computed as

$$r(t) = \frac{1}{N_f(t)} \sum_{n \in \hbar(t)} f_n \left(y_n - \bar{y}_w(t)\right)^2 = V_f(t) + \left(\bar{y}_f(t) - \bar{y}_w(t)\right)^2.$$

7.2.2. Risk Estimation of a Tree

For both classification trees and regression trees, the risk estimate $R(T)$ for tree T is calculated by aggregating risk estimates for the terminal nodes $r(t)$:

$$R(T) = \frac{\sum_{t \in T'} N_f(t) r(t)}{\sum_{t \in T'} N_f(t)},$$

where T' is the set of terminal nodes in the tree.

7.3. Model Explanation

7.3.1. Classification Table

Classification table is computed only for categorical target.

Suppose j is one of the observed category, and j^* is one of the predicted category, then the count of cell $< j^*, j >$ in the classification table is computed

$C_{<j^*,j>} = \sum_{t \in T'_{j^*}} N_{f,j}(t),$

where T'_{j^*} denotes the set of leaf nodes whose node assignment is j^*. Insight and Interestingness Algorithms

8.1. Grouping Leaf Nodes

8.1.1. Continuous Target

Leaf nodes can be partitioned into groups with low, middle, or high target means, by the following procedure.

1. To simplify the formulas, we assume that leaf nodes in the collection $\{t_1, t_2, \ldots, t_{|T'|}\}$ are already sorted in descending order according to target means. The target mean of leaf node t_q is $\bar{y}_{t_q} = \bar{y}_f(t_q)$, the count is $N_{t_q} = N_f(t_q)$, and the corresponding standard error is computed as

   $s_{t_q} = \sqrt{\frac{1}{N_{t_q}(N-|T'|)} \sum_{i=1}^{|T'|} N_{t_i} V_f(t_i)}, q = 1, \ldots, |T'|$

   where $N = \sum_{i=1}^{|T'|} N_{t_i}$.
2. Conduct a one-sample t-test for the leaf node with the largest target mean. The

   hypothesis is $H_0 : \bar{y}_1 = \bar{y}$ vs. $H_A : \bar{y}_1 > \bar{y}$, where $\bar{y} = \frac{1}{N} \sum_{i=1}^{|T'|} N_{t_i} \bar{y}_{t_i}$. We use the one-tail test

   because it will provide more power. The t statistic is

   $t = \frac{\bar{y}_{t_1} - \bar{y}}{s_{t_1}}.$

   The test statistic has an asymptotic t distribution with degrees of freedom $d = N - |T'|$. The corresponding p-value is computed as p-value=$1 - prob(t_d \le t)$.

If p-value <= α (significance level, default 0.05), then the high group is formed by including the leaf node with the largest target mean.

3. Repeat the same process for the next leaf node, i.e. comparing $\bar{y}_{t_2}$ with $\bar{y}$, $\bar{y}_{t_3}$ with $\bar{y}$, etc. until no leaf node can be added into the high group.
4. Similarly, conduct a one-sample t-test for the leaf node with the smallest target mean. The hypothesis is $H_0 : \bar{y}_{t_{|T'|}} = \bar{y}$ vs. $H_A : \bar{y}_{t_{|T'|}} < \bar{y}$. The t statistic is

   $t = \frac{\bar{y}_{t_{|T'|}} - \bar{y}}{s_{t_{|T'|}}}$.

   The test statistic has an asymptotic t distribution with degrees of freedom $d = N - |T'|$. The corresponding p-value is computed as

   p-value=$1 - prob(t_d \le |t|)$.

   If p-value <= α (significance level, default 0.05), then the low group is formed by including the leaf node with the smallest target mean.
5. Repeat the same process for the next leaf node, i.e. comparing $\bar{y}_{t_{|T'|-1}}$ with $\bar{y}$, $\bar{y}_{t_{|T'|-2}}$ with $\bar{y}$, etc. until no leaf node can be added into the low group.
6. If some leaf nodes still exist after forming the high and low groups, they are grouped into the middle group.
7. The output is a list of the leaf nodes for the high, low, and medium groups with relevant test statistics.

8.1.2. Categorical Target

For categorical target, leaf nodes are grouped according to the mode of the target variable in each node, which is computed as

$$j^{**}(t) = arg\max_j N_{f,j}(t), t \in T'.$$

Notice that if one leaf node has multiple modes, it will belong to several groups. This results in overlaps between groups of leaf nodes. For each mode, a confidence value is computed as the difference of probabilities between the mode category and the category with the second largest frequency.

8.2. Unusual Leaf Nodes

8.2.1. Continuous Target

Detection of leaves with unusual low/high target means is based on the modified z-score method. This method is implemented by the procedure of $ModifiedZScore(A[\cdot], W[\cdot])$ (See Appendix B for details).

By calling this procedure, we let $A[\cdot]$ be the array of target means $\bar{y}_f(t)$ of leaf nodes and $W[\cdot]$ be the array of corresponding counts of cases $N_f(t)$, $t \in T'$.

The procedure returns an outlier strength value $O(t)$ for each leaf node. This value can be interpreted as

$$\begin{cases} \text{Leaf node } t \text{ has unusually high target mean,} & O(t) > 3, \\ \text{Leaf node } t \text{ has unusually low target mean,} & O(t) < -3. \end{cases}$$

Moreover, the outlier strength value $O(t)$ can be mapped into an interestingness score by calling the procedure of $MonotoneCubicInterpolation(S_t, I_t, |O(t)|)$ (See Appendix C for details), where we let the set of threshold values for outlier strength S_t be $\{0.0, 2.0, 3.0, +\infty\}$, and the set of threshold values for interestingness I_t be $\{0.00, 0.33, 0.67, 1.00\}$.

8.2.2. Categorical Target

For categorical target, unusual leaf nodes are defined as those who have significantly different target distributions from the population. Thus, unusual leaf nodes herein can also been called as significant leaf nodes. Moreover, we define influential categories as those who have the most contributions to the significance/unusualness.

Detect significant leaf nodes

1. For each leaf node t, calculate the test statistic,

   $$\chi_t^2 = \sum_{j=1}^{J} \frac{\left(N_{f,j}(t) - N_f(t)p_j\right)^2}{N_f(t)p_j}$$

   where $p_j = N_{f,j}(t_r)/N_f(t_r)$, and t_r is the root node. The statistic χ_t^2 follows a chi-squared distribution with degrees of freedom $J - 1$. The corresponding p-value is computed, and if p-value$\leq \alpha$ (significance level, default 0.05), leaf node t will be considered as a significant leaf node.

2. For each leaf node t, calculate the effect size,

   $$E_t = \left(\frac{\chi_t^2}{N_f(t)(J-1)}\right)^{\frac{1}{2}}$$

Detect influential categories

1. For each category of a significant leaf node, calculate the test statistic,

   $$\chi_{t,j}^2 = \frac{\left(N_{f,j}(t) - N_f(t)p_j\right)^2}{N_f(t)p_j(1-p_j)}$$

   The statistic $\chi_{t,j}^2$ follows a chi-squared distribution with 1 degree of freedom. The corresponding p-value is computed and adjusted by multiplying a constant J , and if the adjusted p-value is not larger than α (significance level, default 0.05), the jth category is considered as an influential category. In addition, it is an influential high category if $N_{f,j}(t) > N_f(t)p_j$, and an influential low category otherwise.
2. For each influential category, calculate the effect size,

   $$E_{t,j} = \left(\frac{\chi_{t,j}^2}{N_f(t)}\right)^{\frac{1}{2}}$$

Display strategies

If the above analyses generate too many significant leaf nodes and /or influential target categories, we can apply the following strategy to limit them.

1. Sort all significant leaf nodes by their effect size values in descending order. Then we can export/recommend top-*k* ones (default *k* = 3).
2. Sort high and low influential target categories in each significant leaf node by their effect size values in descending order separately. Then we can export/recommend top-*n* influential high categories and top-*n* influential low categories (default *n* = 1). If effect size is tied, then all ties in top-*n* would be exported.

Notice that the effect size of each leaf node can be mapped into an interestingness score by calling the procedure of $MonotoneCubicInterpolation(S_t, I_t, E_t)$ (See Appendix C for details), where we let the set of threshold values for effect size S_t be $\{0.0, 0.2, 0.6, 1.0, +\infty\}$, and the set of threshold values for interestingness I_t be $\{0.00, 0.33, 0.67, 1.00, 1.00\}$, i.e.

$$f(x) = MonotoneCubicInterpolation(S_t, I_t, x),$$

where x is the effect size E_t.

Considering significance and effect size together, we will use the following mapping function for the final interestingness score:

$$Interestingness(x, y) = \begin{cases} 0, & y > \alpha \\ f(x), & y \leq \alpha \end{cases}$$

where x is E_t and y is p-value.

Small numbers in chi-square tests

Monte Carlo method will be used to compute exact p-values when the expected counts in chi-square test are less than δ (default 5).

1. Randomly sample N_m (default 10,000) leaf node configurations $N_{f,j}^k(t)$ based on the marginal distribution p_j, $j = 1, \dots, J$, where $\sum_{j=1}^J N_{f,j}^k(t) = N_f(t)$, and $k \in [1, N_m]$.
2. Calculate the probability of each configuration,
$$p^k = \frac{N_f(t)!}{\prod_{j=1}^J N_{f,j}^k(t)!} \prod_{j=1}^J (p_j)^{N_{f,j}^k(t)}$$
3. Calculate the chi-square value for each configuration,
$$\chi_t^{2,k} = \sum_{j=1}^J \frac{\left(N_{f,j}^k(t) - N_f(t)p_j\right)^2}{N_f(t)p_j}$$
4. Calculate the exact p-value for leaf node t,
$$p_t^{exact} = \sum_{k=1}^{N_m} p^k I(\chi_t^{2,k} \geq \chi_t^2)$$
5. If $p_t^{exact} \leq \alpha$ (significance level, default 0.05), leaf node t is considered as a significant leaf node.
6. Further, collect the chi-square test statistic for each configuration and for each target category
$$\chi_{t,j}^{2,k} = \frac{\left(N_{f,j}^k(t) - N_f(t)p_j\right)^2}{N_f(t)p_j(1 - p_j)}$$
Then compute the exact p-value for each target category of a significant leaf node
$$p_{t,j}^{exact} = \sum_{k=1}^{N_m} p^k I(\chi_{t,j}^{2,k} \geq \chi_{t,j}^2)$$

8.3. Target Class Analysis

Target class analysis (TCA) applies only for a categorical target, and it is an approach of discovering insights from a couple of leaf node groups which are formed by including the leaf node with the highest probability of the target class one-by-one. The target class can be user-specified or determined automatically. In default, the target class is the minority class, that is, the one which has the minimal frequency.

To simplify the formulas, we assume that leaf nodes in the collection $\left\{t_1, t_2, \dots, t_{|T'|}\right\}$ are already sorted in descending order according to the probability of the target class. Then the first group G_0 is assumed to be empty, while group G_1 is formed by node t_1, and group G_2 is formed by node t_1 and t_2, and so on.

Notice that if ties occur when ranking according to the probability of the target class, do the follows:

a) Rank the tied nodes in descending order according to node sizes.

b) If ties occur in a), rank the tied nodes in ascending order according to node IDs.

For each group G_k, $k = 0, 1, 2, \dots, K$, where $K = |T'|$, the assignment of nodes in the group is the target class, while for other nodes the assignment is the class with the highest probability among non-target ones. For details, please refer to Section 7.1.1. Then the classification table, i.e. confusion matrix, is

	1	2	$\dots$	c^*	$\dots$	J
1	$N_{11}^{<k>}$	$N_{12}^{<k>}$	$\dots$	$N_{1c^*}^{<k>}$	$\dots$	$N_{1J}^{<k>}$
2	$N_{21}^{<k>}$	$N_{22}^{<k>}$	$\dots$	$N_{2c^*}^{<k>}$	$\dots$	$N_{2J}^{<k>}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c^*	$N_{c^*1}^{<k>}$	$N_{c^*2}^{<k>}$	$\dots$	$N_{c^*c^*}^{<k>}$	$\dots$	$N_{c^*J}^{<k>}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
J	$N_{J1}^{<k>}$	$N_{J2}^{<k>}$	$\dots$	$N_{Jc^*}^{<k>}$	$\dots$	$N_{JJ}^{<k>}$

Note that c^* denotes the target class, and $N_{ij}^{<k>} = \sum_{t\in T_i'} N_{f,j}(t)$, where T_i' is the set of leaf nodes whose assignment is class i. In the matrix, the rows give the predicted class labels, while the columns give the actual ones.

8.3.1. Model Accuracy

Model accuracy determined by group G_k is

$$ACC_{G_k} = \frac{\sum_{j=1}^{J} N_{jj}^{<k>}}{N_f},$$

where N_f is the total count of cases.

8.3.2. Group Size

Total number of cases in group G_k is

$$N_{G_k} = \sum_{j=1}^{J} N_{c^*j}^{<k>}.$$

Percentage of cases is

$$PTG_{G_k} = \frac{\sum_{j=1}^{J} N_{c^*j}^{<k>}}{N_f}.$$

8.3.3. True Positive Rate

For class j, true positive rate, i.e. recall rate, is

$$TPR_{G_k}^{j} = \frac{N_{jj}^{<k>}}{\sum_{i=1}^{J} N_{ij}^{<k>}}, j = 1, \dots, J.$$

8.3.4. False Positive Rate

For target class c^*, false positive rate is

$$FPR_{G_k} = \frac{\sum_{j \neq c^*} N_{c^*j}^{<k>}}{\sum_{i=1}^{J} \sum_{j \neq c^*} N_{ij}^{<k>}}.$$

8.3.5. Positive Predictive Value

For target class c^*, positive predictive value, i.e. precision, is

$$PPV_{G_k} = \frac{N_{c^*c^*}^{<k>}}{\sum_{j=1}^{J} N_{c^*j}^{<k>}}.$$

8.3.6. G-Mean

G-mean determined by group G_k is

$$Gmean_{G_k} = \left(\prod_{j=1}^{J} TPR_{G_k}^{j}\right)^{1/J}.$$

Notice that classes whose recall rate is constant zero across groups will be excluded from the calculation of the G-mean measure, and the number of J in the formula will be adjusted accordingly.

8.3.7. F-Measure

F-measure determined by group G_k is

$$Fmeasure_{G_k} = \frac{2*TPR_{G_k}^{c^*}*PPV_{G_k}}{TPR_{G_k}^{c^*}+PPV_{G_k}}.$$

8.3.8. Decision Rule Set

In this section, we describe how to get a simplified decision rule set for each group G_k by collapsing the original tree with respect to the target class. Moreover, we compute simplicity measures for the rule set, and use them later to select concise rule sets.

Given the original tree T, we do the follows:

1. If all the sibling leaf nodes have the same target class assignment, collapse all of them into the parent node, and take the target class as assignment of the parent node.
2. Else, merge all the sibling nodes which have the same target class assignment into a new leaf node, and take the target class as assignment of the new node.

The two steps above will be repeated until the tree cannot be collapsed further. Then, the simplified decision rule set consists of rules of all leaf nodes with target class assignment in the collapsed tree. A flag variable will be used to indicate whether the original decision rule has been collapsed.

For the simplified decision rule set, the first simplicity measure is

$$S_{G_k}^{1} = \sum_{t \in T^*} d(t),$$

where T^* is the set of leaf nodes with target class assignment in the collapsed tree, and $d(t)$ denotes the number of different predictors used by the rule of leaf node t, that is, an adjusted depth. If T^* is empty, let $S_{G_k}^1 = 0$.

The second simplicity measure is

$$S_{G_k}^2 = \frac{\sum_{t \in T^*} d(t)}{\sum_{t \in T^{**}} d(t)}$$

where T^{**} is the set of leaf nodes with target class assignment in the original tree. If T^{**} is empty, let $S_{G_k}^2 = 0$.

8.3.9. Concise Rule Set

An optimal decision rule set could be defined using any of goodness measures, e.g. model accuracy, G-mean, F-measure, etc. However, such an optimal rule set may often be too complicated to be understood. Concerning this, we provide an alternative rule set, which is not-bad but simple enough, i.e. concise rule set.

Suppose the goodness measure of the decision rule set for group G_k is T_{G_k}, $k = 1, 2, \dots, K$. The goodness measure of the optimal rule set is A, and correspondingly the first simplicity measure is B.

To determine the concise rule set, we use the following procedure:

1. Order all the decision rule sets in ascending order according to the second simplicity measure $S_{G_k}^2$.
2. The concise rule set is the first one that satisfies

    a. Goodness measure threshold: $\frac{T_{G_k}}{\mathrm{A}} > \delta$, default $\delta = 90\%$.

    b. Simplicity threshold: $\frac{S_{G_k}^1}{\mathrm{B}} < \delta$, default $\delta = 90\%$.

Notice that if B equals zero, only condition a will be checked.

8.4. Tree Interestingness

The above interestingness indices are defined for tree nodes. In this section, we describe interestingness indices for tree models.

As illustrated in the following table, there are many sub-indices, each of which characterizes one aspect of a tree model. These sub-indices can be combined into an overall interestingness index (See Appendix D for details), which can be used to rank different tree models.

Overall index	**Sub-index**	**Description**
Overall interestingness for a classification/regression tree	Model size, i.e. number of tree nodes	Given the trees have grown fully and optimally, smaller trees would be more interesting, since they could provide simpler and more intuitive decision rules. Model size, i.e. number of tree nodes N_t, can be mapped into an interestingness score by calling the procedure of $MonotoneCubicInterpolation(S_t, I_t, N_t)$, where we let the set of threshold values for model size S_t be $\{3, 50, 100, +\infty\}$, and the set of threshold values for interestingness I_t be $\{1.00, 0.67, 0.33, 0.00\}$.
	Unusualness of leaf nodes	The unusualness sub-index for a tree is computed by averaging on unusualness interestingness indices of leaf nodes, as defined in section 8.2.
	Model accuracy	The accuracy of a classification tree is $Acc = \frac{\sum_{t \in T'} N_{f,j^*(t)}(t)}{\sum_{t \in T'} N_f(t)}$. The accuracy of a random classification tree (using Mode) is $Acc_0 = \frac{N_{f,\hat{j}(t_r)}(t_r)}{N_f(t_r)}$, where t_r denotes the root node, and $\hat{j}(t_r)$ is the mode of the root node. The accuracy of a regression tree is $Acc = R_{square} = 1 - \frac{\sum_{i=1}^{\lvert T' \rvert} N_f(t_i) V_f(t_i)}{N_f(t_r) V_f(t_r)}$. The accuracy of a random regression tree (using Mean), Acc_0, is zero. Then, model accuracy can be mapped into an interestingness score by calling the procedure

		of $MonotoneCubicInterpolation(S_t, I_t, Acc)$, where we let the set of threshold values for model accuracy S_t be $\{Acc_0, 1\}$, and the set of threshold values for interestingness I_t be $\{0.00, 1.00\}$. If the model accuracy is lower than Acc_0, the interestingness will be zero. Based on the accuracy interestingness $I(Acc)$, the accuracy Acc can be interpreted as $Insight(Acc) = \begin{cases} weak, & I(Acc) \leq 0.33 \\ moderate, & 0.33 < I(Acc) \leq 0.67 \\ strong, & I(Acc) > 0.67 \end{cases}$ Note: Relative error, that is $1 - Acc$, will be computed and exported for a regression tree.

References

[1] Biggs, D., B. de Ville, and E. Suen. 1991. A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18, 49-62.

[2] Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. New York: Chapman & Hall/CRC.

[3] Fan Li, and Damir Spisic. Map-Reduce Algorithms for Univariate Statistics(ADD).

[4] Goodman, L. A. 1979. Simple models for the analysis of association in cross-classifications having ordered categories. *Journal of the American Statistical Association*, 74, 537-552.

[5] Jane Chu, Sier Han. Linear Engine Phase I – Algorithm.

[6] Jing Xu. Comparison of binning methods.

[7] Jing Xu, Xueying Zhang. ADD - Interestingness and Insights.

[8] Kass, G. 1980. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29:2, 119-127.

[9] Sier Han, James Xu, Weicai Zhong. Algorithm: SmartReports Engine.

Appendix A. Map-Reduce Functions

A.1. Map Function

```
Inputs:
   –  Training cases in data split $k$
   –  $T(d)$      // Current tree of depth $d$
   –  $Q$         // Set of non-terminal leaf nodes
Outputs:
   <Continuous target>
   –  $N_{f,i}^{<m,t>(k)}$
   –  $\bar{y}_{f,i}^{<m,t>(k)}$
   –  $V_{f,i}^{<m,t>(k)}$
   –  $N_{w,i}^{<m,t>(k)}$
   –  $\bar{y}_{w,i}^{<m,t>(k)}$
   –  $V_{w,i}^{<m,t>(k)}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>(k)}$
   –  $w_{i,j}^{<m,t>(k)}$
   where $i = 1, \dots, I_m$, $j = 1, \dots, J$, $m = 1, \dots, M$, and $t \in \mathrm{Q}$
Procedure:
1. Start with
      $l = 0$;
      $N_{f,i}^{<m,t>}(l) = 0$;      // For continuous target
      $\bar{y}_{f,i}^{<m,t>}(l) = 0$;
      $V_{f,i}^{<m,t>}(l) = 0$;
      $N_{w,i}^{<m,t>}(l) = 0$;
      $\bar{y}_{w,i}^{<m,t>}(l) = 0$;
      $V_{w,i}^{<m,t>}(l) = 0$;
      $n_{i,j}^{<m,t>}(l) = 0$;      // For categorical target
      $w_{i,j}^{<m,t>}(l) = 0$;
2. Iterator points to the first case;
   While (Iterator does not point to NULL),{
      Get the current case $n$;
      If ($y_n$ is not missing),
      and($f_n$ is not missing, zero, or negative),
      and($w_n$ is not missing, zero, or negative),{
         Assign case $n$ to a leaf node $t$ by following the splits in tree $T(d)$;
         // In order to assign cases to leaf nodes efficiently, we should take a
         // proper data structure for tree $T(d)$
         If ($t \in Q$),{
            $N_{f,i}^{<m,t>}(l+1) = N_{f,i}^{<m,t>}(l) + f_n \mathrm{I}(x_{m,n} = i)$;      // For continuous target
            $\bar{y}_{f,i}^{<m,t>}(l+1) = \bar{y}_{f,i}^{<m,t>}(l) + \frac{f_n}{N_{f,i}^{<m,t>}(l+1)} \mathrm{I}(x_{m,n} = i)[y_n - \bar{y}_{f,i}^{<m,t>}(l)]$;
            $V_{f,i}^{<m,t>}(l+1) = \frac{N_{f,i}^{<m,t>}(l)}{N_{f,i}^{<m,t>}(l+1)} \left[ V_{f,i}^{<m,t>}(l) + \frac{f_n}{N_{f,i}^{<m,t>}(l+1)} \mathrm{I}\left(x_{m,n} = i\right) \left(y_n - \bar{y}_{f,i}^{<m,t>}(l)\right)^2 \right]$;
```

```
                $N_{w,i}^{<m,t>}(l+1) = N_{w,i}^{<m,t>}(l) + w_n f_n \mathrm{I}(x_{m,n} = i)$,
                $\bar{y}_{w,i}^{<m,t>}(l+1) = \bar{y}_{w,i}^{<m,t>}(l) + \frac{w_n f_n}{N_{w,i}^{<m,t>}(l+1)} \mathrm{I}(x_{m,n} = i)[y_n - \bar{y}_{w,i}^{<m,t>}(l)]$;
                $V_{w,i}^{<m,t>}(l+1) = \frac{N_{w,i}^{<m,t>}(l)}{N_{w,i}^{<m,t>}(l+1)} \left[V_{w,i}^{<m,t>}(l) + \frac{w_n f_n}{N_{w,i}^{<m,t>}(l+1)} \mathrm{I}(x_{m,n} = i) \left(y_n - \bar{y}_{w,i}^{<m,t>}(l)\right)^2\right]$;
                $n_{i,j}^{<m,t>}(l+1) = n_{i,j}^{<m,t>}(l) + f_n \mathrm{I}(x_{m,n} = i \cap y_n = j)$;    // For categorical target
                $w_{i,j}^{<m,t>}(l+1) = w_{i,j}^{<m,t>}(l) + w_n f_n \mathrm{I}(x_{m,n} = i \cap y_n = j)$;
                $l = l + 1$;
             }
          }
          Iterator points to the next case;
       }
3. Return the following statistics
       $N_{f,i}^{<m,t>(k)} = N_{f,i}^{<m,t>}(l)$;       // For continuous target
       $\bar{y}_{f,i}^{<m,t>(k)} = \bar{y}_{f,i}^{<m,t>}(l)$;
       $V_{f,i}^{<m,t>(k)} = V_{f,i}^{<m,t>}(l)$;
       $N_{w,i}^{<m,t>(k)} = N_{w,i}^{<m,t>}(l)$;
       $\bar{y}_{w,i}^{<m,t>(k)} = \bar{y}_{w,i}^{<m,t>}(l)$;
       $V_{w,i}^{<m,t>(k)} = V_{w,i}^{<m,t>}(l)$;
       $n_{i,j}^{<m,t>(k)} = n_{i,j}^{<m,t>}(l)$;       // For categorical target
       $w_{i,j}^{<m,t>(k)} = w_{i,j}^{<m,t>}(l)$;
```

A.2. Reduce Function

```
**Inputs:**
   // Local summary statistics
   <Continuous target>
   –  $N_{f,i}^{<m,t>(k)}$
   –  $\bar{y}_{f,i}^{<m,t>(k)}$
   –  $V_{f,i}^{<m,t>(k)}$
   –  $N_{w,i}^{<m,t>(k)}$
   –  $\bar{y}_{w,i}^{<m,t>(k)}$
   –  $V_{w,i}^{<m,t>(k)}$
   <Categorical target>
   –  $n_{i,j}^{<m,t>(k)}$
   –  $w_{i,j}^{<m,t>(k)}$
   where $k = 1, \dots, K$, $i = 1, \dots, I_m$, $j = 1, \dots, J$, and $< m, t > \in \Psi_r$, $\Psi_r$ denotes the set of keys
   that are allocated to the $r$th Reducer

   <Parameter settings>
   –  TreeGrowingMethod        // {‘p-value’, ‘effectsize’}
   –  AlphaMerge               // Default 0.05
   –  AlphaSplit               // Default 0.05
   –  AlphaSplitMerge          // Default 0.025
   –  EffectSizeThreshold
   –  BonferroniAdjustment     // {true, false}, default true
   –  ChiSquareType            // {‘pearson’, ‘likelihood’}, default ‘pearson’
   –  Epsilon                  // Default 0.001
   –  MaxIterations            // Default 100
   –  MinChildCasesABS         // Default 50
   –  MinChildCasesPct         // Default 1
   –  NodeSizeRequirement      // {‘absolute’, ‘percentage’}, default ‘absolute’
   –  Scores                   // Vector value, scores for categories of $Y$
   –  SplitMergedCategories    // {true, false}, default false
   –  MergingMethod            // {‘CHAID’, ‘Exhaustive CHAID’}, default ‘CHAID’
Outputs:
   –  $\Theta^{<m,t>}$                  // The set of merged categories
   –  $p_{value}^{<m,t>}$                  // P-value, computed for $\Theta^{<m,t>}$
   –  TestStatistic            // Test statistic associated with $p_{value}^{<m,t>}$
   –  FreedomDegrees           // Freedom degrees associated with $p_{value}^{<m,t>}$
   –  $E_s^{<m,t>}$                  // Effect size
   –  $N_{f,i}^{<m,t>}$                  // For continuous target
   –  $\bar{y}_{f,i}^{<m,t>}$
   –  $V_{f,i}^{<m,t>}$
   –  $N_{w,i}^{<m,t>}$
   –  $\bar{y}_{w,i}^{<m,t>}$
   –  $V_{w,i}^{<m,t>}$
   –  $n_{i,j}^{<m,t>}$                  // For categorical target
   –  $w_{i,j}^{<m,t>}$
   where $i \in \Theta^{<m,t>}$, $j = 1, \dots, J$, and $< m,t > \in \Psi_r^*$, $\Psi_r^*$ is the set of keys with the locally
   smallest p-values
Procedure:
1. For ($\forall < m,t > \in \Psi_r$),{
      $N_{f,i}^{<m,t>} = \sum_{k=1}^{K} N_{f,i}^{<m,t>(k)}$;      // For continuous target
      $\bar{y}_{f,i}^{<m,t>} = \sum_{k=1}^{K} \frac{N_{f,i}^{<m,t>(k)}}{N_{f,i}^{<m,t>}} \bar{y}_{f,i}^{<m,t>(k)}$;
      $V_{f,i}^{<m,t>} = \sum_{k=1}^{K} \frac{N_{f,i}^{<m,t>(k)}}{N_{f,i}^{<m,t>}} V_{f,i}^{<m,t>(k)} + \sum_{k=1}^{K} \frac{N_{f,i}^{<m,t>(k)}}{N_{f,i}^{<m,t>}} \left(\bar{y}_{f,i}^{<m,t>(k)} - \bar{y}_{f,i}^{<m,t>}\right)\left(\bar{y}_{f,i}^{<m,t>(k)} + \bar{y}_{f,i}^{<m,t>}\right)$;

      $N_{w,i}^{<m,t>} = \sum_{k=1}^{K} N_{w,i}^{<m,t>(k)}$;
      $\bar{y}_{w,i}^{<m,t>} = \sum_{k=1}^{K} \frac{N_{w,i}^{<m,t>(k)}}{N_{w,i}^{<m,t>}} \bar{y}_{w,i}^{<m,t>(k)}$;
      $V_{w,i}^{<m,t>} = \sum_{k=1}^{K} \frac{N_{w,i}^{<m,t>(k)}}{N_{w,i}^{<m,t>}} V_{w,i}^{<m,t>(k)} + \sum_{k=1}^{K} \frac{N_{w,i}^{<m,t>(k)}}{N_{w,i}^{<m,t>}} \left(\bar{y}_{w,i}^{<m,t>(k)} - \bar{y}_{w,i}^{<m,t>}\right)\left(\bar{y}_{w,i}^{<m,t>(k)} + \bar{y}_{w,i}^{<m,t>}\right)$;
      $n_{i,j}^{<m,t>} = \sum_{k=1}^{K} n_{i,j}^{<m,t>(k)}$;       // For categorical target
      $w_{i,j}^{<m,t>} = \sum_{k=1}^{K} w_{i,j}^{<m,t>(k)}$;
      If (MergingMethod=‘CHAID’),
         Run CHAID_Merging();
      Else,
         Run ExhaustiveCHAID_Merging();
   }
2. Run FindLocalBest();       // Get the local best set of keys $\Psi_r^*$
```

A.2. Controller

The Controller is responsible for launching a series of map-reduce jobs during the tree growth. Moreover, it grows the tree directly by performing tree-specific operations, e.g. splitting, stopping, etc.

```
Inputs:
   <Parameter settings>
   –  TreeGrowthThreshold     // Default 1,000,000
   –  AlphaMerge              // Default 0.05
   –  AlphaSplit              // Default 0.05
   –  AlphaSplitMerge         // Default 0.025
   –  EffectSizeChisqTest     // Default 0.05
   –  EffectSizeFTest         // Default 0.05
   –  BonferroniAdjustment    // {true, false}, default true
   –  ChiSquareType           // {‘pearson’, ‘likelihood’}, default ‘pearson’
   –  Costs                   // Misclassification costs
   –  Epsilon                 // Default 0.001
   –  MaxIterations           // Default 100
   –  MaxTreeDepth            // Default 5
   –  MaxNodeNumber           // Default 1,000
   –  MinChildCasesABS        // Default 50
   –  MinChildCasesPct        // Default 1
   –  MinParentCasesABS       // Default 100
   –  MinParentCasesPct       // Default 2
   –  NodeSizeRequirement     // {‘absolute’, ‘percentage’}, default ‘absolute’
   –  Scores                  // Vector value, scores for categories of $Y$
   –  SplitMergedCategories   // {true, false}, default false
   –  MergingMethod           // {‘CHAID’, ‘Exhaustive CHAID’}, default ‘CHAID’
Outputs:
   –  PMML         // Save the model of CHAID tree
   –  StatXML      // Save model diagnostics
Procedure:
1. If ($N_f \leq$ TreeGrowthThreshold),
      TreeGrowingMethod=‘p-value’;
   Else,
      TreeGrowingMethod=‘effectsize’;
   If (Target is continuous),
      EffectSizeThreshold=EffectSizeFTest;
   If (Target is categorical),
      EffectSizeThreshold=EffectSizeChisqTest;
2. Initially let $Q$ be an empty set;
3. Run CreateRootNode();
4. Add the root node into $Q$;
5. Let $count = 1$;    // Current number of tree nodes
   Let $d = 0$;        // Current tree depth
6. While ($Q$ is not empty),{
      Launch a map-reduce job, and get the following statistics
         $\Theta^{<m,t>}$,                     // The set of merged categories
   $p_{value}^{<m,t>}$,                    // P-value, computed for $\Theta^{<m,t>}$
   TestStatistic,                          // Test statistic associated with $p_{value}^{<m,t>}$
   FreedomDegrees,                         // Freedom degrees associated with $p_{value}^{<m,t>}$
   $E_s^{<m,t>}$                           // Effect size
   $N_{f,i}^{<m,t>}$,                      // For continuous target
   $\bar{y}_{f,i}^{<m,t>}$,
   $V_{f,i}^{<m,t>}$,
   $N_{w,i}^{<m,t>}$,
   $\bar{y}_{w,i}^{<m,t>}$,
   $V_{w,i}^{<m,t>}$,
   $n_{i,j}^{<m,t>}$,                      // For categorical target
   $w_{i,j}^{<m,t>}$,
where $i \in \Theta^{<m,t>}$, $j = 1, \dots, J$, and $< m,t > \in \Psi_r^*$, $r = 1, \dots, R$;
Run FindGlobalBest();          // Get the set $\Psi^*$
If ($\Psi^*$ is empty) and ($d = 0$),
   Return an error of “Stopping rules prevent any tree growth”;
   // In other words, no inputs are sufficiently related to the target
If ($d = 0$),{
   // $\Psi^*$ just contains the key for root node
   Save the following statistics for root node $t$:
      $p_{value}^{<m,t>}$,                 // P-value, computed for $\Theta^{<m,t>}$
      TestStatistic,                       // Test statistic associated with $p_{value}^{<m,t>}$
      FreedomDegrees,                      // Freedom degrees associated with $p_{value}^{<m,t>}$
      $E_s^{<m,t>}$                        // Effect size
   // For continuous target
   $N_f(t) = \sum_{i \in \Theta^{<m,t>}} N_{f,i}^{<m,t>}$;
   $\bar{y}_f(t) = \sum_{i \in \Theta^{<m,t>}} \frac{N_{f,i}^{<m,t>}}{N_f(t)} \bar{y}_{f,i}^{<m,t>}$;
   $V_f(t) = \sum_{i \in \Theta^{<m,t>}} \frac{N_{f,i}^{<m,t>}}{N_f(t)} V_{f,i}^{<m,t>} + \sum_{i \in \Theta^{<m,t>}} \frac{N_{f,i}^{<m,t>}}{N_f(t)} \left(\bar{y}_{f,i}^{<m,t>} - \bar{y}_f(t)\right)\left(\bar{y}_{f,i}^{<m,t>} + \bar{y}_f(t)\right)$;
   $N_w(t) = \sum_{i \in \Theta^{<m,t>}} N_{w,i}^{<m,t>}$;
   $\bar{y}_w(t) = \sum_{i \in \Theta^{<m,t>}} \frac{N_{w,i}^{<m,t>}}{N_w(t)} \bar{y}_{w,i}^{<m,t>}$;
   $V_w(t) = \sum_{i \in \Theta^{<m,t>}} \frac{N_{w,i}^{<m,t>}}{N_w(t)} V_{w,i}^{<m,t>} + \sum_{i \in \Theta^{<m,t>}} \frac{N_{w,i}^{<m,t>}}{N_w(t)} \left(\bar{y}_{w,i}^{<m,t>} - \bar{y}_w(t)\right)\left(\bar{y}_{w,i}^{<m,t>} + \bar{y}_w(t)\right)$;
   // For categorical target
   $N_{w,j}(t) = \sum_{i \in \Theta^{<m,t>}} w_{i,j}^{<m,t>}$, $j = 1, \dots, J$;
   $N_{f,j}(t) = \sum_{i \in \Theta^{<m,t>}} n_{i,j}^{<m,t>}$, $j = 1, \dots, J$;
}
If ($\Psi^*$ is empty),
   Let $Q$ be empty;
Else,{
   Compute the number of new splits: $newSplits = \sum_{<m,t> \in \Psi^*} |\Theta^{<m,t>}|$;
   If ($count + newSplits >$ MaxNodeNumber),{
      If ($d = 0$),
         Return an error of “The very first split has too many nodes”;
      Let $Q$ be empty;
   }
   Else,{
      Let $count = count + newSplits$;
```

```
            Run Splitting();    // Get tree $T(d+1)$ and new set $Q$
            Let $d = d + 1$;
            Run Stopping();
         }
      }
   }
7. Calculate node assignment and risk estimation for tree $T(d)$; // See section 7
8. Save $T(d)$ in PMML;
9. Save model diagnostics in StatXML;
```

Appendix B. Modified Z-Score Method

The procedure of $ModifiedZScore(A[\cdot], W[\cdot])$ is as follows:

1. Get the number of members in $A[\cdot]$, suppose it is K.
2. Find the median of $A[k]$, incorporating the corresponding frequencies $W[k]$. Denote the median as M , then $M = median\left(A[1]_{W[1]},..., A[K]_{W[K]}\right)$, where $A[k]_{W[k]}$ is a set which contains only $A[k]$ value with frequency $W[k]$.
3. Compute the median absolute deviation (MAD) of $A[k]$, again including the frequencies $W[k]$,

   $$MAD = median(\left|A[1]-M\right|_{W[1]},...,\left|A[K]-M\right|_{W[K]}),$$

   where $\left|A[k]-M\right|_{W[k]}$ is a set which contains only $\left|A[k]-M\right|$ value frequency $W[k]$.

4. If $MAD = 0$, compute outlier strength for each $A[k]$

   $$O[k] = \frac{A[k]-M}{1.253314 * MeanAD}$$

   where $MeanAD = \frac{\sum_{k=1}^{K} W[k]\left|A[k]-M\right|}{\sum_{k=1}^{K} W[k]}$.
5. Else, compute outlier strength as

   $$O[k] = \frac{A[k]-M}{1.4826 * MAD}.$$

Appendix C. Monotone Cubic Interpolation Method

$Interestingness(x) = MonotoneCubicInterpolation(S_t, I_t, x)$

where x is the input statistic which must have a monotonically increasing relationship with the interestingness score threshold values. S_t is a set of distinct threshold values for the input statistics, which have been accepted and commonly used by expert users to interpret the statistics. The positive infinity (+∞) is included if the input statistic is not bounded from above. I_t is a set of distinct threshold values for the interestingness scores that S_t corresponds to. The threshold values must be between 0 and 1. The size of S_t and I_t must be the same. There are at least two values in S_t excluding positive infinity (+∞).

Pre-processing

Let $\{x_k\} = sorted(S_t)$ such that $x_1 < \cdots < x_n$, where n is the number of values in S_t. Let $\{y\} = sorted(I_t)$ such that $y_1 < \cdots < y_n$.

Condition A: There are more than two threshold values for input statistics, and they are all finite numbers

Preparing for cubic interpolation

The following steps should be taken for preparing a cubic interpolation function construction.

Step 1: Compute the slopes of the secant lines between successive points.

$$\Delta_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

for $k = 1, \cdots, n-1$.

Step 2: Initialize the tangents at every data point as the average of the secants,

$$m_k = \frac{\Delta_{k-1} + \Delta_k}{2}$$

for $k = 2, \cdots, n-1$; these may be updated in further steps. For the endpoints, use one-sided differences: $m_1 = \Delta_1$ and $m_n = \Delta_{n-1}$.

Step 3: Let $\alpha_k = m_k / \Delta_k$ and $\beta_k = m_{k+1} / \Delta_k$ for $k = 1, \cdots, n - 1$.

If α or β are computed to be zero, then the input data points are not strictly monotone. In such cases, piecewise monotone curves can still be generated by choosing $m_k = m_{k+1} = 0$, although global strict monotonicity is not possible.

Step 4: Update $\mathbf{m_k}$

If $\alpha^2 + \beta^2 > 9$, then set $m_k = \tau_k \alpha_k \Delta_k$ and $m_{k+1} = \tau_k \beta_k \Delta_k$ where $\tau_k = \frac{3}{\sqrt{\alpha^2+\beta^2}}$.

Cubic interpolation

After the preprocessing, evaluation of the interpolated spline is equivalent to cubic Hermite spline, using the data x_k, y_k, and m_k for k=1,...,n.

To evaluate x in the range $[x_k, x_{k+1}]$ for k=1,...,n-1, calculate

$\mathrm{h} = \mathrm{x}_{\mathrm{k}+1} - \mathrm{x}_{\mathrm{k}}$ and $\mathrm{t} = \frac{\mathrm{x}-\mathrm{x}_{\mathrm{k}}}{\mathrm{h}}$

then the interpolant is

$\mathrm{f(x)} = \mathrm{y}_{\mathrm{k}}\mathrm{h}_{00}(\mathrm{t}) + \mathrm{h} * \mathrm{m}_{\mathrm{k}}\mathrm{h}_{10}(\mathrm{t}) + \mathrm{y}_{\mathrm{k}+1}\mathrm{h}_{01}(\mathrm{t}) + \mathrm{h} * \mathrm{m}_{\mathrm{k}+1}\mathrm{h}_{11}(\mathrm{t})$

where $\mathrm{h}_{\mathrm{ii}}(\mathrm{t})$ are the basis functions for the cubic Hermite spline.

$h_{00}(t)$	$2t^3 - 3t^2 + 1$
$h_{10}(t)$	$t^3 - 2t^2 + t$
$h_{01}(t)$	$-2t^3 + 3t^2$
$h_{11}(t)$	$t^3 - t^2$

Condition B: There are two threshold values for input statistics

As we have clarified in the beginning that there are at least two values in S_t excluding positive infinity (+∞), they must be both finite numbers when there are only two threshold values.

In this case the mapping function is a straight line connecting (x_1, y_1) and (x_2, y_2).

$$\mathrm{f(x)} = \mathbf{y_1} + (\mathbf{y_2} - \mathbf{y_1})\frac{\mathbf{x} - \mathbf{x_1}}{\mathbf{x_2} - \mathbf{x_1}}$$

Condition C: Threshold values include infinity

Note that there are at least two values in S_t excluding positive infinity (+∞). Take the last three statistic threshold values and threshold values for the interestingness scores from the sorted lists, we have three pairs of data (x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1}) and $(+\infty, y_n)$.

An exponential function

$$\mathrm{f(x)} = \mathrm{a} - \mathrm{be}^{-\mathrm{cx}}$$

can be defined by the pairs, where

$$\mathrm{a} = \mathrm{y_n},$$

$$\mathrm{b} = \sqrt[(\mathrm{x_{n-1}} - \mathrm{x_{n-2}})]{(\mathrm{y_n} - \mathrm{y_{n-2}})^{\mathrm{x_{n-1}}} \Big/ (\mathrm{y_n} - \mathrm{y_{n-1}})^{\mathrm{x_{n-2}}}},$$

$$\mathrm{c} = \frac{1}{\mathrm{x_{n-1}} - \mathrm{x_{n-2}}} \ln \frac{\mathrm{y_n} - \mathrm{y_{n-2}}}{\mathrm{y_n} - \mathrm{y_{n-1}}}.$$

If $n = 3$, which means there are only two distinct values in S_t excluding positive infinity (+∞), the exponential function is employed for evaluating x in the range [x_1, +∞).

Otherwise, the exponential function is for evaluating x in the range [$x_{\mathrm{n-1}}$, +∞). To evaluate x in the range [x_1, $x_{\mathrm{n-1}}$), use procedures under "condition A: There are more than two threshold values for input statistics, and they are all finite numbers" with data set $\{x_1, \cdots, x_{n'}\}$ and $\{y_1, \cdots, y_{n'}\}$, where $n^{'} = n - 1$. To insure a smooth transition to the exponential function, the tangent $m_{n'}$ at data point $x_{n'}$ is given as

$$m_{n'} = \frac{\mathrm{d}(\mathrm{a} - \mathrm{be}^{-\mathrm{cx}})}{\mathrm{dx}}\bigg|_{\mathrm{x} = \mathrm{x_{n'}}} = bce^{-cx_{n'}}$$

where a, b, c are computed as above.

Appendix D. Overall Interestingness Methods

The following methods can be used to combine interestingness sub-indices I_d $(d = 1,2, \dots, D)$ into an overall interestingness index.

Note that undefined interestingness sub-indices should be excluded from the calculation.

D.1. Weighted Average

The overall interestingness by the method of Weighted Average is

$$OverallInterestingness = \frac{\sum_{d=1}^{D} W_{I_d} I_d}{\sum_{d=1}^{D} W_{I_d}},$$

where W_{I_d} is the weight corresponding to the interestingness sub-index I_d.

In default, the weights are set as $1/D$. Another more comprehensive choice is to use normalized interestingness sub-indices as weights, i.e.

$$W_{I_d} = \frac{I_d}{\sum_{d=1}^{D} I_d}.$$

D.2. Maximum

The overall interestingness by the method of Maximum is

$$OverallInterestingness = max\{I_d, d = 1,2, \dots, D\}.$$

TwoStep-AS Cluster Algorithms

1. Introduction

Clustering technique is widely used by retail and consumer product companies who need to learn more about their customers in order to apply 1-to-1 marketing strategies. By means of clustering technique, customers are partitioned into groups by their buying habits, gender, age, income level, etc., and retail and consumer product companies can tailor their marketing and product development strategy to each customer group.

Traditional clustering algorithms can broadly be classified into partitional clustering and hierarchical clustering. Partitional clustering algorithms divide data cases into clusters by optimizing certain criterion function. A well-known representative of this class is the *k*-means clustering. Hierarchical clustering algorithms proceed by stages producing a sequence of partitions in which each partition is nested into the next partition in the sequence. Hierarchical clustering can be agglomerative and divisive. Agglomerative clustering starts with a singleton cluster (for example, a cluster that contains one data case only) and proceeds by successively merging the clusters at each stage. On the contrary, divisive clustering starts with one single cluster that contains all data cases and proceeds by successively separating the cluster into smaller clusters. Notice that no initial values are needed for hierarchical clustering.

However, traditional clustering algorithms do not adequately address the problem of large datasets. This is where the two-step clustering method can be helpful (see ref. [1][2]). This method first performs a pre-clustering step by scanning the entire dataset and storing the dense regions of data cases in terms of summary statistics called cluster features. The cluster features are stored in memory in a data structure called the CF-tree. Then an agglomerative hierarchical clustering algorithm is applied to cluster the set of cluster features. Since the set of cluster features is much smaller than the original dataset, the hierarchical clustering can perform well in terms of speed. Notice that the CF-tree is incremental in the sense that it does not require the whole dataset in advance and only scans the dataset once.

One essential element in the clustering algorithms above is the distance measure that reflects the relative similarity or dissimilarity of the clusters. *Chiu et al* proposed a new distance measure that enables clustering on data sets in which the features are of mixed types. The features can be continuous, nominal, categorical, or count. This distance measure is derived from a probabilistic model in the way that the distance is equivalent to the decrease in log-likelihood value as a result of merging two clusters. In the following, the new distance measure will be used in both the CF-tree growth and the clustering process, unless otherwise stated.

In this chapter, we extend the two-step clustering method into the distributed setting, specifically under the map-reduce framework. In addition to generating a clustering solution, we also provide mechanisms for selecting the most relevant features for clustering given data, as well as detecting rare outlier points. Moreover, we provide an enhanced set of evaluation and diagnostic features enabling insight, interactivity, and an improved overall user experience as required by the Analytic Catalyst application.

The chapter is organized as follows. We first declare some general notes about algorithms, development, etc. Then we define the notations used in the document. Operations for data pre-processing are introduced in section 4. In section 5, we briefly describe the data and the measures such as distance, tightness, and so on. In section 6, 7, and 8, we present the key algorithms used in model building, including CF-tree growth, Hierarchical Agglomerative Clustering (HAC), and determination of the number of clusters, respectively. Section 9 describes the entire solution of distributed clustering on Hadoop. Section 10 describes how to score new cases (to assign cluster memberships). Finally, Section 11 includes various measures used for model evaluation and model diagnostics. Insights and interestingness are also derived.

2. Notes

- To create CF-trees efficiently, we assume that operations within a main memory environment (for example, RAM) is efficient, and the size of the main memory can be allocated or controlled by user.
- We assume that the data is randomly partitioned. If this assumption is not allowed, sequential partition can still be applied. But note that the clustering result can be impacted, particularly if the data is ordered in some special way.
- CE is implemented in the Analytic Framework.

3. Notations

The following notations are used throughout this chapter unless otherwise stated:

R	Number of data partitions/splits.
N_j	Number of cases in cluster C_j.
N_{jk}	Number of cases in cluster C_j which have non-missing values in the kth feature.
K	Number of features used for clustering.
$x_i = (x_{i1}, \dots, x_{iK})$	The ith data case. x_i is a K-dimensional vector.
$x_{ik}^A, k = 1, \dots, K^A$	Value of the kth continuous feature of the ith data case x_i. There are K^A number of continuous features.
$x_{ik}^B, k = 1, \dots, K^B$	Value of the kth categorical feature of the ith data case x_i. There are K^B number of categorical features.
$L_k, k = 1, \dots, K^B$	Number of categories of the kth categorical feature in the entire data.
$N_{jkl}, k = 1, \dots, K^B, l = 1, \dots, L_k$	Number of cases in cluster C_j whose kth categorical feature takes the lth category.
$s_{jk}, k = 1, \dots, K^A$	Sum of values of the kth continuous feature in cluster C_j.
$s_{jk}^2, k = 1, \dots, K^A$	Sum of squared values of the kth continuous feature in cluster C_j.
$d(j, s)$	Distance between clusters C_j and C_s.
$C_{<j,s>}$	Cluster formed by combining clusters C_j and C_s.

4. Data Pre-processing

Data pre-processing includes the following transformations:

- Trailing blanks are trimmed
- Date/time features are transformed into continuous ones
- Normalize continuous features
- Category values of a categorical feature are mapped into integer. As such, the expression "$x_{ik}^B = l$" indicates that the kth categorical feature of the ith case takes the lth category.
- System/user missing and invalid values are all considered as missing.

- Cases with missing values in all features are discarded.

5. Data and Measures

Let x_i be the ith data case. Denote I_j as the index set of cluster C_j, $I_j = \{i: x_i \in C_j\}$. Let $K = K^A + K^B$ be the total number of features in which K^A of them are continuous and K^B are categorical. Without loss of generality, write x_i as

$$x_i = (x_{i1}, \dots, x_{iK}) = (x_{i1}^A, \dots, x_{iK^A}^A, x_{i1}^B, \dots, x_{iK^B}^B) \tag{1}$$

where x_{ik}^A is the value of the kth continuous feature, $k = 1, \dots, K^A$, and x_{ik}^B is the value of the kth categorical feature, $k = 1, \dots, K^B$. Express x_{ik}^B as a vector $(x_{ik1}^B, \dots, x_{ikL_k}^B)$ of L_k values in which each entry is either zero or one:

$$x_{ikl}^B = \begin{cases} 1, & \text{if } x_{ik}^B \text{ takes the } l\text{th category} \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

5.1. Cluster Feature of a Cluster

The cluster feature (sufficient statistics set) CF_j of a cluster C_j is a collection of statistics that summarizes the characteristics of a cluster. A possible set CF_j is given as

$$CF_j = \{N_j, \vec{N}_j, s_j, s_j^2, N_j^B\} \tag{3}$$

where N_j is the number of data cases in cluster C_j, $\vec{N}_j = \left(N_{jk}, \dots, N_{jK^A}, N'_{j1}, \dots, N'_{jK^B}\right)$ is a K-dimensional vector; the kth entry is the number of data cases in cluster C_j which have non-missing values in the kth feature. $s_j = \left(s_{j1}, \dots, s_{jK^A}\right)$ is a K^A-dimensional vector; the kth entry is the sum of the non-missing values of the kth continuous feature in cluster C_j, i.e.

$$s_{jk} = \sum_{i \in I_j} x_{ik}^A \tag{4}$$

for $k = 1, \dots, K^A$. Similarly, $s_j^2 = \left(s_{j1}^2, \dots, s_{jK^A}^2\right)$ is a K^A-dimensional vector such that the kth entry is the sum of squared non-missing values of the kth continuous feature in cluster C_j, i.e.

$$s_{jk}^2 = \sum_{i \in I_j} (x_{ik}^A)^2 \tag{5}$$

for $k = 1, \dots, K^A$.

Similarly, $N_j^B = \left(N_{j1}^B, \dots, N_{jK^B}^B\right)$ is a $\sum_{k=1}^{K^B}(L_k - 1)$-dimensional vector where the kth sub-vector N_{jk}^B is $(L_k - 1)$ dimensional, given by

$$N_{jk}^B = (N_{jk1}, \dots, N_{jk(L_k-1)}) \tag{6}$$

for $k = 1, \dots, K^B$. The lth entry N_{jkl} represents the total number of cases in cluster C_j whose kth categorical feature takes the lth category, $l = 1, \dots, L_k - 1$, i.e.

$$N_{jkl} = \sum_{i\in I_j} x_{ikl}^{B}. \quad (7)$$

5.2. Updating Cluster Feature when Merging Two Clusters

When two clusters C_j and C_s are said to merge, it simply means that the two corresponding sets of data points are merged together to form a union. In this case, the $CF_{<j,s>}$ for the merged cluster $C_{<j,s>}$ can be calculated by simply adding the corresponding elements in CF_j and CF_s, that is,

$$CF_{<j,s>} = \{N_j + N_s, \vec{N}_j + \vec{N}_s, s_j + s_s, s_j^2 + s_s^2, N_j^B + N_s^B\}. \quad (8)$$

5.3. Tightness of a Cluster

The interpretation of tightness of a cluster is that the smaller of the value of tightness, the less variation of the data cases within the cluster. In CE, there are two tightness measures, and they will be used depending on the specified distance measure, log-likelihood distance or Euclidean distance.

5.3.1. Tightness based on Log-likelihood Distance

The tightness $\tilde{\eta}_j$ of a cluster C_j can be defined as average negative log-likelihood function of the cluster evaluated at the maximum likelihood estimates of the model parameters. See Ref. 1 for statistical reasoning for definition.

The tightness $\tilde{\eta}_j$ of a cluster C_j is given by

$$\hat{\eta}_j = \frac{1}{2}\sum_{k=1}^{K^A} \ln\left(1 + \frac{\hat{\sigma}_{jk}^2}{\Delta_k}\right) + \sum_{k=1}^{K^B} \hat{E}_{jk} \quad (9)$$

where $\hat{\sigma}_{jk}^2$ is the maximum likelihood variance estimate of the kth continuous feature in cluster C_j.

$$\hat{\sigma}_{jk}^2 = \frac{s_{jk}^2 - N_{jk}(\hat{\mu}_{jk})^2}{N_{jk}} \quad (10)$$

in which $\hat{\mu}_{jk}$ is the sample mean,

$$\hat{\mu}_{jk} = \frac{s_{jk}}{N_{jk}}. \quad (11)$$

$\hat{E}_{jk}$ is the entropy of the kth categorical feature in cluster C_j,

$$\hat{E}_{jk} = -\sum_{l=1}^{L_k} \hat{q}_{jkl} \ln \hat{q}_{jkl} \quad (12)$$

in which $\hat{q}_{jkl}$ is the portion of data cases in cluster C_j whose kth categorical feature takes the lth category,

$$\hat{q}_{jkl} = \frac{N_{jkl}}{N'_{jk}}. \quad (13)$$

Finally, Δ_k is appositive scalar which is added to handle the degenerating conditions and balance the contributions between a continuous feature and a categorical one. The default value of Δ_k is 0.01.

To handle missing values, the definition of tightness assumes that the distribution of missing values is the same as for the observed non-missing points in the cluster.

Moreover, the following assumption is always applied:

$$x\ln(x) = 0, \text{if } x = 0. \tag{14}$$

5.3.2. Tightness based on Euclidean Distance

The tightness $\tilde{\eta}_j$ of a cluster C_j can be defined as the average Euclidean distance from member cases to the center/centroid of the cluster.

The tightness $\tilde{\eta}_j$ of a cluster C_j is given by

$$\tilde{\eta}_j = \sqrt{\sum_{k=1}^{K} \frac{s_{jk}^2 - N_{jk}(\hat{\mu}_{jk})^2}{N_{jk}}}. \tag{15}$$

Notice that if any feature in cluster C_j has all missing values, the feature will not be used in the computation.

5.4. Distance Measures between Two Clusters

Suppose clusters C_j and C_s are merged to form a new cluster $C_{<j,s>}$ that consists of the union of all data cases in C_j and C_s. Two distance measures are available.

5.4.1. Log-likelihood Distance

The distance between C_j and C_s can be captured by observing the corresponding decrease in log-likelihood as the result of combining C_j and C_s to form $C_{<j,s>}$.

The distance measure between two clusters C_j and C_s is defined as

$$d(j,s) = \sum_{k=1}^{K^A+K^B} d_k(j,s) = \tilde{\xi}_j + \tilde{\xi}_s - \tilde{\xi}_{<j,s>} \tag{16}$$

where

$$\tilde{\xi}_j = -\frac{1}{2}\sum_{k=1}^{K^A} N_{jk} \ln\left(\hat{\sigma}_{jk}^2 + \Delta_k\right) - \sum_{k=1}^{K^B} N'_{jk}\hat{E}_{jk} \tag{17}$$

and

$$d_k(j,s) = \begin{cases} \{-N_{jk} \ln\left(\hat{\sigma}_{jk}^2 + \Delta_k\right) - N_{sk} \ln(\hat{\sigma}_{sk}^2 + \Delta_k) + N_{<j,s>k} \ln\left(\hat{\sigma}_{<j,s>k}^2 + \Delta_k\right)\}/2, & \text{if feature } k \text{ is continuous} \\ -N'_{jk}\hat{E}_{jk} - N'_{sk}\hat{E}_{sk} + N'_{<j,s>k}\hat{E}_{<j,s>k}, & \text{if feature } k \text{ is categorical} \end{cases} \tag{18}$$

Note that since $\tilde{\xi}_{<j,s>}$ can be calculated by using the statistics in $CF_{<j,s>}$, the distance can be calculated by first updating the $CF_{<j,s>}$ of the merged cluster $C_{<j,s>}$.

To handle missing values, the definition of distance assumes that the contribution of missing values equals zero.

5.4.2. Euclidean Distance

The Euclidean distance can only be applied if all features are continuous.

The distance between two cases is clearly defined. The distance between two clusters is here defined by the Euclidean distance between the two cluster centers. A cluster center is defined as the vector of cluster means of each feature.

Suppose the centers/centroids of clusters C_j and C_s are $(\hat{\mu}_{j1}, \dots, \hat{\mu}_{jK})$ and $(\hat{\mu}_{s1}, \dots, \hat{\mu}_{sK})$ respectively, then

$$d(j,s) = \sqrt{\sum_{k=1}^{K} d_k^2(j,s)} = \sqrt{\sum_{k=1}^{K} \left(\hat{\mu}_{jk} - \hat{\mu}_{sk}\right)^2} \tag{19}$$

where

$$d_k(j,s) = \left|\hat{\mu}_{jk} - \hat{\mu}_{sk}\right|. \tag{20}$$

Again, any feature in cluster C_j with all missing values will not be used in the computation.

6. CF-Tree Building

CF-tree is a very compact summary of dataset in the way that each entry (leaf entry) in the leaf node is a sub-cluster which absorbs the data cases that are close together, as measured by the tightness index $\tilde{\eta}$ and controlled by a specific threshold value T. CF-tree is built dynamically as new data case is inserted, it is used to guide to a new insertion into the correct sub-cluster for clustering purposes.

CF-tree is a height-balanced tree with four parameters:

1. The branching factor B for the non-leaf nodes. It is the maximum number of entries that a non-leaf node can hold. A non-leaf entry is of the form $[CF_i, child_i], i = 1, \dots, B$, in which $child_i$ is a pointer to its ith child node and CF_i is the cluster feature of the sub-cluster represented by this child.
2. The branching factor L for the leaf nodes. It is the maximum number of entries that a leaf node can hold. A leaf entry is similar to a non-leaf entry except that is does not have a pointer. It is of the form $[CF_i], i = 1, \dots, L$.
3. The threshold parameter T that controls the tightness $\tilde{\eta}$ of any leaf entries. That is, all leaf entries in a leaf node must satisfy a threshold requirement that the tightness has to be less than T, i.e. $\tilde{\eta} \leq T$.
4. Maximum tree height H.

In addition, each leaf node has two pointers: "$prev$" and "$next$" which are used to chain all leaf nodes together for efficient scanning.

Figure 1 illustrates a CF-tree of branching factors $B = 2$, $L = 3$, and $H = 1$.

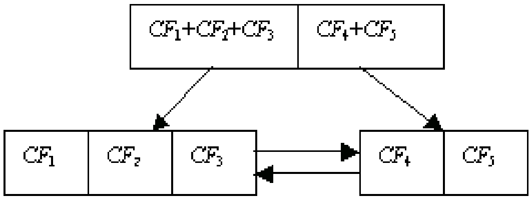


Figure 1. Example of a CF-tree.

6.1. Inserting a Single Case or a Sub-cluster into a CF-Tree

The procedure for inserting a data case or a sub-cluster (abbrev. "Ent") into a CF-tree is as follows.

Step 1. Identify the appropriate leaf node.
Starting from the root node, recursively descend the CF-tree by choosing the closest child node according to the distance measure d.

Step 2. Modify the leaf node.
Upon reaching a leaf node, find the closest leaf entry $[CF_i]$, say, and see if Ent can be absorbed into $[CF_i]$ without violating the threshold requirement $\tilde{\eta} \leq T$. If so, update the CF information in $[CF_i]$ to reflect the absorbing action. If not, add a new entry for Ent to the leaf. If there is space on the leaf for this new entry to fit in, then we are done. Otherwise, split the leaf node by choosing the farthest pair of entries as seeds, and redistribute the remaining entries based on the closest criteria.

Step 3. Modify the path to the leaf node.
After inserting Ent into a leaf node, update the CF information for each non-leaf entry on the path to the leaf node. If there is no leaf split, then only the corresponding CF information is needed to update to reflect the absorbing of Ent. If a leaf split happens, then it is necessary to insert a new non-leaf entry into the parent node in order to describe the newly created leaf. If the parent has space for this entry, at all higher levels, only the CF information is needed to update to reflect the absorbing of Ent. In general, however, the parent node has to split as well, and so on up to the root node. If the root node is split, the tree height increases by one.

Notice that the growth of CF-tree is sensitive to case order. If the same data case is inserted twice but at different time, the two copies might be entered into two distinct leaf entries. It is possible that two sub-clusters that should be in one cluster are split across nodes. Similarly, it is also possible that two sub-clusters that should not be in one cluster are kept together in the same node.

6.2. Threshold Heuristic

In building the CF-tree, the algorithm starts with an initial threshold value (default is 0). Then it scans the data cases and inserts into the tree. If the main memory runs out before data scanning is finished, the threshold value is increased to rebuild a new smaller CF-tree, by re-inserting the leaf entries of the old tree into the new one. After the old leaf entries have been re-inserted, data scanning is resumed from the case at which it was interrupted. The following strategy is used to update the threshold values.

Suppose that at step i, the CF-tree of the threshold T_i is too big for the main memory after N_i data cases in the data have been scanned, and an estimate of the next (larger) threshold T_{i+1} is needed to rebuild a new smaller CF-tree.

Specifically, we find the first two closest entries whose tightness is greater than the current threshold, and take it as the next threshold value. However, searching the closest entries can be tedious. So we follow BIRCH's heuristic to traverse along a path from the root to the most crowded leaf that has the most entries and find the pair of leaf entries that satisfies the condition.

6.3. Rebuilding CF-Tree

When the CF-tree size exceeds the size of the main memory, or the CF-tree height is larger than H, the CF-tree is rebuilt to a smaller one by increasing the tightness threshold.

Assume that within each node of CF-tree t_i, the entries are labeled contiguously from 0 to $n_k - 1$, where n_k is the number of entries in that node. Then a path from an entry in the root (level 1) to a leaf node (level h) can be uniquely represented by $(i_1, i_2, \ldots, i_{h-1})$, where $i_j, j = 1, \ldots, h-1$, is the label of the jth level entry on that path. So naturally, path $\left(i_1^{(1)}, i_2^{(1)}, \ldots, i_{h-1}^{(1)}\right)$ is before (or <) path $\left(i_1^{(2)}, i_2^{(2)}, \ldots, i_{h-1}^{(2)}\right)$ if $i_1^{(1)} = i_1^{(2)}, \ldots, i_{j-1}^{(1)} = i_{j-1}^{(2)}$, and $i_j^{(1)} <= i_j^{(2)}$ for $0 \leq j \leq h-1$. It is obvious that each leaf node corresponds to a path, since we are dealing with tree structure, and we will just use "path" and "leaf node" interchangeably from now on.

With the natural path order defined above, it scans and frees the old tree, path by path, and at the same time creates the new tree path by path. The procedure is as follows.

Step 1. Let the new tree start with NULL and OldCurrentPath be initially the leftmost path in the old tree.

Step 2. Create the corresponding NewCurrentPath in the new tree.
Copy the nodes along OldCurrentPath in the old tree into the new tree as the (current) rightmost path; call this NewCurrentPath

Step 3. Insert leaf entries in OldCurrentPath to the new tree.
With the new threshold, each leaf entry in OldCurrentPath is tested against the new tree to see if it can either by absorbed by an existing leaf entry, or fit in as a new leaf entry without splitting, in the NewClosestPath that is found top-down with the closest criteria in the new tree. If yes and NewClosestPath is before NewCurrentPath, then it is inserted to NewClosestPath, and deleted from the leaf node in NewCurrentPath.

Step 4. Free space in OldCurrentPath and NewCurrentPath.
Once all leaf entries in OldCurrentPath are processed, the nodes along OldCurrentPath can be deleted from the old tree. It is also likely that some nodes along NewCurrentPath are empty because leaf entries that originally corresponded to this path have been "pushed forward." In this case, the empty nodes can be deleted from the new tree.

Step 5. Process the next path in the old tree.
OldCurrentPath is set to the next path in the old tree if there still exists one, and go to step 2.

6.4. Delayed-Split Option

If the CF-tree that resulted by inserting a data case is too big for the main memory, it may be possible that some other data cases in the data can still fit in the current CF-tree without causing a split on any node in the CF-tree (thus the size of the current tree remains the same and can still be in the main memory).

Similarly, if the CF-tree resulted by inserting a data case exceeds the maximum height, it may be possible that some other data cases in the data can still fit in the current CF-tree without increasing the tree height.

Once any of the two conditions happens, such cases are written out to disk (with S_1 amount of disk space put aside for this purpose) and data scanning continues until the disk space runs out as well. The advantage of this approach is that more data cases can fit into the tree before a new tree is rebuilt. Figure 2 illustrates the control flow of delayed-split option.

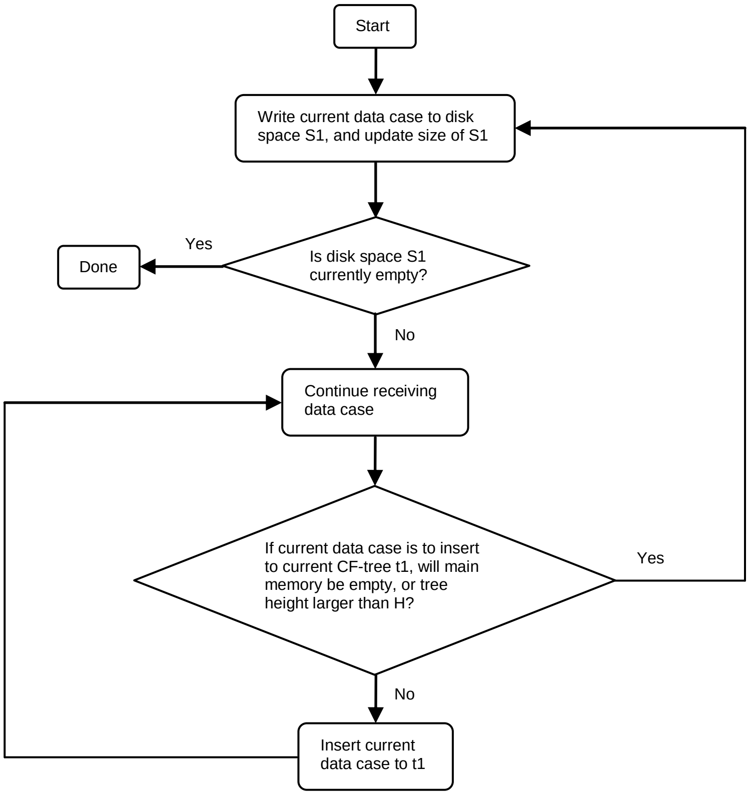

Start
Write current data case to disk space S1, and update size of S1
Is disk space S1 currently empty?
Yes
Done
No
Continue receiving data case
If current data case is to insert to current CF-tree t1, will main memory be empty, or tree height larger than H?
Yes
No
Insert current data case to t1

Figure 2. Control flow of delayed-split option.

6.5. Outlier-Handling Option

Outlier is defined as leaf entry (sub-cluster) of low density, which contains less than N_{min} (default 10) cases.

Similar to the delayed-split option, some disk space S_2 is allocated for handling outliers. When the current CF-tree is too big for the main memory, some leaf entries are treated as potential outliers (based on the definition of outlier) and are written out to disk. The others are used to rebuild the CF-tree. Figure 3 shows the control flow of the outlier-handling option.

Implementation notes:

- The size of any outlier leaf entry should also be less than 20% of the maximal size of leaf entries.
- The CF-tree t1 should be updated once any leaf entry is written to disk space S_2.

- Outliers identified here are local candidates, and they will be analyzed further in later steps, where the final outliers will be determined.

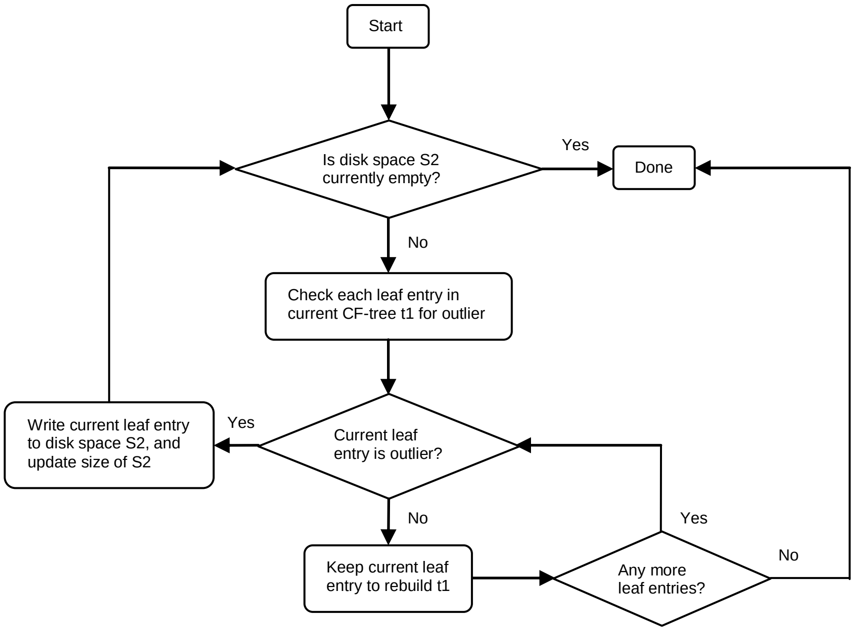

Start
Is disk space S2 currently empty?
Yes
Done
No
Check each leaf entry in current CF-tree t1 for outlier
Write current leaf entry to disk space S2, and update size of S2
Yes
Current leaf entry is outlier?
No
Yes
No
Keep current leaf entry to rebuild t1
Any more leaf entries?

Figure 3. Control flow of outlier-handling option.

6.6. Overview of CF-Tree Building

Figure 4 provides an overview of building a CF-tree for the whole algorithm. Initially a threshold value is set, data is scanned, and the CF-tree is built dynamically. When the main memory runs out, or the tree height is larger than the maximum height before the whole data is scanned, the algorithm performs the delayed-split option, outlier-handling option, and the tree rebuilding step to rebuild a new smaller CF-tree that can fit into the main memory. The process continues until all cases in the data are processed. When all data is scanned, cases in disk space S_1 are absorbed and entries in disk space S_2 are scanned again to verify if they are indeed outliers.

Implementation notes:

- When all data is scanned, all cases in disk space S_1 will be inserted into the tree. This may result in rebuilding the tree if necessary.

The following table shows the parameters involved in CF-tree building and their default values.

Parameter	Default value
Assigned main memory (M)	80*1024 bytes (TBD)
Assigned disk space for outlier-handling (S_2)	20% of M

Assigned disk space for delayed-split (S_1)	10% of M
Adjustment constant to the tightness and distance measures, $\Delta_k, k = 1, \ldots, K^A$	0.01
Distance measure (Log-likelihood/Euclidean)	Log-likelihood
Initial threshold value (T)	0
Branching factor (B)	8
Branching factor (L)	8
Maximum tree height (H)	3
Delayed-split option (on/off)	On
Outlier-handling option (on/off)	On
Outlier definition (N_{min})	Leaf entry which contains less than N_{min} cases, default 10

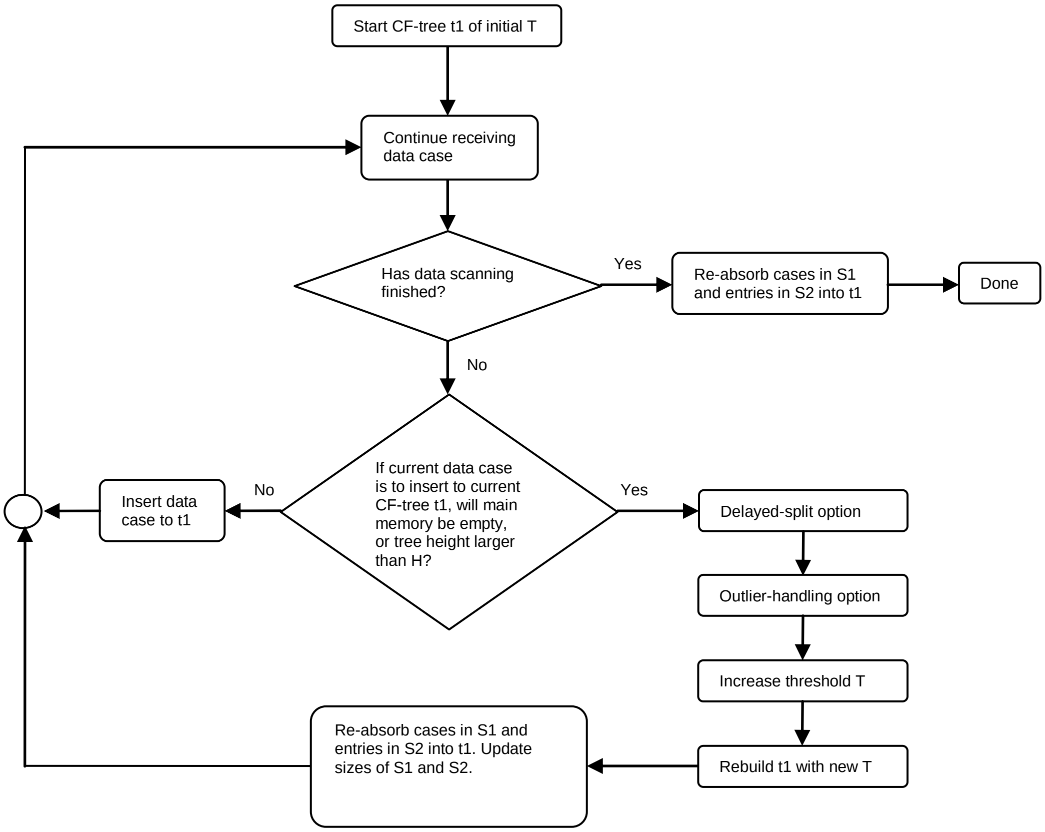

Start CF-tree t1 of initial T
Continue receiving data case
Has data scanning finished?
Yes
Re-absorb cases in S1 and entries in S2 into t1
Done
No
If current data case is to insert to current CF-tree t1, will main memory be empty, or tree height larger than H?
No
Insert data case to t1
Yes
Delayed-split option
Outlier-handling option
Increase threshold T
Rebuild t1 with new T
Re-absorb cases in S1 and entries in S2 into t1. Update sizes of S1 and S2.

Figure 4. Control flow of CF-tree building.

7. Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) proceeds by steps producing a sequence of partitions in which each partition is nested into the next partition in the sequence. See ref. [3] for details.

HAC can be implemented using two methods, as described below.

7.1. Matrix Based HAC

Suppose that matrix based HAC starts with J_0 clusters. At each subsequent step, a pair of clusters is chosen. The two clusters C_j and C_s in the pair are closest together in terms of the distance measure $d(j,s)$. A new cluster $C_{<j,s>}$ is formed to replace one of the clusters in the pair, C_j, say. This new cluster contains all data cases in C_j and C_s. The other cluster C_s is discarded. Hence the number of clusters is reduced by one at each step. The process stops when the desired number of clusters J_1 is reached. Since the distance measure between any two clusters that are not involved in the merge does not change, the algorithm is designed to update the distance measures between the new cluster and the other clusters efficiently.

The procedure of matrix based HAC is as follows.

```
Step 1.  For $j = 1, \dots, J_0 - 1$, {
                 Compute $d(j,s)$ for $s = j + 1, \dots, J_0$;
                 Find $\delta_j = \min_{s=j+1,\dots,J_0} d(j,s)$ and $s_j = \arg\min_{s=j+1,\dots,J_0} d(j,s)$;
         }
         Find $\delta_* = \min_{j=1,\dots,J_0-1} \delta_j$ and $j_* = \arg\min_{j=1,\dots,J_0-1} \delta_j$, the closest pair is $< j_*, s_{j_*} >$;
Step 2.  For $J = J_0 - 1, \dots, J_1$, {
                 Merge the closest pair $< j_*, s_{j_*} >$, and replace $C_{j_*}$ by $C_{<j_*,s_{j_*}>}$;
                 For $j = 1, \dots, j_* - 1$, {
                     If $s_j = s_{j_*}$, recompute all distances $d(j,s)$, $s = j + 1, \dots, J$, and update $\delta_j$ and $s_j$;
                     If $s_j \neq s_{j_*}$, {
                                 Compute $d = d(j, j_*)$;
                                 If $d < \delta_j$, update $\delta_j = d$ and $s_j = j_*$;
                                 If $d = \delta_j$, no change;
                                 If $d > \delta_j$ and $s_j = j_*$,
                                             Recompute all distances $d(j,s)$, $s = j + 1, \dots, J$, and update $\delta_j$ and $s_j$;
                                 If $d > \delta_j$ and $s_j \neq j_*$, no change;
                     }
                 }
                 For $j = j_*$, recompute all distances $d(j,s)$, $s = j + 1, \dots, J$, and update $\delta_j$ and $s_j$;
                 For $j = j_* + 1, \dots, s_{j_*} - 1$, {
                     If $s_j = s_{j_*}$, recompute all distances $d(j,s)$, $s = j + 1, \dots, J$, and update $\delta_j$ and $s_j$;
                     If $s_j \neq s_{j_*}$, no change;
                 }
                 For $j = s_{j_*} + 1, \dots, J$, no change;
                 Erase $C_{s_{j_*}}$;
                 Find $\delta_* = \min_{j=1,\dots,J} \delta_j$ and $j_* = \arg\min_{j=1,\dots,J} \delta_j$, the closest pair is $< j_*, s_{j_*} >$;
         }
```

Implementation notes:

- In order to reduce the memory requirement, it is not necessary to create an actual distance matrix when determining the closest clusters.
- If the Euclidean distance is used, the ward measure will be used to find the closest clusters. We just replace the distance measure $d(j,s)$ by $\frac{N_j N_s}{N_j + N_s} d^2(j,s)$. This also applies below for CF-tree based HAC.

7.2. CF-tree Based HAC

Suppose that CF-tree based HAC starts with K_0 CF-trees T_{CF}^k, $k = 1, \dots, K_0$ which contain J_0 leaf entries C_i, $i = 1, \dots, J_0$. Let $l(C_i)$ be the index of the CF-tree which contains the leaf entry C_i. For convenience, suppose $C_s > C_j$ if $l(C_s) > l(C_j)$.

At each subsequent step, a pair of leaf entries is chosen. The two leaf entries C_j and C_s in the pair are closest together in terms of the distance measure $d(j, s)$. A new leaf entry $C_{<j,s>}$ is formed to replace one of the leaf entries in the pair, C_j, say. This new leaf entry contains all data cases in C_j and C_s. The other leaf entry C_s is discarded. Hence the number of leaf entries is reduced by one at each step. Meanwhile, the involved CF-trees will be updated accordingly. The process stops when the desired number of leaf entries J_1 is reached. The output is the set of updated CF-trees, whose leaf entries indicate the produced clusters.

The procedure of CF-tree based HAC is as follows.

```
Step 1.  For $j = 1, \dots, J_0 - 1$, {
             Find the closest leaf entry $C_{s^k}$ in each CF-tree $T_{CF}^k$ for $k = l(C_j), \dots, K_0$, following the involved
             tree structure;
             Find $\delta_j = \min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$ and $s_j = \arg\min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$;
         }
         Find $\delta_* = \min_{j = 1, \dots, J_0 - 1} \delta_j$ and $j_* = \arg\min_{j = 1, \dots, J_0 - 1} \delta_j$, the closest pair is $< j_*, s_{j_*} >$;
Step 2.  For $J = J_0 - 1, \dots, J_1$, {
             Merge the closest pair $< j_*, s_{j_*} >$, update CF-tree $T_{CF}^{l(C_{j_*})}$ by the new leaf entry $C_{< j_*, s_{j_*} >}$, and
             remove the leaf entry $C_{s_{j_*}}$ from CF-tree $T_{CF}^{l(C_{s_{j_*}})}$;
             For $j = 1, \dots, j_* - 1$, {
                 If $s_j = s_{j_*}$, {
                         Find the closest leaf entry $C_{s^k}$ in each CF-tree $T_{CF}^k$ for $k = l(C_j), \dots, K_0$;
                         Find $\delta_j = \min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$ and $s_j = \arg\min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$;
                 }
                 If $s_j \neq s_{j_*}$, {
                         Compute $d = d(j, j_*)$;
                         If $d < \delta_j$, update $\delta_j = d$ and $s_j = j_*$;
                         If $d = \delta_j$, no change;
                         If $d > \delta_j$ and $s_j = j_*$, {
                                 Find the closest leaf entry $C_{s^k}$ in each CF-tree $T_{CF}^k$ for $k = l(C_j), \dots, K_0$;
                                 Find $\delta_j = \min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$ and $s_j = \arg\min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$;
                         }
                         If $d > \delta_j$ and $s_j \neq j_*$, no change;
                 }
             }
            For $j = j_*$, {
                Find the closest leaf entry $C_{s^k}$ in each CF-tree $T_{CF}^k$ for $k = l(C_j), \dots, K_0$;
                Find $\delta_j = \min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$ and $s_j = \arg\min_{s^k > j, k = l(C_j), \dots, K_0} d(j, s^k)$;
            }
             For $j = j_* + 1, \dots, s_{j_*} - 1$, {
                 If $s_j = s_{j_*}$, {
```

```
                Find the closest leaf entry $C_{s^k}$ in each CF-tree $T_{CF}^{k}$ for $k = l(C_j), \dots, K_0$;
                Find $\delta_j = \min_{s^k > j, k=l(C_j),\dots,K_0} d(j, s^k)$ and $s_j = \arg\min_{s^k > j, k=l(C_j),\dots,K_0} d(j, s^k)$;
            }
            If $s_j \neq s_{j_*}$, no change;
        }
        For $j = s_{j_*} + 1, \dots, J$, no change;
        Find $\delta_* = \min_{j=1,\dots,J} \delta_j$ and $j_* = \arg\min_{j=1,\dots,J} \delta_j$, the closest pair is $< j_*, s_{j_*} >$;
    }
Step 3.  Export updated non-empty CF-trees;
```

Clearly, CF-tree based HAC is very similar to matrix based HAC. The only difference is that CF-tree based HAC takes advantage of CF-tree structures to efficiently find the closest pair, rather than checking all possible pairs as in matrix based HAC.

8. Determination of the Number of Clusters

Assume that the hierarchical clustering method has been used to produce 1, 2 … clusters already. We consider the following two criterion indices in order to find the appropriate number of final clusters.

Bayesian Information Criterion (BIC):

$$\mathrm{BIC}(J) = -2\sum_{j=1}^{J} \tilde{\xi}_j + m_J \ln(N), \tag{23}$$

where N is the total number of cases in all the J clusters,

$$m_J = J\left\{2K^A + \sum_{k=1}^{K^B} (L_k - 1)\right\}. \tag{24}$$

Akaike Information Criterion (AIC):

$$\mathrm{AIC}(J) = -2\sum_{j=1}^{J} \tilde{\xi}_j + 2m_J. \tag{25}$$

Let $\mathrm{I}(J)$ be the criterion index (BIC or AIC) of J clusters, $\mathrm{d}(J)$ be the distance measure between the two clusters merged in merging $J+1$ clusters to J clusters, and J_C be the total number of sub-clusters from which to determine the appropriate number of final clusters.

Users can supply the range for the number of clusters $[J_{min}, J_{max}]$ in which they believe the “true” number of clusters should lie. Notice that if $J_C < J_{max}$, reset $J_{max} = J_C$.

The following four methods are proposed:

Method 1. Finding the number of clusters by information convergence.

Let $\Delta\mathrm{I}(J) = \mathrm{I}(J) - \mathrm{I}(J+1)$, where $\mathrm{I}(J)$ can be either $\mathrm{BIC}(J)$ or $\mathrm{AIC}(J)$ depending on user’s choice.

If $\Delta\mathrm{I}(1) \leq 0, J_I = J_{min}$. Else, let$R_1(J) = \Delta\mathrm{I}(J)/\Delta\mathrm{I}(1)$;

Let J_I be the smallest J in $[J_{min}, J_{max} - 1]$ which satisfies $R_1(J) < 0.1$, If none J satisfies the condition, let $J_I = J_{max}$.

Method 2. Finding the number of cluster by the largest distance jump.

To report $J_d = 1 + \arg\max_{J \in [J_{min}-1, J_{max}-1]}(\mathrm{d}(J)/\mathrm{d}(J+1))$ as the number of clusters.

Method 3. Finding the number of clusters by combining distance jump and information convergence aggressively

The process goes as follows:

a) Let $R_2(J) = \mathrm{d}(J)/\mathrm{d}(J+1)$.
b) Let J_1 be the largest J in $[J_{min}, J_I - 1]$ which satisfies $R_1(J) > 0.3$. If none J satisfies the condition, let $J_1 = J_{min} - 1$.
c) Calculate $R_2(J)$ for J in $[J_1, J_I - 1]$. Suppose that the max and the second max of $R_2(J)$ occurred at m_1 and m_2.
d) If $\frac{R_2(m_1)}{R_2(m_2)} > 1.3$, report $1 + m_1$ as the cluster number.
e) Otherwise, report $1 + \mathrm{MIN}(m_1, m_2)$.

Method 4. Finding the number of clusters by combining distance jump and information convergence conservatively

This method performs the same steps from a) to d) in method 3. But in step e), method 4 reports $1 + \mathrm{MAX}(m_1, m_2)$.

By default, method 3 is used with BIC as the information criterion.

9. Overview of the Entire Clustering Solution

Figure 5 illustrates the overview of the entire clustering solution.

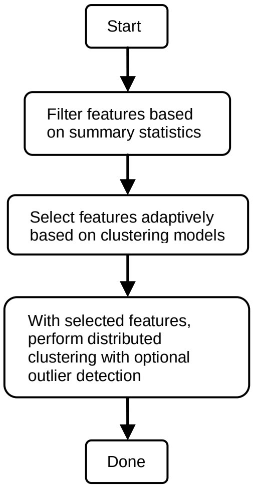

Start
Filter features based on summary statistics
Select features adaptively based on clustering models
With selected features, perform distributed clustering with optional outlier detection
Done

Figure 5. Control flow of the entire clustering solutin.

9.1. Feature Selection

9.1.1. Feature Filtering

Based on the summary statistics produced by DE, CE will perform an initial analysis and determine the features that are not useful for making the clustering solution. Specifically, the following features will be excluded.

#	Rule	Status	Comment
1	Frequency/analysis weight features	Required	
2	Identity features	Required	
3	Constant features	Required	
4	The percentage of missing values in any feature is larger than δ (default 70%)	Required	
5	The distribution of the categories of a categorical feature is extremely imbalanced, that is $RMSSE > \delta$ (default 0.7)	Discarded	The statistic of $RMSSE$ is the effect size for one sample chi-square test.
6	One category makes up the overwhelming majority of total population above a given percentage threshold δ (default 95%)	Required	
7	The number of categories of a categorical feature is larger than δ (default 24)	Required	
8	There are categories of a categorical feature with extremely high or low frequency, that is, the outlier strength is larger than δ (default 3)	Discarded	
9	The absolute coefficient of variation of a continuous feature is smaller than δ (default 0.05)	Required	

The remaining features will be saved for adaptive feature selection in the next step.

9.1.2. Adaptive Feature Selection

Adaptive feature selection selects the most important features for the final clustering solution. Specifically, it performs the following steps.

Step 1. Divide the distributed data into data splits.
Step 2. Build a local CF-tree on each data split.
Step 3. Distribute local CF-trees into multiple computing units. A unique key is assigned to each CF-tree.
Step 4. On each computing unit, start with all available features:

a. Perform matrix based HAC with all features on the leaf entries to get an approximate clustering solution, S0. Suppose there are J^* final clusters.
b. Compute importance for the set of all features.
c. Let $S^* = S0$ and I_{ref} be the information criteria of S0.
d. Remove features with non-positive importance as many as possible, and update S^* and I_{ref}.
e. Repeat to do the follows:
    i. Select the most unimportant feature from remaining features which are not checked.
    ii. Perform matrix based HAC with remaining features (not including the selected one) on the leaf entries to get a new approximate clustering solution, S1, with the fixed number of J^* clusters.

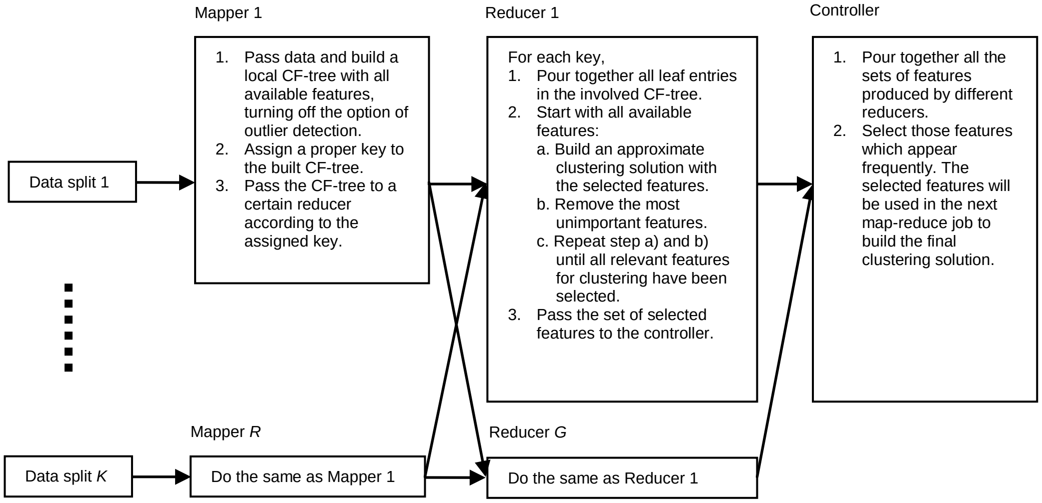
iii. If the information criteria of S1 plus the information of all discarded features determined by S1 is lower than I_{ref}, then remove the selected feature, and let $S^* = S1$.

iv. Continue to check the next feature.

f. Select the set of features corresponding to S^*.

Step 5. Pour together all the sets of features produced by different computing units. Discard any feature if its occurring frequency is less than $R * \beta$ (default $\beta = 50\%$). The remaining features will be used to build the final clustering solution.

The process described above can be implemented in parallel using one map-reduce job under the Hadoop framework, as illustrated in Figure 6. See appendix A for details the map-reduce implementation.



Mapper 1:

1. Pass data and build a local CF-tree with all available features, turning off the option of outlier detection.
2. Assign a proper key to the built CF-tree.
3. Pass the CF-tree to a certain reducer according to the assigned key.

Reducer 1:

For each key,
1. Pour together all leaf entries in the involved CF-tree.
2. Start with all available features:
   a. Build an approximate clustering solution with the selected features.
   b. Remove the most unimportant features.
   c. Repeat step a) and b) until all relevant features for clustering have been selected.
3. Pass the set of selected features to the controller.

Controller:

1. Pour together all the sets of features produced by different reducers.
2. Select those features which appear frequently. The selected features will be used in the next map-reduce job to build the final clustering solution.

Figure 6. One map-reduce job for feature selection.

Implementation notes:

- In default, the information based feature importance is used for the log-likelihood distance measure, and the effect size based feature importance is for the Euclidean distance.
- If no features are selected, just report all features.

9.2. Distributed Clustering

The Clustering Engine (CE) can identify clusters from distributed data with high performance and accuracy. Specifically, it performs the following steps:

Step 1. Divide the distributed data into data splits.

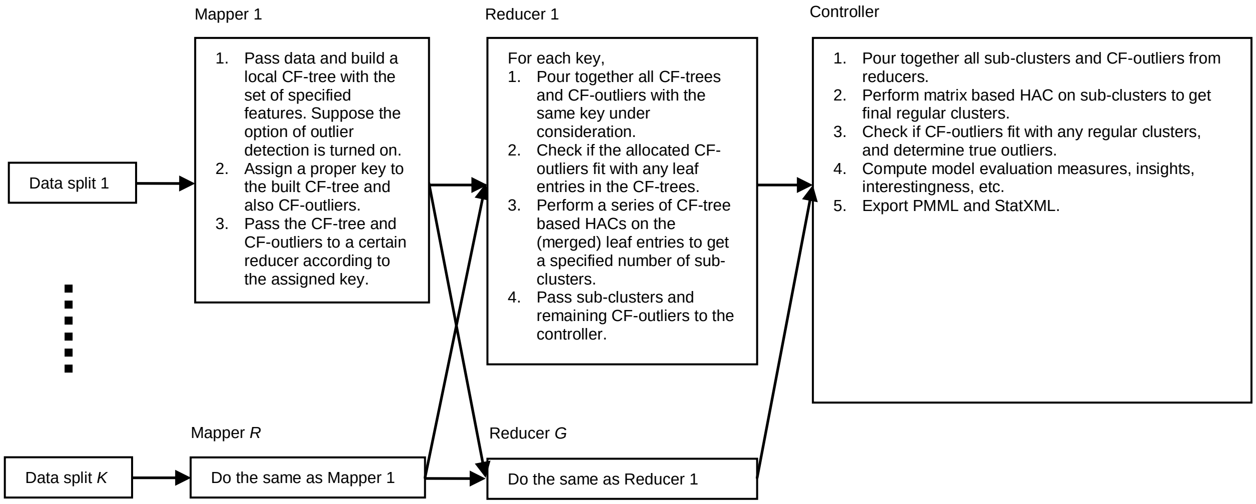
Step 2. Build a local CF-tree on each data split.

Step 3. Distribute local CF-trees into multiple computing units. Note that multiple CF-trees may be distributed to the same computing unit.

Step 4. On each computing unit, with all CF-entries in the involved CF-trees, perform a series of CF-tree based HACs, and get a specified number of sub-clusters.

Step 5. Pour together all the sub-clusters produced by different computing unit, and perform matrix based HAC to get the final clusters.
The number of final clusters is determined automatically or using a fixed one depending on the settings.

The process described above can be implemented in parallel using one map-reduce job under the Hadoop framework, as illustrated in Figure 7. See appendix B for details of the map-reduce implementation.



Mapper 1:

1. Pass data and build a local CF-tree with the set of specified features. Suppose the option of outlier detection is turned on.
2. Assign a proper key to the built CF-tree and also CF-outliers.
3. Pass the CF-tree and CF-outliers to a certain reducer according to the assigned key.

Reducer 1:

For each key,

1. Pour together all CF-trees and CF-outliers with the same key under consideration.
2. Check if the allocated CF-outliers fit with any leaf entries in the CF-trees.
3. Perform a series of CF-tree based HACs on the (merged) leaf entries to get a specified number of sub-clusters.
4. Pass sub-clusters and remaining CF-outliers to the controller.

Controller:

1. Pour together all sub-clusters and CF-outliers from reducers.
2. Perform matrix based HAC on sub-clusters to get final regular clusters.
3. Check if CF-outliers fit with any regular clusters, and determine true outliers.
4. Compute model evaluation measures, insights, interestingness, etc.
5. Export PMML and StatXML.

Figure 7. One map-reduce job for distributed clustering with outlier delection.

Implementation notes:

- The number of computing units is
  $Q = \lceil min(R * S/D_1, D_2/C_{min}) \rceil$, (28)
  where D_1 (default 50,000) is the number of data points which are suitable to perform CF-tree based HAC, D_2 (default 5,000) is the number of data points which are suitable to perform matrix based HAC, C_{min} is the minimal number of sub-clusters produced by each computing unit, and S is the maximal number of leaf entries, i.e. $B^H * L$, in a single CF-tree.
- The number of sub-clusters produced by each computing unit is
  $J_1 = \lfloor min(R * S, D_2)/Q \rfloor$. (29)

9.3. Distributed Outlier Detection

Outlier detection in the Clustering Engine will be based and will build upon the outlier handling method described previously in section 6. It is also extended to the distributed setting with the following steps:

Step 1. On each data split, perform the following:

1) Generate local candidate outliers according to the method described in section 6.
2) Distribute the local candidate outliers together with the associated CF-tree to a certain computing unit.

Step 2. Each computing unit is allocated with a set of candidate outliers and also a set of CF-trees containing regular leaf entries. For each member in the set of candidate outliers, it will be merged with the closest regular leaf entry only if the merging does not break the maximal tightness threshold among the involved CF-trees. Note that we will pass the CF-trees in order to enhance the performance of finding the closest regular leaf entry.

Step 3. Pour together all the remaining candidate outliers and sub-clusters produced by computing machines. Do the following:

1) Perform matrix based HAC on sub-clusters, and get the final regular clusters.
2) Keep only candidate outliers whose distance from the closest regular cluster to the center of the outlier candidate is greater than the corresponding cluster distance threshold $D_t(j)$
3) Merge the rest of candidate outliers with the corresponding closest regular clusters and update the distance threshold for each regular cluster.
4) For each remaining outlier cluster, compute its outlier strength.
5) Sort remaining outlier clusters according to outlier strength in descending order, and get the minimum outlier strength for the top P (default 5%) percent of outliers, and use it as an outlier threshold in scoring.
6) Export a specified number of the most extreme outlier clusters (default 20), along with the following measures for each cluster: cluster size, outlier strength, probabilities of belonging to each regular cluster.

Outlier strength of a cluster C_s is computed as

$$O(s) = \sum_{j=1}^{J} \frac{\tilde{d}(j,s)}{D_t(j)} p(j|s), \tag{30}$$

where $D_t(j)$ is the distance threshold of cluster C_j, which is the maximum distance from cluster C_j to each center of its starting sub-clusters in matrix based HAC, $\tilde{d}(j,s)$ is the distance from cluster C_j to the center of cluster C_s, and $p(j|s)$ is the probability of cluster C_s belonging to cluster C_j, that is

$$p(j|s) = \frac{\exp(-\tilde{d}(j,s))}{\sum_{j=1}^{J} \exp(-\tilde{d}(j,s))}. \tag{31}$$

Notice that the distance between the cluster center and a cluster C_j is computed by considering the center of cluster C_s as a singleton cluster $C_{s'}$. The cluster center herein is defined as the mean for a continuous feature, while being the mode for a categorical feature.

10. Cluster Membership Assignment

10.1. Without Outlier-Handling

Assign a case to the closest cluster according to the distance measure. Meanwhile, produce the probabilities of the case belonging to each regular cluster.

10.2. With Outlier-Handling

10.2.1. Legacy Method

Log-likelihood distance

Assume outliers follow a uniform distribution. Calculate both the log-likelihood resulting from assigning a case to a noise cluster and that resulting from assigning it to the closest non-noise cluster. The case is then assigned to the cluster which leads to the larger log-likelihood. This is equivalent to assigning a case to its closest non-noise cluster if the distance between them is smaller than a critical value $C = \ln(\prod_k R_k \prod_m L_m)$, where $\prod_k R_k$ is the product of ranges of continuous fields, and $\prod_m L_m$ is the product of category numbers of categorical fields. Otherwise, designate it as an outlier.

Euclidean distance

Assign a case to its closest non-noise cluster if the Euclidean distance between them is smaller than a critical value $C = 2\sqrt{\frac{1}{JK_A}\sum_{j=1}^{J}\sum_{k=1}^{K_A}\hat{\sigma}_{jk}^2}$. Otherwise, designate it as an outlier.

10.2.2. New Method

When scoring a new case, we compute the outlier strength of the case. If the computed outlier strength is greater than the outlier threshold, then the case is an outlier and otherwise belongs to the closest cluster. Meanwhile, the probabilities of the case belonging to each regular cluster are produced.

Alternatively, users can specify a customized outlier threshold (3, for example) rather than using the one found from the data.

11. Clustering Model Evaluation

Clustering model evaluation enables users to understand the identified cluster structure, and also to learn useful insights and interestingness derived from the clustering solution.

Note that clustering model evaluation can be done using cluster features and also the hierarchical dendrogram when forming the clustering solution.

11.1. Across-Cluster Feature Importance

Across-cluster feature importance indicates how influential a feature is in building the clustering solution. This measure is very useful for users to understand the clusters in their data. Moreover, it helps for feature selection, as described in section 12.2.

Across-cluster feature importance can be defined using two methods.

11.1.1. Information Criterion Based Method

If BIC is used as the information criterion, the importance of feature k is

$$\text{Importance}_k = \frac{\text{BIC}_k^0 - \text{BIC}_k}{\text{diff}_K^{max}}, \tag{32}$$

where

$$\text{BIC}_k^0 = \begin{cases} N_k \ln(\hat{\sigma}_k^2 + \Delta_k) + 2\ln(N), & \text{if feature } k \text{ is continuous} \\ 2N'_k \hat{E}_k + (L_k - 1)\ln(N), & \text{if feature } k \text{ is categorical}' \end{cases}$$

$$\text{BIC}_k = \begin{cases} \sum_{j=1}^{J} N_{jk} \ln\left(\hat{\sigma}_{jk}^2 + \Delta_k\right) + 2J\ln(N), & \text{if feature } k \text{ is continuous} \\ 2\sum_{j=1}^{J} N'_{jk} \hat{E}_{jk} + J(L_k - 1)\ln(N), & \text{if feature } k \text{ is categorical}' \end{cases}$$

$$\text{diff}_K^{max} = \max_k(\text{BIC}_k^0 - \text{BIC}_k),$$

and N_k, N'_k is the total valid count of feature k in the data, $\hat{\sigma}_k^2$ is the grand variance, and $\hat{E}_k$ is the grand entropy.

Notice that the information measure for the overall population has been decomposed as

$$\text{BIC}^0 = \sum_{k=1}^{K^A+K^B} \text{BIC}_k^0.$$

While if AIC is used, across-cluster importance is

$$\text{Importance}_k = \frac{\text{AIC}_k^0 - \text{AIC}_k}{\text{diff}_K^{max}}, \tag{33}$$

where

$$\text{AIC}_k^0 = \begin{cases} N_k \ln(\hat{\sigma}_k^2 + \Delta_k) + 4, & \text{if feature } k \text{ is continuous} \\ 2N'_k \hat{E}_k + 2(L_k - 1), & \text{if feature } k \text{ is categorical}' \end{cases}$$

$$\text{AIC}_k = \begin{cases} \sum_{j=1}^{J} N_{jk} \ln\left(\hat{\sigma}_{jk}^2 + \Delta_k\right) + 4J, & \text{if feature } k \text{ is continuous} \\ 2\sum_{j=1}^{J} N'_{jk} \hat{E}_{jk} + 2J(L_k - 1), & \text{if feature } k \text{ is categorical}' \end{cases}$$

$$\text{diff}_K^{max} = \max_k(\text{AIC}_k^0 - \text{AIC}_k).$$

Notice that, if the importance computed as above is negative, set it as zero. This also applies in the following.

Notice that the importance of a feature will be zero if the information difference corresponding to the feature is negative. This applies for all the calculations of information-based importance.

11.1.2. Effect Size Based Method

This method is similar to that used for defining association interestingness for bivariate variables. See ref. 6 for details.

Categorical Feature

For a categorical feature k, compute Pearson chi-square test statistic

$$\chi_p^2 = \sum_{l=1}^{L_k} \sum_{j=1}^{J} \frac{(N_{jkl}-E_{jkl})^2}{E_{jkl}}, \tag{34}$$

where

$$E_{jkl} = \frac{N_{jk\cdot} N_{\cdot kl}}{N_{\cdot k \cdot}}, \tag{35}$$

and

$$N_{jk\cdot} = \sum_{l=1}^{L_k} N_{jkl}, \tag{36}$$

$$N_{\cdot kl} = \sum_{j=1}^{J} N_{jkl}, \tag{37}$$

$$N_{\cdot k \cdot} = \sum_{l=1}^{L_k} \sum_{j=1}^{J} N_{jkl}. \tag{38}$$

The p-value is computed as

$$p_{value} = \text{Prob}\{\mathrm{X}^2 > \chi_p^2\}, \tag{39}$$

in which X^2 is a random variable that follows a chi-square distribution with freedom degree of $(J-1)(L_k-1)$. Note that categories with $N_{jk\cdot} = 0$ or $N_{\cdot kl} = 0$ will be excluded when computing the statistic and degrees of freedom.

The effect size, Cramer's V, is

$$E_s = \left(\frac{\chi_p^2}{N_{\cdot k \cdot}(q-1)}\right)^{1/2}, \tag{40}$$

where

$$q = \min(J, L_k). \tag{41}$$

The importance of feature k is produced by the following mapping function

$$\text{Importance}_k = \begin{cases} 0, & p_{value} \geq sig. \\ MonotoneCubicInterpolation(S_t, I_t, E_s), & p_{value} < sig. \end{cases} \tag{42}$$

where $sig.$ is significance level (default 0.05), S_t is a set of threshold values to assess effect size (default $S_t = \{0.0, 0.2, 0.6, 1.0\}$), I_t is a set of corresponding thresholds of importance (default $I_t = \{0.00, 0.33, 0.67, 1.00\}$), and $MonotoneCubicInterpolation(\cdot)$ is a monotone cubic interpolation mapping function between S_t and I_t.

Continuous Feature

For a continuous feature k, compute F test statistic

$$F = \frac{SSR/(J-1)}{SSE/(N_{\cdot k}-J)}, \tag{43}$$

where

$$SSR = \sum_{j=1}^{J} N_{jk}\left(\hat{\mu}_{jk} - \hat{\mu}_k\right)^2, \tag{44}$$

$$SSE = \sum_{j=1}^{J} N_{jk}\hat{\sigma}_{jk}^2, \tag{45}$$

$$N_{\cdot k} = \sum_{j=1}^{J} N_{jk}, \tag{46}$$

$$\hat{\mu}_k = \frac{\sum_{j=1}^{J} N_{jk}\hat{\mu}_{jk}}{N_{\cdot k}}. \tag{47}$$

The F statistic is undefined if the denominator equals zero. Accordingly, the p-value is calculated as

$$p_{value} = \begin{cases} \text{undefined,} & \text{if both the numerator and denominator of } F \text{ are zero;} \\ 0, & \text{else if the denominator of } F \text{ is zero;} \\ \text{Prob}\{F(J-1, N_{\cdot k} - J) > F\}, & \text{else.} \end{cases} \tag{48}$$

in which $F(J-1, N_{\cdot k} - J)$ is a random variable that follows a F-distribution with degrees of freedom $J-1$ and $N_{\cdot k} - J$.

The effect size, Eta square, is

$$E_s = 1 - \frac{\sum_{j=1}^{J} N_{jk}\hat{\sigma}_{jk}^2}{N_{\cdot k}\hat{\sigma}_k^2}, \tag{49}$$

where

$$\hat{\sigma}_k^2 = \frac{\sum_{j=1}^{J} s_{jk}^2 - N_{\cdot k}\hat{\mu}_k^2}{N_{\cdot k}}. \tag{50}$$

The importance of feature k is produced using the same mapping function as (42), and default $S_t = \{0.0, 0.04, 0.36, 1.0\}$.

11.2. Within-Cluster Feature Importance

Within-cluster feature importance indicates how influential a feature is in forming a cluster. Similar to across-cluster feature importance, within-cluster feature importance can also be defined using two methods.

11.2.1. Information Criterion Based Method

If BIC is used as the information criterion, the importance of feature k within cluster C_j $(j = 1, \dots, J)$ is

$$\text{Importance}_{k,j} = \frac{\text{BIC}_k^0 - \text{BIC}_{k,j}}{\text{diff}_{K,j}^{max}}, \tag{51}$$

where

$$\text{BIC}_{k,j} = \begin{cases} N_{jk}\ln\left(\hat{\sigma}_{jk}^2 + \Delta_k\right) + N_{j^ck}\ln\left(\hat{\sigma}_{j^ck}^2 + \Delta_k\right) + 2 * 2\ln{(N)}, & \text{if feature } k \text{ is continuous} \\ 2N'_{jk}\hat{E}_{jk} + 2N'_{j^ck}\hat{E}_{j^ck} + 2 * (L_k - 1)\ln(N)\,, & \text{if feature } k \text{ is categorical}' \end{cases} \tag{52}$$

$$\text{diff}_{K,j}^{max} = \max_k(\text{BIC}_k^0 - \text{BIC}_{k,j}). \tag{53}$$

Notice that j^c represents the complement set of j in J.

If AIC is used as the information criterion, the importance of feature k within cluster C_j $(j = 1, \dots, J)$ is

$$\text{Importance}_{k,j} = \frac{\text{AIC}_k^0 - \text{AIC}_{k,j}}{\text{diff}_{K,j}^{max}}, \tag{54}$$

where

$$\text{AIC}_{k,j} = \begin{cases} N_{jk}\ln\left(\hat{\sigma}_{jk}^2 + \Delta_k\right) + N_{j^ck}\ln\left(\hat{\sigma}_{j^ck}^2 + \Delta_k\right) + 4 * 2, & \text{if feature } k \text{ is continuous} \\ 2N'_{jk}\hat{E}_{jk} + 2N'_{j^ck}\hat{E}_{j^ck} + 2 * 2(L_k - 1), & \text{if feature } k \text{ is categorical} \end{cases} \tag{55}$$

$$\text{diff}_{K,j}^{max} = \max_k(\text{AIC}_k^0 - \text{AIC}_{k,j}). \tag{56}$$

11.2.2. Effect Size Based Method

Within-cluster importance is defined by comparing the distribution of the feature within a cluster with the overall distribution.

Categorical Feature

For cluster C_j $(j = 1, \dots, J)$ and a categorical feature k, compute Pearson chi-square test statistic

$$\chi_p^2 = \sum_{l=1}^{L_k} \frac{(N_{jkl} - E_{jkl})^2}{E_{jkl}}, \tag{57}$$

where

$$E_{jkl} = \frac{N_{jk\cdot} N_{\cdot kl}}{N_{\cdot k\cdot}}. \tag{58}$$

The p-value is computed as

$$p_{value} = \text{Prob}\{\text{X}^2 > \chi_p^2\}, \tag{59}$$

in which X^2 is a random variable that follows a chi-square distribution with freedom degree of $L_k - 1$. Note that importance for feature k within cluster C_j will be undefined if $N_{jk\cdot}$ equals zero.

The effect size is

$$E_s = \left(\frac{\chi_p^2}{N_{jk\cdot}(L_k-1)}\right)^{\frac{1}{2}}. \tag{60}$$

The importance of feature k within cluster C_j is produced using the same mapping function as (42), and default $S_t = \{0.0, 0.2, 0.6, 1.0\}$.

Continuous Feature

For cluster C_j $(j = 1, \dots, J)$ and a continuous feature k, compute t test statistic

$$t = \frac{\hat{\mu}_{jk} - \hat{\mu}_k}{s_d/\sqrt{N_{jk}}}, \tag{61}$$

where

$$s_d = \sqrt{\frac{N_{jk}}{N_{jk}-1}\hat{\sigma}_{jk}^2}. \tag{62}$$

The p-value is calculated as

$$p_{value} = \begin{cases} \text{undefined}, & \text{if both the numerator and denominator of } t \text{ are zero;} \\ 0, & \text{else if the denominator of } t \text{ is zero;} \\ 1 - \text{Prob}\{|T(N_{jk}-1)| \le |t|\}, & \text{else.} \end{cases} \tag{63}$$

in which $T(N_{jk} - 1)$ is a random variable that follows a t-distribution with degrees of freedom $N_{jk} - 1$.

The effect size is

$$E_s = \frac{|\hat{\mu}_{jk} - \hat{\mu}_k|}{s_d}.$$
(64)

The importance of feature k within cluster C_j is produced using the same mapping function as (42), and default $S_t = \{0.0, 0.2, 0.6, 1.0\}$.

11.3. Clustering Model Goodness

Clustering model goodness indicates the quality of a clustering solution. This measure will be computed for the final clustering solution, and it will also be computed for approximate clustering solutions during the process of adaptive feature selection.

Suppose there are J regular clusters, denoted as C_1,..., C_J. Let $l(i)$ be the regular cluster label assigned to sub-cluster i.

Then for each sub-cluster i, the Silhouette coefficient is computed approximately as

$$S_i = \frac{\Omega - \Phi}{\max(\Phi, \Omega)}, \tag{65}$$

where

Φ is the weighted average distance from the center of sub-cluster i to the center of every other sub-cluster assigned to the same regular cluster, that is,

$$\Phi = \frac{\sum_{s \neq i \text{ and } l(s)=l(i)} N_s d(i,s)}{\sum_{s \neq i \text{ and } l(s)=l(i)} N_s}, \tag{66}$$

Ω is the minimal average distance from the center of sub-cluster i to the center of sub-clusters in a different regular cluster among all different regular clusters, that is,

$$\Omega = \min\left\{ \frac{\sum_{l(s)=C_j} N_s d(i,s)}{\sum_{l(s)=C_j} N_s} \middle| j = 1, \ldots, J \text{ and } C_j \neq l(i) \right\}. \tag{67}$$

Clustering model goodness is defined as the weighted average Silhouette coefficient over all starting sub-clusters in the final stage of regular HAC, that is,

$$Goodness = \frac{\sum_j N_j S_j}{\sum_j N_j}. \tag{68}$$

The average Silhouette coefficient ranges between -1 (indicating a very poor model) and +1 (indicating an excellent model). As found by Kaufman and Rousseeuw (1990), average Silhouette greater than 0.5 indicates reasonable partitioning of data; lower than 0.2 means that data does not exhibit cluster structure. In this regard, we can use the following function to map $Goodness$ into an interestingness score:

$$Interestingness(Goodness) = MonotoneCubicInterpolation(S_t, I_t, Goodness), \tag{69}$$

where $S_t = \{-1.0, 0.2, 0.5, 1.0\}$, and $I_t = \{0.0, 0.0, 0.5, 1.0\}$.

Implementation notes:

- Please refer to section 9.3 for the definition of cluster center and also for the calculation of distance.
- When there is only a single sub-cluster in the regular cluster, let Φ be the tightness of the sub-cluster.

11.4. Special Clusters

With the clustering solution, we can find special clusters, which could be regular clusters with high quality, extreme outlier clusters, and so on.

11.4.1. Regular Cluster Ranking

To select the most useful or interesting regular clusters, we can rank them according to any of the measures described below.

Cluster tightness

Cluster tightness is given by equation (9) or (15).

Cluster tightness is not scale-free, and it is a measure of cluster cohesion.

Cluster importance

Cluster importance indicates the quality of the regular cluster in the clustering solution. A higher importance value means a better quality of the regular cluster.

If BIC is used as the information criterion, the importance for regular cluster C_j is

$$\text{Importance}_j = \frac{\text{BIC}^0 - \text{BIC}_j}{\text{diff}_J^{max}}, \tag{70}$$

where

$\text{BIC}_j = \sum_{k=1}^{K^A+K^B} \text{BIC}_{k,j}$,

$\text{BIC}^0 = \sum_{k=1}^{K^A+K^B} \text{BIC}_k^0$,

$\text{diff}_J^{max} = \max_j(\text{BIC}^0 - \text{BIC}_j)$.

If AIC is used as the information criterion, the importance for regular cluster C_j is

$$\text{Importance}_j = \frac{\text{AIC}^0 - \text{AIC}_j}{\text{diff}_J^{max}}, \tag{71}$$

where

$\text{AIC}_j = \sum_{k=1}^{K^A+K^B} \text{AIC}_{k,j}$,

$\text{AIC}^0 = \sum_{k=1}^{K^A+K^B} \text{AIC}_k^0$,

$\text{diff}_J^{max} = \max_j(\text{AIC}^0 - \text{AIC}_j)$.

Cluster importance is scale-free, and in some sense it is a normalized measure of cluster cohesion.

Cluster goodness

The goodness measure for regular cluster C_j is defined as the weighted average Silhouette coefficient over all starting sub-clusters in regular cluster C_j, that is,

$$Goodness_j = \frac{\sum_{l(i)=c_j} N_i s_i}{\sum_{l(i)=c_j} N_i}. \tag{72}$$

We can also map $Goodness_j$ into an interestingness score using equation (69).

Cluster goodness is also scale-free, and it is a measure of balancing cluster cohesion and cluster separation.

11.4.2. Outlier Clusters Ranking

For each outlier cluster, we have the following measures: cluster size, outlier strength. Each of the measures can be used to rank outlier clusters, so as to find the most interesting ones.

11.4.3. Outlier Clusters Grouping

Outlier clusters can be grouped by the nearest regular cluster, using probability values.

Appendix A. Map-Reduce Job for Feature Selection

Mapper

Each mapper will handle one data split and use it to build a local CF-tree. The local CF-tree is assigned with a unique key. Notice that if the option of outlier handling is turned on, outliers will not be passed to reducers in case of feature selection.

Let $T_{CF}^{(r)}(key_r)$ be the CF-tree with the key of key_r on data split r ($r = 1, \dots, R$).

The map function is as follows.

```
Inputs:
   −  Data split $r$                          // $r = 1, \dots, R$
   −  $key_r$                                 // $r = 1, \dots, R$

   <Parameter settings>
   −  MainMemory                          // Default 80*1024 bytes
   −  OutlierHandling                     // {on, off}, default on
   −  OutlierHandlingDiskSpace            // Default 20% of MainMemory
   −  OutlierQualification                // Default 10 cases
   −  DelayedSplit                        // {on, off}, default on
   −  DelayedSplitDiskSpace               // Default 10% of MainMemory
   −  Adjustment                          // Default 0.01
   −  DistanceMeasure                     // {Log-likelihood, Euclidean}, default
                                          // Log-likelihood
   −  InitialThreshold                    // Default 0
   −  NonLeafNodeBranchingFactor          // Default 8
   −  LeafNodeBranchingFactor             // Default 8
   −  MaxTreeHeight                       // Default 3
Outputs:
   −  $T_{CF}^{(r)}(key_r)$
Procedure:
1. Build a CF-tree on data split $r$ based on specified features and settings;
2. Assign $key_r$ to the CF-tree;
3. Export $T_{CF}^{(r)}(key_r)$;
```

Reducer

Each reducer can handle several keys. For each key, it first pours together all CF-trees which have the same key. Then it builds approximate clustering solutions iteratively in order to find the most influential features. The selected features will be passed to the controller.

Let $F^*(key_r)$ be the set of features produced for the key of key_r, $r = 1, \dots, R$.

The reduce function for each key is as follows.

```
Inputs:
  –  $T_{CF}^{(r)}(key_r)$

  <Parameter settings>
  –  Adjustment                  // Default 0.01
  –  DistanceMeasure             // {Log-likelihood, Euclidean}, default Log-
                                 // likelihood
  –  AutoClustering              // {on, off}, default on
  –  MaximumClusterNumber        // Default 15
  –  MinimumClusterNumber        // Default 2
  –  FixedClusterNumber          // Default 5
  –  ClusteringCriterion         // {BIC, AIC}, default BIC
  –  AutoClusteringMethod        // {information criterion, distance jump,
                                 // maximum, minimum}, default minimum
Outputs:
  –  $F^*(key_r)$
Procedure:
1. Let $F(key_r)$ be the set of all available features;
2. With all leaf entries in CF-tree $T_{CF}^{(r)}(key_r)$ and using features $F(key_r)$, perform
   matrix based HAC to get an approximate cluster solution S0. Suppose the
   number of approximate final clusters is $J^*$, which is determined automatically
   or using a fixed one depending on the settings;
   Compute importance for each feature in $F(key_r)$;
   // Importance values should not be truncated
   Compute I(S0), the information criterion of S0;
3. Let $F^*(key_r) = F(key_r)$ and $I_{\text{ref}}$ =I(S0);
   Find $F_\alpha(key_r)$, the set of features in $F(key_r)$ with non-positive importance;
   Let $\tilde{F}(key_r) = F(key_r) - F_\alpha(key_r)$;
4. With all leaf entries in CF-tree $T_{CF}^{(r)}(key_r)$ and using features $\tilde{F}(key_r)$, perform
   matrix based HAC to get a new solution S1 with fixed $J^*$;
   Compute I(S1), the information criterion of S1;
   Compute the information of all discarded features I($F(key_r) - \tilde{F}(key_r)$),
   determined by S1, as $\sum_{k \in F(key_r) - \tilde{F}(key_r)} \text{BIC}_k$ , or $\sum_{k \in F(key_r) - \tilde{F}(key_r)} \text{AIC}_k$ , depending on the
   setting, where
         $\text{BIC}_k = \begin{cases} \sum_{j=1}^{J} N_{jk} \ln(\hat{\sigma}_{jk}^2 + \Delta_k) + 2J\ln(N), & \text{if feature } k \text{ is continuous} \\ 2\sum_{j=1}^{J} N'_{jk} \hat{E}_{jk} + J(L_k - 1)\ln(N), & \text{if feature } k \text{ is categorical} \end{cases}$;
         $\text{AIC}_k = \begin{cases} \sum_{j=1}^{J} N_{jk} \ln(\hat{\sigma}_{jk}^2 + \Delta_k) + 4J, & \text{if feature } k \text{ is continuous} \\ 2\sum_{j=1}^{J} N'_{jk} \hat{E}_{jk} + 2J(L_k - 1), & \text{if feature } k \text{ is categorical} \end{cases}$;
    // Though the discarded features are not used to build S1, their
    // statistics are still available in CFs of final clusters in S1.
5. While (I(S1)+I($F(key_r) - \tilde{F}(key_r)$)>$I_{\text{ref}}$){
      Find the most important feature $k$ in $F_\alpha(key_r)$;
      Let $\tilde{F}(key_r) = \tilde{F}(key_r) + \{k\}$, and $F_\alpha(key_r) = F_\alpha(key_r) - \{k\}$;
      With all leaf entries in CF-tree $T_{CF}^{(r)}(key_r)$ and using features $\tilde{F}(key_r)$,
      perform matrix based HAC to get a new solution S1 with fixed $J^*$;
      Compute I(S1), the information criterion of S1;
      Compute I($F(key_r) - \tilde{F}(key_r)$), the information of all discarded features;
   }
   Let $F^*(key_r) = \tilde{F}(key_r)$ and $I_{\text{ref}}$ = I(S1)+I($F(key_r) - \tilde{F}(key_r)$);
6. While ($\tilde{F}(key_r)$ is not empty){
        Find the most unimportant feature $k$ in $\tilde{F}(key_r)$;
        Let $\tilde{F}(key_r) = \tilde{F}(key_r) - \{k\}$;
        If ($F(key_r) - F_\alpha(key_r) - \{k\}$ is empty), break;
        With all leaf entries in CF-tree $T_{CF}^{(r)}(key_r)$ and using features
        $F(key_r) - F_\alpha(key_r) - \{k\}$, perform matrix based HAC to get a new solution S1
        with fixed $J^*$;
        Compute I(S1), the information criterion of S1;
        Compute I($F_\alpha(key_r) + \{k\}$), the information of all discarded features;
        If (I(S1)+ I($F_\alpha(key_r) + \{k\}$)<=$I_{\text{ref}}$){
            Let $F^*(key_r) = F^*(key_r) - \{k\}$;
            Let $F_\alpha(key_r) = F_\alpha(key_r) + \{k\}$;
            Let $I_{\text{ref}} = \text{I(S1)} + \text{I}(F_\alpha(key_r) + \{k\})$;
            Let $\tilde{F}(key_r) = \tilde{F}(key_r) - \{k\}$;
        }
    }
7. Export $F^*(key_r)$;
```

Controller

The controller pours together all sets of features produced by reducers, and selects those features which appear frequently. The selected features will be used in the next map-reduce job to build the final clustering solution.

The controller runs the following procedure.

```
Inputs:
   <Parameter settings>
   -  MinFrequency                // Default 50%
Outputs:
   -  $F^*$                       // Set of selected features
Procedure:
1. Let $\beta$= MinFrequency, and $F^*$ be empty;
2. Launch a map-reduce job, and get $F^*(key_r)$, for $r = 1, \ldots, R$ from the reducers;
3. Compute $F = \bigcup_{r=1}^{R} F^*(key_r)$;
4. For each feature in $F$,
      If the occurring frequency is larger than $R * \beta$, add the feature into $F^*$;
5. Export $F^*$;
```

Appendix B. Map-Reduce Job for Distributed Clustering

Mapper

Each mapper will handle one data split and use it to build a local CF-tree.

Local outlier candidates and the local CF-tree will be distributed to a certain reducer. This is achieved by assigning them a key, which is randomly selected from the key set $\{key_1, \dots, key_Q\}$. The number of keys Q is computed by equation (28).

For convenience, in the following we call leaf entries as pre-clusters. Let $T_{CF}^{(r)}(key_i)$ and $S_{out}^{(r)}(key_i)$ be the CF-tree and the set of outliers, respectively, with the key of key_i ($i = 1, \dots, Q$), on data split r ($r = 1, \dots, R$).

The map function is as follows.

```
Inputs:
   –  Data split $r$                          // $r = 1, \dots, R$
   –  $key_i$                                 // $i = 1, \dots, Q$

   <Parameter settings>
   –  MainMemory                             // Default 80*1024 bytes
   –  OutlierHandling                        // {on, off}, default on
   –  OutlierHandlingDiskSpace               // Default 20% of MainMemory
   –  OutlierQualification                   // Default 10 cases
   –  DelayedSplit                           // {on, off}, default on
   –  DelayedSplitDiskSpace                  // Default 10% of MainMemory
   –  Adjustment                             // Default 0.01
   –  DistanceMeasure                        // {Log-likelihood, Euclidean}, default
                                             // Log-likelihood
   –  InitialThreshold                       // Default 0
   –  NonLeafNodeBranchingFactor             // Default 8
   –  LeafNodeBranchingFactor                // Default 8
   –  MaxTreeHeight                          // Default 3
Outputs:
   –  $T^{(r)}(key_i)$                        // Tightness threshold
   –  $T_{CF}^{(r)}(key_i)$
   –  $S_{out}^{(r)}(key_i)$
Procedure:
1. Build a CF-tree on data split $r$ based on specified features and settings;
2. If (DelayedSplit='on'),
       Absorb cases in disk space $S_1$ with tree rebuilding if necessary;
2. If (OutlierHandling='on'),{
       Absorb entries in disk space $S_2$ without tree rebuilding;
       Check the final CF-tree for outliers;
       Mark the identified outliers and remaining entries in disk space $S_2$ as
       local outlier candidates;
   }
3. Assign $key_i$ to the CF-tree and the set of outlier candidates;
4. Export $T^{(r)}(key_i)$, $T_{CF}^{(r)}(key_i)$, and $S_{out}^{(r)}(key_i)$;
```

Reducer

Each reducer can handle several keys. For each key, it first pours together all CF-trees which have the same key. Then with all leaf entries in the involved CF-trees, it performs a series of CF-tree based HACs to get a specified number of sub-clusters. Finally, the sub-clusters are passed to the controller. The number of sub-clusters produced for each key is computed by equation (29).

Let $\Omega(key_i)$ be the set of data split indices r whose key is key_i, $S_{sub}(key_i)$ and $S_{out}(key_i)$ be the set of sub-clusters and the set of outliers, respectively, produced for the key of key_i, $i = 1, \dots, Q$.

The reduce function for each key is as follows.

```
Inputs:
   –  $T^{(r)}(key_i)$, $r \in \Omega(key_i)$
   –  $T_{CF}^{(r)}(key_i)$, $r \in \Omega(key_i)$
   –  $S_{out}^{(r)}(key_i)$, $r \in \Omega(key_i)$

   <Parameter settings>
   –  OutlierHandling           // {on, off}, default on
   –  Adjustment                // Default 0.01
   –  DistanceMeasure           // {Log-likelihood, Euclidean}, default Log-
                                // likelihood
   –  NumSubClusters            // Number of sub-clusters produced for each key
   –  MinSubClusters            // Minimum sub-clusters produced for each key
                                // default 500
   –  MaximumDataPoitsCFHAC     // Maximum data points for HAC, default 50,000
Outputs:
   –  $S_{sub}(key_i)$
   –  $S_{out}(key_i)$
Procedure:
1. Let $J_1$= NumSubClusters, $C_{min}$= MinSubClusters, and $D_1$= MaximumDataPoitsCFHAC;
2. Compute $T(key_i) = max\{T^{(r)}(key_i), r \in \Omega(key_i)\}$;
3. Compute $S_{CF}(key_i) = \left\{T_{CF}^{(r)}(key_i) | r \in \Omega(key_i)\right\}$;
4. If OutlierHandling is 'on',{
       Compute $S_{out}(key_i) = \cup_{r \in \Omega(key_i)} S_{out}^{(r)}(key_i)$;
       For each member in $S_{out}(key_i)$,{
           Find the closest leaf entry in the set of CF-trees $S_{CF}(key_i)$;
           If the closest leaf entry can absorb the outlier member without
           violating the threshold requirement $T(key_i)$, then merge them, and
           update$S_{out}(key_i)$ and the involved CF-tree;
       }
   }
5. Let $c_1$ be the total number of leaf entries in $S_{CF}(key_i)$;
   While $c_1 > D_1$,{
       Divide the set of CF-trees $S_{CF}(key_i)$ randomly into $c_2$ groups,
       where $c_2 = \lceil c_1/D_1 \rceil$;
       For each group which has a larger number of leaf entries than $c_3$, perform
        CF-tree based HAC to get $c_3$ leaf entries, where $c_3 = \lfloor max(D_1/c_2, C_{min}) \rfloor$;
       Update $S_{CF}(key_i)$ with new CF-trees produced in the above step;
       Compute the total number of remaining leaf entries $c_1$;
   }
6. With the set of CF-trees $S_{CF}(key_i)$, perform CF-tree based HAC to get a set of $J_1$
   sub-clusters, i.e. $S_{sub}(key_i)$;
7. Export $S_{sub}(key_i)$ and $S_{out}(key_i)$;
```

Controller

The controller pours together all sub-clusters produced by reducers, and performs matrix based HAC to get the final clusters. It identifies outlier clusters as well if the option of outlier handling is turned on. Moreover, it computes model evaluation measures, and derives insights and interestingness from the clustering results.

The controller runs the following procedure.

```
Inputs:
   <Parameter settings>
   -  MainMemory                    // Default 80*1024 bytes
   -  OutlierHandling               // {on, off}, default on
   -  OutlierHandlingDiskSpace      // Default 20% of MainMemory
   -  OutlierQualification          // Default 10 cases
   -  ExtremeOutlierClusters        // Default 20
   -  DelayedSplit                  // {on, off}, default on
   -  DelayedSplitDiskSpace         // Default 10% of MainMemory
   -  Adjustment                    // Default 0.01
   -  DistanceMeasure               // {Log-likelihood, Euclidean}, default
                                    // Log-likelihood
   -  InitialThreshold              // Default 0
   -  NonLeafNodeBranchingFactor    // Default 8
   -  LeafNodeBranchingFactor       // Default 8
   -  MaximumTreeHeight             // Default 3
   -  AutoClustering                // {on, off}, default on
   -  MaximumClusterNumber          // Default 15
   -  MinimumClusterNumber          // Default 2
   -  FixedClusterNumber            // Default 5
   -  ClusteringCriterion           // {BIC, AIC}, default BIC
   -  AutoClusteringMethod          // {information criterion, distance jump,
                                    // maximum, minimum}, default minimum
   -  MinSubClusters                // Minimum sub-clusters produced for each key,
                                    // default 500
   -  MaxDataPoitsCFHAC             // Maximum data points for CF-tree based HAC,
                                    // default 50,000
   -  MaxDataPoitsMatrixHAC         // Maximum data points for matrix based HAC,
                                    // default 5,000
Outputs:
   -  PMML
   -  StatXML
Procedure:
1. Let $C_{min}$ = MinSubClusters, $D_1$ = MaximumDataPoitsCFHAC, and $D_2$ =
   MaximumDataPoitsMatrixHAC;
2. Compute the number of keys
```

```
   NumKeys = $Q = \lceil min(R * S/D_1, D_2/C_{min}) \rceil$;
   // Each mapper is assigned a key which is selected randomly from the $Q$ keys
3. Compute the number of sub-clusters produced for each key
   NumSubClusters = $\lfloor min(R * S, D_2)/Q \rfloor$;
4. Launch a map-reduce job, and get $S_{sub}(key_i)$ and $S_{out}(key_i)$, for $i = 1, \dots, Q$;
5. Compute $S_{sub} = \bigcup_{i=1}^{Q} S_{sub}(key_i)$;
6. Perform matrix based HAC on $S_{sub}$ to get the set of final regular clusters $S_{sub}^{*}$,
   where the number of final clusters is determined automatically or using a
   fixed one depending on the settings;
7. If OutlierHandling is ‘on’, perform the steps from 2) to 7) in Step 3 in
    section 9.3;
8. Compute model evaluation measures, insights, and interestingness;
9. Export the clustering model in PMML, and other statistics in StatXML;
```

Implementation notes:

- The general procedure of the controller consists of both the controller procedure in appendix A and that in appendix B.

Appendix C. Procedure for MonotoneCubicInterpolation( )

$f(x) = MonotoneCubicInterpolation(S_t, I_t, x)$,

where

x is the input statistic that characterizes fields or field pairs in particular aspects (for example, distribution), association strength, etc. Its value range must be bounded below, and it must have a monotonically increasing relationship with the interestingness score threshold values. If the two conditions are not met, a conversion (e.g. $x = -x'$ or $x = |x|$, etc) should be carried out.

S_t is a set of distinct threshold values for the input statistics, which have been accepted and commonly used by expert users to interpret the statistics. The positive infinity (+∞) is included if the input statistic is not bounded from above.

I_t is a set of distinct threshold values for the interestingness scores that S_t corresponds to. The threshold values must be between 0 and 1.

The size of S_t and I_t must be the same. There are at least two values in S_t excluding positive infinity (+∞).

Pre-processing

Let $\{x_k\} = sorted(S_t)$ such that $x_1 < \cdots < x_n$, where n is the number of values in S_t.

Let $\{y\} = sorted(I_t)$ such that $y_1 < \cdots < y_n$.

Condition A: There are more than two threshold values for input statistics, and they are all finite numbers

Preparing for cubic interpolation

The following steps should be taken for preparing a cubic interpolation function construction.

Step 1, compute the slopes of the secant lines between successive points.

$$\Delta_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

for $k = 1, \cdots, n-1$.

Step 2, Initialize the tangents at every data point as the average of the secants,

$$m_k = \frac{\Delta_{k-1} + \Delta_k}{2}$$

for $k = 2, \cdots, n-1$; these may be updated in further steps. For the endpoints, use one-sided differences: $m_1 = \Delta_1$ and $m_n = \Delta_{n-1}$.

Step 3, let $\alpha_k = m_k / \Delta_k$ and $\beta_k = m_{k+1} / \Delta_k$ for $k = 1, \cdots, n-1$.

If α or β are computed to be zero, then the input data points are not strictly monotone. In such cases, piecewise monotone curves can still be generated by choosing $m_k = m_{k+1} = 0$, although global strict monotonicity is not possible.

Step 4, update $\mathbf{m_k}$

If $\alpha^2 + \beta^2 > 9$, then set $m_k = \tau_k \alpha_k \Delta_k$ and $m_{k+1} = \tau_k \beta_k \Delta_k$ where $\tau_k = \frac{3}{\sqrt{\alpha^2+\beta^2}}$.

Note:

1. Only one pass of the algorithm is required.

2. For $k = 1, \cdots, n-1$, if $\Delta_k = 0$ (if two successive $y_k = y_{k+1}$ are equal), then set $m_k = m_{k+1} = 0$, as the spline connecting these points must be flat to preserve monotonicity. Ignore step 3 and 4 for those k.

Cubic interpolation

After the preprocessing, evaluation of the interpolated spline is equivalent to cubic Hermite spline, using the data x_k, y_k, and m_k for $k = 1,...,n$.

To evaluate x in the range $[x_k, x_{k+1}]$ for $k = 1,...,n$-1, calculate

$\mathrm{h} = \mathrm{x_{k+1}} - \mathrm{x_k}$ and $\mathrm{t} = \frac{\mathrm{x}-\mathrm{x_k}}{\mathrm{h}}$

then the interpolant is

$$\mathrm{f(x)} = \mathrm{y_k h_{00}(t)} + \mathrm{h} * \mathrm{m_k h_{10}(t)} + \mathrm{y_{k+1} h_{01}(t)} + \mathrm{h} * \mathrm{m_{k+1} h_{11}(t)}$$

where hii(t) are the basis functions for the cubic Hermite spline.

$h_{00}(t)$	$2t^3 - 3t^2 + 1$
$h_{10}(t)$	$t^3 - 2t^2 + t$
$h_{01}(t)$	$-2t^3 + 3t^2$
$h_{11}(t)$	$t^3 - t^2$

<u>Condition B: There are two threshold values for input statistics</u>

As we have clarified in the beginning that there are at least two values in S_t excluding positive infinity (+∞), they must be both finite numbers when there are only two threshold values.

In this case the mapping function is a straight line connecting (x_1, y_1) and (x_2, y_2).

$$\mathrm{f(x)} = \mathbf{y_1} + (\mathbf{y_2} - \mathbf{y_1}) \frac{\mathbf{x} - \mathbf{x_1}}{\mathbf{x_2} - \mathbf{x_1}}$$

Condition C: Threshold values include infinity

Note that there are at least two values in S_t excluding positive infinity (+∞). Take the last three statistic threshold values and threshold values for the interestingness scores from the sorted lists, we have three pairs of data (x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1}) and $(+\infty, y_n)$.

An exponential function

$$\mathrm{f(x) = a - be^{-cx}}$$

can be defined by the pairs, where

$$\mathrm{a} = \mathrm{y_n}$$

$$\mathrm{b} = \sqrt[(\mathrm{x_{n-1}} - \mathrm{x_{n-2}})]{(\mathrm{y_n} - \mathrm{y_{n-2}})^{\mathrm{x_{n-1}}} \Big/ (\mathrm{y_n} - \mathrm{y_{n-1}})^{\mathrm{x_{n-2}}}}$$

$$\mathrm{c} = \frac{1}{\mathrm{x_{n-1}} - \mathrm{x_{n-2}}} \ln \frac{\mathrm{y_n} - \mathrm{y_{n-2}}}{\mathrm{y_n} - \mathrm{y_{n-1}}}$$

If $n = 3$, which means there are only two distinct values in S_t excluding positive infinity (+∞), the exponential function is employed for evaluating x in the range $[x_1, +\infty)$.

Otherwise, the exponential function is for evaluating x in the range $[x_{n-1}, +\infty)$. To evaluate x in the range $[x_1, x_{n-1})$, use procedures under "condition A: There are more than two threshold values for input statistics, and they are all finite numbers" with data set $\{x_1, \cdots, x_{n'}\}$ and $\{y_1, \cdots, y_{n'}\}$ where $n^{'} = n - 1$. To insure a smooth transition to the exponential function, the tangent $m_{n'}$ at data point $x_{n'}$ is given as

$$m_{n'} = \frac{\mathrm{d(a - be^{-cx})}}{\mathrm{dx}}\bigg|_{\mathrm{x=x_{n'}}} = bce^{-cx_{n'}}$$

again

$$\mathrm{a} = \mathrm{y_n}$$

$$\mathrm{b} = \sqrt[(\mathrm{x_{n-1}} - \mathrm{x_{n-2}})]{(\mathrm{y_n} - \mathrm{y_{n-2}})^{\mathrm{x_{n-1}}} \Big/ (\mathrm{y_n} - \mathrm{y_{n-1}})^{\mathrm{x_{n-2}}}}$$

$$\mathrm{c} = \frac{1}{\mathrm{x_{n-1}} - \mathrm{x_{n-2}}} \ln \frac{\mathrm{y_n} - \mathrm{y_{n-2}}}{\mathrm{y_n} - \mathrm{y_{n-1}}}$$

References

[1] Chiu, T. (2000a). mBIRCH Clustering Algorithm (Phase 1 – Preclustering). *IBM SPSS Internal Document*.

[2] Chiu, T., Fang, D., Chen, J., Wang, Y., and Jeris, C. (2001). A Robust and Scalable Clustering Algorithm for Mixed Type Attributes in Large Database Environment. *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*, 263.

[3] Chiu, T. (1999b). Hierarchical Agglomerative Clustering Algorithm. *IBM SPSS Internal Document*.

[4] Chiu, T. (2004). Algorithm Design: Enhancements of Two-Step Clustering. *IBM SPSS Internal Document*.

[5] Fang, D. (2000). Auto-Cluster in SPSS Clustering Component. *IBM SPSS Internal Document*.

[6] Xu, J. (2011). ADD – Interestingness and Insights. *IBM SPSS Internal Document*

[7] Zhang, T., Ramakrishnon, R., and Livny M. (1996). BIRCH: An Efficient Data Clustering Method for Very Large Databases. *Proceedings of the ACM SIGMOD conference on Management of Data*, p. 103-114, Montreal, Canada.

[8] Kaufman, L., and Rousseeuw, P., J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. *Wiley Series in Probability and Statistics*. John Wiley and Sons, New York.