IBM SPSS Modeler 16 Algorithms Guide

IBM®

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

5   Bayesian Networks Algorithms   41

10 CHAID Algorithms

11 Cluster Evaluation Algorithms

12 COXREG Algorithms

16 Factor Analysis/PCA Algorithms 145

17 Feature Selection Algorithm 159

28 OPTIMAL BINNING Algorithms   319

29 Predictor Importance Algorithms   325

30 QUEST Algorithms   329

36 TwoStep Cluster Algorithms   393

Appendix

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.

3

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, …, 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 $1 \leq k_{anomaly} \leq K+1$ 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 $h = 1, \ldots, H$, 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, average$(GDI)_h = 0$, but the GDI between case *i* and cluster *h* is nonzero; that is, $GDI(i, h) \neq 0$. 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,

$$
\begin{cases}
\dfrac{GDI(i,h)}{\frac{1}{(n-n_h)}\Sigma_{s=1,\neq h}^{H}\,n_s \cdot average(GDI)_s} & \text{if } \dfrac{1}{(n-n_h)}\displaystyle\sum_{s=1,\neq h}^{H} n_s \cdot average(GDI)_s \neq 0 \\[2ex]
\dfrac{n/H}{n_h} & \text{otherwise}
\end{cases}
$$

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 $pct_{anomaly}$ % (or alternatively, the top $n_{anomaly}$) gives the anomaly list, subject to the restriction that cases with an anomaly index less than or equal to $cutpoint_{anomaly}$ 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 $k_{anomaly}$ 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 $\geq s_{\min}$ 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	$\sum_{i=1}^{n} f_i I(x_i \text{ is not missing})$, where I (expression) is the indicator function taking value 1 when the expression is true, 0 otherwise.
W_X	$\sum_{i=1}^{n} f_i w_i I(x_i \text{ is not missing})$
N_{XY}	$\sum_{i=1}^{n} f_i I(x_i \text{ and } y_i \text{ are not missing})$
W_{XY}	$\sum_{i=1}^{n} f_i w_i I(x_i \text{ and } y_i \text{ are not missing})$
$\overline{x}$	The mean of variable X, $\frac{1}{W_X} \sum_{i=1}^{n} f_i w_i x_i I(x_i \text{ is not missing})$
M_X^r	$\sum_{i=1}^{n} f_i w_i (x_i - \overline{x})^r$
$\overline{x}_y$	$\frac{1}{W_{XY}} \sum_{i=1}^{n} f_i w_i x_i I(x_i \text{ and } y_i \text{ are not missing})$
M_{XY}	$\sum_{i=1}^{n} f_i w_i (x_i - \overline{x}_y)(y_i - \overline{y}_x)$

A note on missing values

Listwise deletion is used in the following sections:

- "Univariate Statistics Collection " on p. 17
- "Basic Variable Screening " on p. 19
- "Measurement Level Recasting " on p. 20
- "Missing Value Handling " on p. 21
- "Outlier Identification and Handling " on p. 20
- "Continuous Predictor Transformations " on p. 22
- "Target Handling " on p. 23
- "Reordering Categories " on p. 28
- "Unsupervised Merge " on p. 33

Pairwise deletion is used in the following sections:

- "Bivariate Statistics Collection " on p. 25
- "Supervised Merge " on p. 28
- "Supervised Binning " on p. 34
- "Feature Selection and Construction " on p. 35
- "Predictive Power " on p. 38

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: $N_X^{missing} = \sum_{i=1}^{n} f_i I\,(x_i \text{ is missing})$
- 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: $c_i = \sum_{j=1}^{n} f_j I\,(x_j = s_i)$
- 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 $Median\,(X) = \min\left\{s_i : \frac{cc_i}{N_X} \geq 0.5\right\}$, 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:

$$N_X^{(j)} = N_X^{(j-1)} + f_j I\left(x_j \text{ is not missing}\right)$$

$$W_X^{(j)} = W_X^{(j-1)} + f_j w_i I\left(x_j \text{ is not missing}\right)$$

$$v_j = \frac{f_j w_j}{W_X^{(j)}}\left(x_j - \overline{x}^{(j-1)}\right)$$

$$\overline{x}^{(j)} = \overline{x}^{(j-1)} + v_j$$

$$M_X^{2(j)} = M_X^{2(j-1)} + \frac{W_X^{(j)} W_X^{(j-1)}}{f_j w_j} v_j^2$$

$$M_X^{3(j)} = M_X^{3(i-1)} - 3v_j M_X^{2(j-1)} + \frac{W_X^{(j)} W_X^{(j-1)}}{(f_j w_j)^2}\left(W_X^{(j)} - 2f_j w_j\right) v_j^3$$

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 $\sqrt{w_i}\left(x_i - \overline{x}_{robust}\right) < -cutoff \times sd_{robust}$ then replace x_i by $\overline{x}_{robust} - cutoff \times sd_{robust}/\sqrt{w_i}$, and if $\sqrt{w_i}\left(x_i - \overline{x}_{robust}\right) > cutoff \times sd_{robust}$ 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 $\left(\overline{x} + (l-1) \times sd, \overline{x} + r \times sd\right]$ 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 $A_{I_i} = M_{I_i}^2 + W_{I_i}\left(\overline{x}_{robust} - \overline{x}_{I_i}\right)^2$.

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: $skew\left(x'\right) = skew\left(x\right)$

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: $skew\left(x^{'}\right) = skew\left(x\right)$

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:

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

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 $c = \min\left(Y\right) - 1$.

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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. 30 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, $N_i. = 0$ 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. 30 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 $X \le i_1$, the second new category will contain all original categories such that $i_1 < X \le i_2$,…, 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. 17, 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:

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

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

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

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

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:

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

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

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

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

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

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

where

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

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

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. $\alpha_i^{(k+1)} = \dfrac{N_{i.}}{\sum_j \overline{w}_{ij}^{-1} \beta_j^{(k)}} = \alpha_i^{(k)} \dfrac{N_{i.}}{\sum_j m_{ij}^{(k)}}$

3. $\beta_j^{(k+1)} = \dfrac{N_{.j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k+1)}}$

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 $\hat{\alpha}_i, \hat{\beta}_j, \hat{\gamma}_i$ and hence $\hat{m}_{ij}$ 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. $\alpha_i^{(k+1)} = \dfrac{N_{.j}}{\sum_j \overline{w}_{ij}^{-1} \beta_j^{(k)} \left(\gamma_i^{(k)} \right)^{(s_j - \overline{s})}} = \alpha_i^{(k)} \dfrac{N_{i.}}{\sum_j m_{ij}^{(k)}}$

3. $\beta_j^{(k+1)} = \dfrac{N_{\cdot j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k+1)} \left(\gamma_i^{(k)} \right)^{(s_j - \overline{s})}}$

4. $m_{ij}^* = \overline{w}_{ij}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)} \left(\gamma_i^{(k)} \right)^{(s_j - \overline{s})}, \ G_i = 1 + \dfrac{\sum_j (s_j - \overline{s}) \left(N_{ij} - m_{ij}^* \right)}{\sum_j (s_j - \overline{s})^2 m_{ij}^*}$

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 $\hat{\alpha}_i, \hat{\beta}_j, \hat{\gamma}_i, \hat{m}_{ij}$. 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 ith 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. 17, 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. 17. 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. 25, 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

$$R = \sqrt{2(J-1)F_{1-\alpha}(J-1, N_{XY} - J)} \text{ and } C = MS \times \sqrt{\frac{\sum_{i=1}^{J} 1/W_{\cdot i}}{J}}, MS = \sqrt{\frac{\sum_{i=1}^{J} M_{\cdot i}^2}{N_{XY} - J}}.$$

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 $|\overline{x}_{\cdot(q)} - \overline{x}_{\cdot(i)}| \leq c_{critical}$, then $\{\overline{x}_{\cdot(i)}, \cdots, \overline{x}_{\cdot(q)}\}$ 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 $A_{\cdot(k)} = M_{\cdot(k)}^2 + W_{\cdot(k)} (\overline{x}_{\cdot(i,q)} - \overline{x}_{\cdot(k)})^2$, 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 ith and $(i+1)$th homogeneous subsets are computed as $cut_i = \overline{x}_{\cdot(i)}^* + \frac{sd_{\cdot(i)}^* + \epsilon}{(sd_{\cdot(i)}^* + sd_{\cdot(i+1)}^* + 2\epsilon)} (\overline{x}_{\cdot(i+1)}^* - \overline{x}_{\cdot(i)}^*)$.

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:

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

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. 36 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. 36.

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. 36 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 $t(N_{XY} - 2)$ 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 $N_{XY} \leq 2$, 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 $X_1, X_2, \cdots, X_m$:

$$Z = a_1 X_1 + a_2 X_2 + \cdots + a_m X_m$$

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 $c_{ZX} = \sum_{i=1}^{m} a_i c_{X_i X}$, and $c_{ZZ} = \sum_{i=1}^{m} a_i^2 c_{X_i X_i} + 2 \sum_{i \neq j} a_i a_j c_{X_i X_j}$.

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

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

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 $N_{XY} \leq 3$, 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. 37 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. 25.

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 $S_e = \sum_{i=1}^{I} M_{i.}^2$ 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. 37, 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 *i*th predictor
π_i	The parent set of the *i*th predictor besides target Y. For TAN models, its size is ≤ 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}	$\Pr\left(X_i = x_i^k \mid (\pi_i, Y) = (\pi_i, Y)^j\right)$
θ_{Y_i}	$\Pr\left(Y = Y_i\right)$
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.
$X_i \rightarrow X_j$	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, $r_i = |X_i|$
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. 47. 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 ith target category Y_i. These probabilities are calculated as

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

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

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

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.
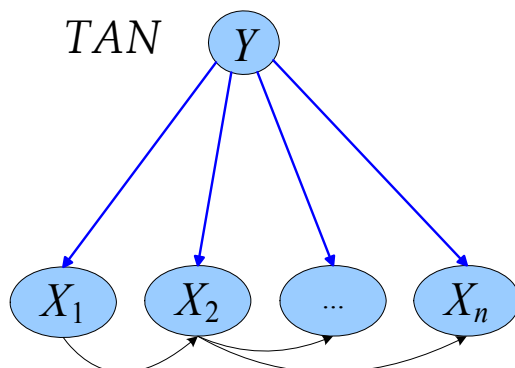
TAN Classifier Learning Procedure

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 $1 \leq i \leq n$.

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 jth value. We use N_{ijk} to denote the number of records in D for which (π_i, Y) take its jth value and for which X_i takes its kth 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 $N_{Y_i}^0$ and N_{ijk}^0 denote corresponding Dirichlet distribution parameters such that $N^0 = \sum_i N_{Y_i}^0$ and $N_{ij}^0 = \sum_k N_{ijk}^0$. Upon observing the dataset *D*, we obtain Dirichlet posterior distributions with the following sets of parameters:

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

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 $\hat{\theta}_{Y_i}^P$ ($1 \leq i \leq |Y|$) and $\hat{\theta}_{ijk}^P$ ($1 \leq k \leq r_i$), $1 \leq i \leq n, 1 \leq j \leq q_i$ using uninformative Dirichlet priors $N_{Y_i}^0 = \frac{2}{|Y|}$ and $N_{ijk}^{'0} = \frac{2}{r_i \cdot q_i}$.

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 $O\left(x_i, y_j\right)$ be the observed count of cases that have $X = x_i$ and $Y = y_j$, and $E\left(x_i, y_j\right)$ 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

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

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

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

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

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

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:

$$\chi^2\left(X, Y | S\right) = \sum_{k} \chi^2\left(X, Y | S = s_k\right)$$

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

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

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

Likelihood Ratio Test

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

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

or equivalently,

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

The conditional version of the G^2 independence test is

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

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

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

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

In the following parts of this document, we use $I(\cdot)$ to uniformly represent the *p*-value of whichever test is applied. If $I(X, Y) > \alpha$, we say variable X and Y are independent, and if $I(X, Y|S) > \alpha$, 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 $I(X_i, X_j)$ for each variable pair in G. If $I(X_i, X_j) > \alpha$, 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 $I(X_i, X_j|S) > \alpha$. 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 $X_j \notin S_{X_i X_j}$

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

$$X_i \quad - \quad X_j \quad - \quad X_k$$
$$\searrow \quad | \quad \swarrow$$
$$X_l$$
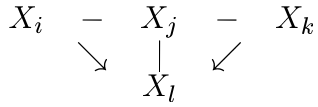
are updated so that $X_j \rightarrow X_l$

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 θ_{ijk} ($1 \leq i \leq n, 1 \leq j \leq q_i, 1 \leq k \leq r_i$) 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 θ_{ijk} ($1 \leq k \leq r_i$), $1 \leq i \leq n, 1 \leq j \leq q_i$ (Heckerman et al., 1999). Let N_{ijk}^0 denote corresponding Dirichlet distributed parameters such that $N_{ij}^0 = \sum_k N_{ijk}^0$. Upon observing the dataset D, we obtain Dirichlet posterior distributions with the following sets of parameters:

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

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 θ_{ijk} $(1 \leq k \leq r_i)$, $1 \leq i \leq n$, $1 \leq j \leq q_i$ 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 ith target category Y_i is calculated as $\Pr(Y = Y_i | \mathbf{X} = \mathbf{x})$. The target category with the highest posterior probability is the predicted category for this case, $Y(\mathbf{x})$, is predicted by

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

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 ith variable in X_{Ch}. The score for each category of Y is computed by:

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

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

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

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

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

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 $(\hat{p}_{j+1} \cdot (r - c)) - ((1 - \hat{p}_{j+1}) \cdot c) \leq 0$

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)
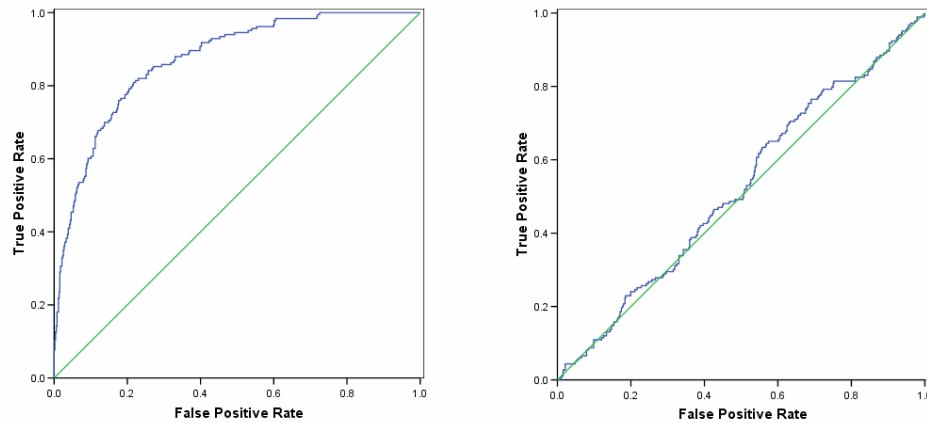
The AUC is computed by identifying segments as unique combinations of predictor values that determine subsets of records which all have the same predicted probability of the target value. The s segments defined by a given model's predictors are sorted in descending order of predicted probability, and 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 $maxSupport < \sigma_i$, 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*$(v) < \sigma_n$, 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. 59) 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 $\Rightarrow$ 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. 59.

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 ith 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

Sex	Employment	Response	Frequency
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.

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. 68.

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. 71.) 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.

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 $N_{f,j}(t)$ is the sum of frequency weights for records in category j for node t, and $N_f(t)$ 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 $p_f(t)$ 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 $N_{f,j}(t)$ 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 $N_{f,j}$ 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

$$p(s^* \approx s_X | t) = \begin{cases} \displaystyle\sum_j \frac{\pi(j) N_{w,j}(s^* \approx s_X, t)}{N_{w,j}(X^* \cap X)} & \text{if } Y \text{ is categorical} \\[2ex] \frac{N_w(s^* \approx s_X, t)}{N_w(X^* \cap X)} & \text{if } Y \text{ is continuous} \end{cases}$$

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 $\pi(j)$ 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 $1 - \frac{1}{k}$, 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 $\Phi(s, t)$ 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 $\Phi(s, t)$.

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. 68.

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

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

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

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

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

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

The split s is chosen to maximize the value of $\Phi(s,t)$.

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 Ns 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 $\pi'(j)$ is the adjusted prior for category j, and $\pi(j)$ 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:

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

$R(T)$ 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 ($\alpha = 0$), 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 $T(\alpha)$, where $T(\alpha)$ 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 ($\alpha = 0$). 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 $\alpha > \alpha'$. 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 $g(t)$:

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

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 $\alpha_1 = 0$ 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

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

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,

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

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, $N_{w,j}(t)$ 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:

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

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. 67.

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. 81.

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. 83.

   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 $\alpha_{split-merge}$, 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. 83.

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

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

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 $F(I - 1, N_{\mathrm{f}} - I)$ is a random variable following an F-distribution with $(I - 1)$ and $(N_{\mathrm{f}} - I)$ 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 $(x_{\mathrm{n}} = i, y_{\mathrm{n}} = j)$ from the independence model as described below. The corresponding p value is calculated as $p = \mathrm{Pr}\left(\chi_d^2 > X^2\right)$, 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 $p = \mathrm{Pr}\left(\chi_d^2 > G^2\right)$

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. $\alpha_i^{(k+1)} = \dfrac{n_{i.}}{\sum_J \overline{w}_{ij}^{-1} \beta_j^{(k)}} = \alpha_i^{(k)} \dfrac{n_{i.}}{\sum_j m_{ij}^{(k)}}$.

3. $\beta_j^{(k+1)} = \dfrac{n_{.j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k+1)}}$

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 $\hat{\alpha}_i$, $\hat{\beta}_j$, $\hat{\gamma}_i$, and hence $\hat{m}_{ij}$ 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. $\alpha_i^{(k+1)} = \dfrac{n_{\cdot j}}{\sum_j \overline{w}_{ij}^{-1} \beta_j^{(k)} \left(\gamma_i^{(k)}\right)^{(s_j - \overline{s})}} = \alpha_i^{(k)} \dfrac{n_{i\cdot}}{\sum_j m_{ij}^{(k)}}$

3. $\beta_j^{(k+1)} = \dfrac{n_{\cdot j}}{\sum_i \overline{w}_{ij}^{-1} \alpha_i^{(k)} \left(\gamma_i^{(k)}\right)^{(s_j - \overline{s})}}$

4. $m_{ij}^* = \overline{w}_{ij}^{-1} \alpha_i^{(k+1)} \beta_j^{(k+1)} \left(\gamma_i^{(k)}\right)^{(s_j - \overline{s})}, \quad G_i = 1 + \dfrac{\sum_j (s_j - \overline{s})\left(n_{ij} - m_{ij}^*\right)}{\sum_j (s_j - \overline{s})^2 m_{ij}^*}$

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 $\hat{\alpha}_i$, $\hat{\beta}_j$, $\hat{\gamma}_i$, and $\hat{m}_{ij}$. 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 $2 \leq r < I$,

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

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, $f_j(t)$ 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

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

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.

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

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

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,

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

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 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:

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

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. 87.

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 sth 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:

$$(B - A) / \max(A, B)$$

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

$$VI_i = \frac{-\log_{10}(sig_i)}{\max_{j \in \Omega}(-\log_{10}(sig_j))}$$

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 $\hat{N}_{ij} = N_{i.}N_{.j}/N(X)$.

- If $N(X) = 0$, the importance is set to be undefined or unknown;
- If $N_{i.} = 0$, subtract one from I for each such category to obtain I';
- If $N_{.j} = 0$, 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$\{F(J-1, N-J) > F\}$,

where

$$F = \frac{\sum\limits_{j=1}^{J} N_j \left(\overline{x}_j - \overline{\overline{x}}\right)^2 / (J-1)}{\sum\limits_{j=1}^{J} (N_j - 1)\, s_j / (N - J)}$$

- 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 = $1 - \text{Prob}\{|T(d)| \leq |t|\}$.

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 jth 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

$$h(t|\mathbf{x}) = \frac{f(t|\mathbf{x})}{S(t|\mathbf{x})}$$

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 $\mathbf{x}$ 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 jth 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

$$D_{\beta_r} = \frac{\partial l}{\partial \beta_r} = \sum_{j=1}^{m} \sum_{i=1}^{k_j} \left(S_{ji}^{(r)} - d_{ji} \frac{\displaystyle\sum_{l \in R_{ji}} w_l x_{lr} e^{\mathbf{x}'_l \beta}}{\displaystyle\sum_{l \in R_{ji}} w_l e^{\mathbf{x}'_l \beta}} \right), \quad r = 1, \ldots, p$$

$S_{ji}^{(r)}$ is the rth 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,

$$\mathrm{BCON} = \left| \frac{\delta}{\text{parameter estimate for previous iteration}} \right|$$

- 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 $\hat{\beta} = \left(\hat{\beta}_1, \ldots, \hat{\beta}_p \right)'$ 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

$$\begin{aligned}
\mathbf{I}_{rs} &= -E \frac{\partial^2}{\partial \beta_r \partial \beta_s} \ln L \\
&= \sum_{j=1}^{m} \sum_{i=1}^{k_j} d_{ji} \left[\frac{\displaystyle\sum_{l \in R_{ji}} w_l x_{ls} x_{lr} e^{\mathbf{x}'_l \beta}}{\displaystyle\sum_{l \in R_{ji}} w_l e^{\mathbf{x}'_l \beta}} - \frac{\left(\displaystyle\sum_{l \in R_{ji}} w_l x_{lr} e^{\mathbf{x}'_l \beta} \right) \left(\displaystyle\sum_{l \in R_{ji}} w_l x_{ls} e^{\mathbf{x}'_l \beta} \right)}{\left(\displaystyle\sum_{l \in R_{ji}} w_l e^{\mathbf{x}'_l \beta} \right)^2} \right]
\end{aligned}$$

We can also write **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 lth diagonal element $v_{ll}(t_{ji})$ defined by

$$v_{ll}(t_{ji}) = p_l(t_{ji})w_l - (w_l p_l(t_{ji}))^2$$

$$p_l(t_{ji}) = \frac{\exp\left(\mathbf{x}^{'}{}_l\hat{\beta}\right)}{\sum_{h \in R_{ji}} w_h \exp\left(\mathbf{x}^{'}{}_h\hat{\beta}\right)}$$

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.

$$\hat{\alpha}_i = \left[1 - \frac{w_i \exp\left(\mathbf{x}'_i \hat{\beta}\right)}{\displaystyle\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l \hat{\beta}\right)} \right]^{\exp\left(-\mathbf{x}'_i \overline{\beta}\right)}$$

- 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

$$\hat{\alpha}_i = \exp\left(\frac{-d_i}{\displaystyle\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l \beta\right)} \right)$$

    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 $\mathbf{a}$ is a $p \times 1$ vector with the jth element defined by

$$\sum_{t_i < t} |D_i| \frac{\displaystyle\sum_{l \in R_i} w_l x_{lj} \exp\left(\mathbf{x}'_l \hat{\beta}\right)}{\left(\displaystyle\sum_{l \in R_i} w_l \exp\left(\mathbf{x}'_l \hat{\beta}\right) \right)^2}$$

and $\mathbf{I}$ 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" in Chapter 23 on p. 259. 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 $\mathbf{I}$ 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 $\mathbf{I}$ 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

$$-2 \big(l \big(\mathbf{b}_{(i)} \big) - l(full) \big)$$

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 $\mathbf{u}$ and $\mathbf{I}$ 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

$$R = \left[\frac{\text{Wald}-2}{-2 \text{ log-likelihood for the intial model}} \right]^{1/2} \times \text{sign of MPLE}$$

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

$$R = \left[\frac{\text{Wald}-2*\text{df}}{-2 \text{ log-likelihood for the intial model}} \right]^{1/2}$$

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 $\mathbf{x}$,

$$\hat{S}(t|\mathbf{x}) = \left[\hat{S}_0(t)\right]^{\exp\left(\mathbf{x}'\hat{\beta}\right)}$$

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. 107.

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 ith 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 mth field (column) for the nth 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 nth record.
A	Collection of all attributes (fields) of X.
A_j	The jth attribute of X.
R	A collection of rules to extend a preceding rule list.
R_k	The kth 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 $L = \varnothing$, add L_{null} to L.

▶ $T = \varnothing$.

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. 113.

▶ 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 ith 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 mth field (column) for the nth 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 nth record.
A	Collection of all attributes (fields) of X.
A_j	The jth 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. 114.
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 $R = \emptyset$, add R_{all} to R.

▶ $T = \emptyset$.

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 $S = \emptyset$, 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 $R = R^*$, $T = \emptyset$, 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 ith category in the list of categories C.
$X_{n,c}$	The value of the split field (attribute) for the nth record (row) of X.
Y	The target field in X.
Y_n	The value of the target field for the nth 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, $S_{L,R} = \{C_i | C_i \in C, 1 \le L \le i \le R \le M\}$.
$\left(p_{S_{L,R}}^-, p_{S_{L,R}}^+\right)$	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.

$$PeakSet = \{C_i | C_i \in C, 0 \le i = I \le M\}$$

where I is a positive integer satisfying the conditions

$$P_I > P_{(I-1)}$$

$$P_I = P_{(I+l)}, 0 \le l \le L - I$$

$$P_L > P_{(L+1)}$$

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

$$Size\left(S_{L,R}\right) > Max(gs_{\min}, d, Max(g \cdot Size(parent)))$$

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. 116.

■ 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

$$\overline{X}_{ij} = \left(\sum_{k=1}^{m_j} f_{jk} X_{ijk} \right)/n_j \qquad \text{(variable } i \text{ in group } j)$$

$$\overline{X}_{i\bullet} = \left(\sum_{j=1}^{g} \sum_{k=1}^{m_j} f_{jk} X_{ijk} \right)/n \quad \text{(variable } i)$$

119

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 Λ 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 $\mathbf{W}$ 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, $\mathbf{W}$ 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 $q{\times}q$. 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

$$
\mathrm{TOL}_i = \begin{cases} 0 & \text{if } w_{ii} = 0 \\ w_{ii}^*/w_{ii} & \text{if variable } i \text{ is not in the analysis and } w_{ii} \neq 0 \\ -1/(w_{ii}^* w_{ii}) & \text{if variable } i \text{ is in the analysis and } w_{ii} \neq 0. \end{cases}
$$

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 $\mathbf{V}$ 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

$$\left(\mathbf{L}^{-1}(\mathbf{T} - \mathbf{W})\mathbf{U}^{-1} - \lambda\mathbf{I}\right)(\mathbf{U}\mathbf{V}) = 0$$

is solved using tridiagonalization and the QL method. The result is m eigenvalues, where $m = \min(q, g - 1)$ and corresponding orthonormal eigenvectors, $\mathbf{U}\mathbf{V}$. 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

$$\frac{\frac{100\lambda_k}{m}}{\sum_{k=1}^{m}\lambda_k}$$

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 $\mathbf{D}$ is computed as

$$\mathbf{D} = \mathbf{S}_{11}\mathbf{V}$$

where

$$\mathbf{S}\text{=diag}\left(\sqrt{w_{11}}, \sqrt{w_{22}}, \ldots, \sqrt{w_{pp}}\right)$$

$\mathbf{S}_{11}$= partition containing the first q rows and columns of $\mathbf{S}$

$\mathbf{V}$ 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 $\mathbf{S}$ and $\mathbf{W}$ 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 $\mathbf{X}$ is the $1 \times q$ vector of discriminating variables for the case, the $1 \times m$ 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. $P(\mathbf{X}|\mathbf{G})$, labeled as $P(\mathbf{D}{>}d|\mathbf{G}{=}g)$ in the output, is the significance level of such a χ_j^2.

■ The classification, or posterior probability, is

$$P\left(\mathbf{G}_j|\mathbf{X}\right) = \frac{P_j|\mathbf{D}_j|^{-1/2}e^{-\chi_j^2/2}}{\displaystyle\sum_{j=1}^{g} P_j|\mathbf{D}_j|^{-1/2}e^{-\chi_j^2/2}}$$

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

$$g_j = \log P_j - \tfrac{1}{2}\left(\log|\mathbf{D}_j| + \chi_j^2\right)$$

$$P\left(\mathbf{G}_j|\mathbf{X}\right) = \begin{cases} \dfrac{\exp\left(g_j - \max_j g_j\right)}{\displaystyle\sum_{j=1}^{g} \exp\left(g_j - \max_j g_j\right)} & \text{if } g_j - \max_j g_j > -46 \\[2em] 0 & \text{otherwise} \end{cases}$$

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 jth group

$$M_{\underset{\sim}{j}} = \frac{1}{n_j}\sum_{k=1}^{m_j} f_{jk}\underset{\sim}{X_{jk}}$$

| $\underset{\sim}{M_{jk}}$ | Sample mean of jth group excluding point $\underset{\sim}{X_{jk}}$ |

$$M_{\underset{\sim}{jk}} = \frac{1}{n_j - f_{jk}}\sum_{\substack{l=1 \\ l \neq k}}^{m_j} f_{jl}\underset{\sim}{X_{jl}}$$

Σ	Polled sample covariance matrix
Σ_j	Sample covariance matrix of jth group
Σ_{jk}	Polled sample covariance matrix without point $\underset{\sim}{X_{jk}}$
Σ_{jk}^{-1}	

$$= \frac{n-g-f_{jk}}{n-g}\left(\Sigma^{-1} + \frac{n_j \Sigma_j^{-1}\left(\underset{\sim}{X_{jk}} - \underset{\sim}{M_j}\right)\left(\underset{\sim}{X_{jk}} - \underset{\sim}{M_j}\right)^T \Sigma_j^{-1}}{\left(n_j - f_{jk}\right)\left(n_j - g\right) - n_j\left(\underset{\sim}{X_{jk}} - \underset{\sim}{M_j}\right)^T \Sigma_j^{-1}\left(\underset{\sim}{X_{jk}} - \underset{\sim}{M_j}\right)}\right)$$

| $d_0^2\left(\underset{\sim}{a}, \underset{\sim}{b}\right)$ | $\left(\underset{\sim}{a} - \underset{\sim}{b}\right)^T \Sigma_{jk}^{-1}\left(\underset{\sim}{a} - \underset{\sim}{b}\right)^T$ |

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 $(\log(P_i) - d_0^2\left(\underset{\sim}{X_{jk}}, \underset{\sim}{M_i}\right)/2 > \log(P_j) - d_0^2\left(\underset{\sim}{X_{jk}}, \underset{\sim}{M_{jk}}\right)/2)$, 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. 136.
- For more information, see the topic "Bagging and Boosting Algorithms" on p. 131.
- For more information, see the topic "Ensembling model scores algorithms" on p. 142.

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 kth record.
y_k	Target value for the kth record.
f_k	Frequency weight for the kth record.
w_k	Analysis weight for the kth 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.
$T^m (\cdot)$	The model built on the mth bootstrap sample.
f_k^m	Simulated frequency weight for the kth record of the mth bootstrap sample.
w_k^m	Updated analysis weight for the kth record of the mth bootstrap sample.
$\hat{y}_k^m = T^m (X_k)$	Predicted target value of the kth record by the mth model.
$P_{l_i}^m (X_k)$	For a categorical target, the probability that the kth record belongs to category l_i, i=1, ..., C, in model m.
$II (\pi)$	For any condition π, $II (\pi)$ 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:

$$f_{mk}^* = \begin{cases} rv.binom\left(N, \frac{f_k}{N}\right) & k = 1 \\[2ex] rv.binom\left(N - \Sigma_{i=1}^{k-1} f_{mi}^*, \ \dfrac{f_k}{N - \Sigma_{i=1}^{k-1} f_i}\right) & \text{otherwise} \end{cases}$$

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

$$\hat{y}_k = \tfrac{1}{M} \sum_{m=1}^{M} \hat{y}_k^m$$

■ Median

Sort $\hat{y}_k^m$ and relabel them $\hat{y}_{(1)} \leq ... \leq \hat{y}_{(M)}$

$$\hat{y}_k = \left\{ \begin{array}{ll} \hat{y}_{\left(\frac{M+1}{2}\right)} & \text{if } M \text{ is odd} \\[2ex] \frac{1}{2}\left(\hat{y}_{\left(\frac{M}{2}\right)} + \hat{y}_{\left(\frac{M}{2}+1\right)}\right) & \text{if } M \text{ is even} \end{array} \right\}$$

Scoring a Categorical Target

■ Voting

$$\hat{y}_k = arg\max_{l_i \in \Omega} \frac{1}{|M_{l_i}|} \sum_{m \in M_{l_i}} P_{l_i}^m(X_k)$$

$$\hat{p}_{\hat{y}_k} = \frac{1}{|M_{\hat{y}_k}|} \sum_{m \in M_{\hat{y}_k}} P_{\hat{y}_k}^m(X_k)$$

where $\Omega = \{arg\max_{l_i} |M_{l_i}|\}$

■ Highest probability

$$\hat{y}_k = arg\max_{l_i}\left(\max_m\left(P_{l_i}^m(X_k)\right)\right)$$

$$\hat{p}_{\hat{y}_k} = \max_m\left(P_{\hat{y}_k}^m(X_k)\right)$$

■ Highest mean probability

$$\hat{y}_k = arg\max_{l_i} \frac{1}{M} \sum_{m=1}^{M} P_{l_i}^m(X_k)$$

$$\hat{p}_{\hat{y}_k} = \frac{1}{M} \sum_{m=1}^{M} P_{\hat{y}_k}^m(X_k)$$

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

$$R^2 = 1 - \frac{\Sigma_{k=1}^{K} f_k \left(y_k - \hat{y}_k \right)^2}{\Sigma_{k=1}^{K} f_k \left(y_k - \overline{y} \right)^2}$$

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 $L \left(y_k \right) = \sum_{m=1}^{M} II \left(y_k = \hat{y}_k^m \right)$.

For continuous targets, diversity is

$$D = \frac{\sum_{k=1}^{K} f_k \left[\frac{1}{M \left(M - 1 \right)} \sum_{m=1}^{M} \sum_{n=1,n \neq m}^{M} \left(y_k - \hat{y}_k^m \right) \left(\hat{y}_k^n - y_k \right) \right]}{\Sigma_{k=1}^{K} f_k \left(y_k - \overline{y}_k \right)^2}$$

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, $\omega^m = \log\left(\dfrac{1 - \sum_{k=1}^K L_k w_k^m f_k}{\sum_{k=1}^K L_k w_k^m f_k} \right)$

   where $L_k = \dfrac{abs(\hat{y}_k^m - y_k)}{\max_k \left(abs(\hat{y}_k^m - y_k) \right)}$.

3. Set weights for the next base model .

   $w_k^{m+1} = \dfrac{a_k^{m+1}}{\Sigma_{i=1}^K a_i^{m+1} f_i}$

   where $a_k^{m+1} = w_k^m \left(\dfrac{\sum_{k=1}^K L_k w_k^m f_k}{1 - \sum_{k=1}^K L_k w_k^m f_k} \right)^{1 - L_k}$. 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:

   $f_k^{m+1} = \begin{cases} rv.binom\left(N, w_k^{m+1} f_k\right) & k = 1 \\ rv.binom\left(N - \Sigma_{i=1}^{k-1} f_k^{m+1}, \dfrac{w_k^{m+1} f_k}{1 - \Sigma_{i=1}^{k-1} w_k^{m+1} f_i}\right) & \text{otherwise} \end{cases}$

   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

$$\sum_{m=1}^{i-1} \omega^m < \frac{1}{2} \sum_{m=1}^{M} \omega^m \le \sum_{m=1}^{i} \omega^m$$

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 $err^m = \sum_{k=1}^{K} w_k^m f_k II(y_k \ne \hat{y}_k^m)$.

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 $a_k^{m+1} = w_k^m \exp(\omega^m II(y_k \ne \hat{y}_k^m))$. 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 $\hat{y}_k^m = arg \max_{l_i} P_{l_i}^m(X_k)$.

The ensemble predicted value is then $\hat{y}_k = arg \max_{l_i} \sum_{m=1}^{M} \omega^m II(\hat{y}_k^m == l_i)$. Ties are resolved at random.

The ensemble predicted probability is $\hat{p}_{\hat{y}_k} = \sum\limits_{m \in M_{\hat{y}_k}} \dfrac{\omega^m}{\sum\limits_{i \in M_{\hat{y}_k}} \omega^i} P^m_{\hat{y}_k}(X_k).$

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

$$\frac{1}{N} \sum_{k=1}^{K} f_k II (y_k == \hat{y}_k)$$

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 $\hat{c}(x_i)$ be the label predicted by the given model. Then the testing error is estimated as:

Categorical target. $E = \dfrac{1}{\sum\limits_{i=1}^{|T|} f_i} \sum\limits_{i=1}^{|T|} \left(f_i \cdot I\left(c_i \neq \hat{c}(x_i) \right) \right)$

Continuous target. $E = \dfrac{1}{\sum\limits_{i=1}^{|T|} f_i} \sum\limits_{i=1}^{|T|} \left(f_i \cdot |y_i - \hat{y}_i| \right)$

Where $I\left(c_i \neq \hat{c}(x_i) \right)$ is 1 if $c_i \neq \hat{c}(x_i)$ 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 ith 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 ith 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 ith 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 ith 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 $\alpha = 0.3$, 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 mth base model's predicted value of case i.

Categorical Target

Voting. Assume that $d_{m,k}$ represents the label output of the mth base model for a given vector of predictor values. $d_{m,k} = 1$ if the label assigned by the mth base model is the kth target category and 0 otherwise. There are total of M base models and K target categories. The majority vote method selects the jth 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 mth 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 kth target category by the mth base model for a given vector of predictor values. The following rules combine the probabilities computed by the base models. The jth category is selected such that it satisfies the corresponding equation.

- Highest probability. $\max_{m=1}^{M}\left(p_{m,j}\right) = \max_{k=1}^{K}\left(\max_{m=1}^{M}\left(p_{m,k}\right)\right)$

- 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:

$$p_i^S = \frac{Exp\,(p_i)}{\sum\limits_{i=1}^{K} Exp\,(p_i)}$$

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 mth 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 mth base model for a given vector of predictor values. $d_{m,k} = 1$ if the label assigned by the mth base model is the kth target category and 0 otherwise. There are total of M base models and K target categories. The majority vote method selects the jth 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 kth target category by the mth base model for a given vector of predictor values. The following rules combine the probabilities computed by the base models. The jth 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.

$$\max_{m=1}^{M} \left(p_{m,j} \right) = \max_{k=1}^{K} (\max_{m=1}^{M} \left(p_{m,k} \right))$$

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 $\Lambda_{m(i)R} = [diag(\Sigma)]^{-\frac{1}{2}} \Lambda_{m(i)}$. 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 sth 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

$$\mathbf{x}_i^1 = \begin{cases} \log\left[\left(1 - \frac{m}{2p}\right)/r^{ii}\right] & \text{for ML and GLS} \\[2mm] \left[\left(1 - \frac{m}{2p}\right)/r^{ii}\right]^{\frac{1}{2}} & \text{for ULS} \end{cases}$$

where m is the number of factors and r^{ii} is the ith 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 $\gamma_1 \leq \gamma_2 \leq ... \leq \gamma_p$ and corresponding eigenvectors $\omega_1, \omega_2,...,\omega_p$ 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

$$\delta_{ij} = \begin{cases} 1 & if \; i = j \\ 0 & if \; i \neq j \end{cases}$$

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,

$$\frac{\partial f}{\partial x_i} = \sum_{k=m+1}^{p} \left(\gamma_k^2 - \gamma_k\right)\omega_{ik}^2$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \delta_{ij}\frac{\partial f}{\partial x_i} + \sum_{k=m+1}^{p} \gamma_k \omega_{ik}\omega_{jk} \left(\sum_{n=1}^{m} \gamma_n \frac{\gamma_k + \gamma_n - 2}{\gamma_k - \gamma_n}\omega_{in}\omega_{jn} + r^{ii} \exp\left(\frac{x_i + x_j}{2}\right) \right)$$

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

$$h_{k(i)} \left(\sum_{j=1}^{m} |\gamma_{j(i)}| \omega_{kj(i)}^2 \right) h_{k(i-1)}$$

The initial values of the communalities, $\mathbf{H}_0$, are

$$h_{io} = \begin{cases} 1 - \frac{1}{r^{ii}} & |\mathbf{R}| \geq 10^{-8} \text{ and all } 0 \leq h_{io} \leq 1 \\ \max_j |r_{ij}| & \text{otherwise} \end{cases}$$

where r^{ii} is the ith 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 ith 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

$$X = \begin{cases} D - \frac{2AB}{n} & \text{Varimax} \\ D - \frac{mAB}{n} & \text{Equamax} \\ D & \text{Quartimax} \end{cases}$$

$$Y = \begin{cases} C - \left(\frac{A^2 - B^2}{n} \right) & \text{Varimax} \\ C - \frac{m(A^2 - B^2)}{2n} & \text{Equamax} \\ C & \text{Quartimax} \end{cases}$$

$$u_{p(i)} = f_{pj(i)}^{*2} - f_{pk(i)}^{*2} \quad v_{p(i)} = 2 f_{pj(i)}^* f_{pk(i)}^* \quad p = 1, ..., n$$

$$A = \sum_{p=1}^{n} u_{p(i)} \qquad B = \sum_{p=1}^{n} v_{p(i)}$$
$$C = \sum_{p=1}^{n} \left[u_{p(i)}^2 - v_{p(i)}^2 \right] \qquad D = \sum_{p=1}^{n} 2u_{p(i)} v_{p(i)}$$

If $|\sin(P)| \leq 10^{-15}$, no rotation is done on the pair of factors.

■ The new rotated factors are

$$\left(\tilde{\underline{\lambda}}_{j(i)}, \tilde{\underline{\lambda}}_{k(i)} \right) = \left(\underline{\lambda}_{j(i)}^*, \underline{\lambda}_{k(i)}^* \right) \begin{vmatrix} \cos(P) & -\sin(P) \\ \sin(P) & \cos(P) \end{vmatrix}$$

where $\underline{\lambda}_{j(i)}^*$ are the last values for factor j calculated in this iteration.

■ The accrued rotation transformation matrix is

$$\left(\tilde{t}_j, \tilde{t}_k \right) = (t_j, t_k) \begin{vmatrix} \cos(P) & -\sin(P) \\ \sin(P) & \cos(P) \end{vmatrix}$$

where t_j and t_k are the last calculated values of the jth and kth 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

$$\tilde{\underline{\lambda}}_j = -\tilde{\underline{\lambda}}_{j(f)}$$

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:

$$s_k = \begin{cases} 1 & \text{if Kaiser} \\ h_k & \text{if no Kaiser} \end{cases} \quad k = 1, ..., n$$

$$u_i = \sum_{j=1}^{n} \lambda_{ji}^{*2} \quad i = 1, ..., m$$

$$v_i = \sum_{j=1}^{n} \lambda_{ji}^{*4}$$

$$x_i = v_i - \left(\frac{\delta}{n}\right) u_i^2$$

$$D = \sum_{i=1}^{m} u_i$$

$$G = \sum_{i=1}^{m} x_i$$

$$H = \sum_{k=1}^{n} s_i^2 - \left(\frac{\delta}{n}\right) D^2$$

$$FO = H - G$$

At each iteration, all possible factor pairs are rotated. For a pair of factors λ_p^* and λ_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

$$\left(\tilde{\lambda}_p^*, \tilde{\lambda}_q^*\right) = \left(\lambda_p^*, \lambda_q^*\right) \begin{vmatrix} t_1 & -a \\ 0 & 1 \end{vmatrix}$$

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

$$\mathbf{R}_{ff} = \mathbf{C} \left(\mathbf{Q}'\mathbf{Q} \right)^{-1} \mathbf{C}'$$

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.

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 $\hat{N}_{ij} = N_{i\cdot}N_{\cdot j}/N$. Under the null hypothesis, Pearson's chi-square converges asymptotically to a chi-square distribution χ^2_d 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 = $\text{Prob}(\chi^2_d > X^2)$, 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 $\hat{N}_{ij} = N_{i\cdot}N_{\cdot j}/N$. Under the null hypothesis, the likelihood ratio chi-square converges asymptotically to a chi-square distribution χ^2_d 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 = $\text{Prob}(\chi^2_d > G^2)$, where

$$G^2 = 2\sum_{i=1}^{I} \sum_{j=1}^{J} G^2_{ij}, \text{ with } G^2_{ij} = \begin{cases} N_{ij} \ln\left(N_{ij}/\hat{N}_{ij} \right) & N_{ij} > 0, \\ 0 & \text{else.} \end{cases}$$

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

$$\lambda(Y|X) = \frac{\sum\limits_{i} \max\limits_{j}(N_{ij}) - \max\limits_{j}(N_{.j})}{N - \max_j(N_{.j})}.$$

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 $Y=j$.
$\bar{x}_j$	The sample mean of predictor X for target class $Y=j$.
s_j^2	The sample variance of predictor X for target class $Y=j$. $$s_j^2 = \sum_{i=1}^{N_j}(x_{ij}-\bar{x}_j)^2/(N_j-1)$$
$\bar{\bar{x}}$	The grand mean of predictor X. $\bar{\bar{x}} = \sum\limits_{j=1}^{J} N_j \bar{x}_j/N$

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

$$
F = \frac{\sum_{j=1}^{J} N_j \left(\overline{x}_j - \overline{\overline{x}} \right)^2 / (J-1)}{\sum_{j=1}^{J} (N_j - 1)\, s_j^2 / (N-J)},
$$

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 jth 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

$$r = \frac{\Sigma_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y})/(N-1)}{\sqrt{s(x)^2 s(y)^2}}.$$

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 ($\mathbf{y}$) or one of the predictor variables ($\mathbf{X}$).
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 $\mathbf{f}$ 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 **η** 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 **η** ; 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 $i=1,\dots,n$. 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 $\mathbf{\Psi} = \mathrm{diag}(f_1\omega_1,\dots f_n\omega_n)$,, 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 $b(\theta)$ and $c(y, \phi/\omega)$ 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 ($V(\mu)$), 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*	$V(\mu)$	Var(*y*)	$V'(\mu)$
Normal	$(-\infty, \infty)$	1	ϕ	0
Inverse Gaussian	$(0, \infty)$	μ^3	$\phi\mu^3$	$3\mu^2$
Gamma	$(0, \infty)$	μ^2	$\phi\mu^2$	2μ
Negative binomial	$0(1)\infty$	$\mu+k\mu^2$	$\mu+k\mu^2$	$1+2k\mu$
Poisson	$0(1)\infty$	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 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 $\mathrm{Var}(y)$ for continuous distributions include the scale parameter ϕ which can be used to scale the relationship of the variance and mean ($\mathrm{Var}(y)$ 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. 184.

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. 184.

Table 18-3
The log-likelihood function for probability distribution

Distribution	ℓ_k and ℓ
Normal	$\ell_k = \sum_{i=1}^{n} -\frac{f_i}{2}\left\{ \frac{\omega_i(y_i - \mu_i)^2}{\phi} + \ln\left(\frac{\phi}{\omega_i}\right) \right\}$ $\ell = \ell_k + \sum_{i=1}^{n} -\frac{f_i}{2}\{\ln(2\pi)\}$
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 Poisson distributions and $y = 0$ or 1 for the binomial distribution, a separate value of the log-likelihood is given. Let $\ell_{k,i}$ be the log-likelihood value for individual case i when $y_i = 0$ 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, $\ell_i = \ell_{k,i}$.

Table 18-4
Log-likelihood

Distribution	$\ell_{k,i}$
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	$\ell_{k,i} = -f_i \frac{\omega_i}{\phi} \mu_i$ if $y_i = 0$
Binomial(m)	$\ell_{k,i} = \begin{cases} f_i \frac{\omega_i}{\phi} \ln\left(1 - \mu_i\right) & \text{if } y_i = 0 \\ f_i \frac{\omega_i}{\phi} \ln\left(\mu_i\right) & \text{if } y_i = 1 \end{cases}$

- $\Gamma(z)$ 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	$\Phi^{-1}(\mu)$, 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 $|\alpha| < 2.2\text{e-}16$, 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, $\mathbf{s}$ is a $p \times 1$ 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. 174, Table 18-2 "Distribution, range and variance of the response, variance function, and its first derivative" on p. 171 and Table 18-6 "The first and second derivatives of link function" on p. 174, respectively.

If the scale parameter ϕ is estimated by the ML method, it is handled by searching for $\ln(\phi)$ since ϕ is required to be greater than zero.

Let $\tau = \ln(\phi)$ so $\phi = \exp(\tau)$, then $\mathbf{s}$ is a $(p+1) \times 1$ vector with the following form

$$\mathbf{s} = \left[\begin{array}{c} \frac{\partial \ell}{\partial \beta} \\ \frac{\partial \ell}{\partial \tau} \end{array} \right]_{(p+1) \times 1} = \left[\begin{array}{c} \frac{1}{\exp(\tau)} \sum_{i=1}^{n} \frac{f_i \omega_i (y_i - \mu_i)}{V(\mu_i) g'(\mu_i)} \cdot x_i \\ \frac{\partial \ell}{\partial \tau} \end{array} \right]$$

where $\partial \ell / \partial \beta$ is the same as the above with ϕ is replaced with $\exp(\tau)$, $\partial \ell / \partial \tau$ 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

$$\hat{\beta} = \left(\sum_{i=1}^{n} f_i \omega_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_i \right)^{-} \left(\sum_{i=1}^{n} f_i \omega_i \mathbf{x}_i^{\mathrm{T}} (y_i - o_i) \right) = \left(\mathbf{X}^{\mathrm{T}} \Psi \mathbf{X} \right)^{-} \left(\mathbf{X}^{\mathrm{T}} \Psi (\mathbf{y} - \mathbf{o}) \right),$$

where $\Psi = \mathrm{diag}(f_1 \omega_1, \dots f_n \omega_n)$ and $(Z)^{-}$ is the generalized inverse of a matrix $\mathbf{Z}$. If the scale parameter ϕ is also estimated by the ML method, the estimate of τ is

$$\hat{\tau} = \ln \left(\hat{\phi} \right) = \ln \left(\frac{1}{N} \sum_{i=1}^{n} f_i \omega_i \left(y_i - \mathbf{x}_i^{\mathrm{T}} \hat{\beta} - o_i \right)^2 \right).$$

Second Derivatives

Let $\mathbf{H}$ be the second derivative (Hessian) matrix. If the scale parameter is not estimated by the ML method, $\mathbf{H}$ is a $p \times p$ 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 $\mathbf{W}$ is an $n \times n$ diagonal matrix. There are two definitions for $\mathbf{W}$ depending on which algorithm is used: $\mathbf{W}_{\mathrm{e}}$ for Fisher scoring and $\mathbf{W}_{\mathrm{o}}$ for Newton-Raphson. The ith 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 ith 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. 171 and Table 18-6 "The first and second derivatives of link function" on p. 174, 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, $\mathbf{H}$ becomes a $(p+1) \times (p+1)$ matrix with the form

$$\mathbf{H} = \left[\begin{array}{cc} \frac{\partial^2 \ell}{\partial \beta \partial \beta^{\mathrm{T}}} & \frac{\partial^2 \ell}{\partial \beta \partial \tau} \\ \frac{\partial^2 \ell}{\partial \tau \partial \beta^{\mathrm{T}}} & \frac{\partial^2 \ell}{\partial \tau^2} \end{array} \right]_{(p+1) \times (p+1)}$$

where $\partial^2 \ell / \partial \beta \partial \tau$ is a $p \times 1$ vector and $\partial^2 \ell / \partial \tau \partial \beta^{\mathrm{T}}$ is a $1 \times p$ vector and the transpose of $\partial^2 \ell / \partial \beta \partial \tau$. For all three continuous distributions:

$$\frac{\partial^2 \ell}{\partial \beta \partial \tau} = \sum_{i=1}^{n} - \frac{f_i \omega_i (y_i - \mu_i)}{\exp(\tau) V(\mu_i) g'(\mu_i)} \cdot \mathbf{x}_i = -\frac{\partial \ell}{\partial \beta}.$$

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	$\frac{\partial^2 \ell}{\partial \tau^2}$
Normal	$\displaystyle\sum_{i=1}^{n} - \frac{f_i \omega_i}{2 \exp(\tau)} (y_i - \mu_i)^2$
Inverse Gaussian	$\displaystyle\sum_{i=1}^{n} - \frac{f_i \omega_i}{2 \exp(\tau) y_i \mu_i^2} (y_i - \mu_i)^2$
Gamma	$\displaystyle\sum_{i=1}^{n} \frac{f_i \omega_i}{\exp(\tau)} \left\{ \ln \left(\frac{\omega_i y_i}{\exp(\tau) \mu_i} \right) + \left(2 - \frac{y_i}{\mu_i} \right) - \psi \left(\frac{\omega_i}{\exp(\tau)} \right) - \frac{\omega_i}{\exp(\tau)} \psi' \left(\frac{\omega_i}{\exp(\tau)} \right) \right\}$

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.
$\epsilon_\ell, \epsilon_{\mathrm{p}}, \epsilon_{\mathbf{H}}$	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,* $\epsilon_\ell, \epsilon_p, \epsilon_H$ 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 $\xi=1$.

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 $\mathbf{H}$. 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 $\{0.5^j : j = 0, \ldots, J-1\}$. 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 $\mathbf{W}_e$ is used to calculate $\mathbf{H}^{(i)}$ if $i \leq K$; $\mathbf{W}_o$ is used to calculate $\mathbf{H}^{(i)}$ if $i > K$.

7. Check if complete or quasi-complete separation of the data is established (see below) if distribution is binomial and the current iteration $i \geq I$. 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 $\tilde{\mu}_i = (y_i m_i + 0.5)/(m_i + 1)$ for a binomial distribution (y_i can be a proportion or 0/1 value) and $\tilde{\mu}_i = y_i$ for a non-binomial distribution. From these derive $\tilde{\eta}_i = g(\tilde{\mu}_i), g'(\tilde{\mu}_i)$ and $V(\tilde{\mu}_i)$. If $\tilde{\eta}_i$ becomes undefined, set $\tilde{\eta}_i = 1$.

2. Calculate the weight matrix $\tilde{W}_e$ with the diagonal element $\tilde{w}_{ei} = \frac{f_i \omega_i}{\phi} \cdot \frac{1}{V(\tilde{\mu}_i)(g'(\tilde{\mu}_i))^2}$, 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 $z_i = (\tilde{\eta}_i - o_i) + (y_i - \tilde{\mu}_i) g'(\tilde{\mu}_i)$ for a binomial distribution and $z_i = (\tilde{\eta}_i - o_i)$ for a non-binomial distribution.

4. Calculate the initial parameter values

$$\beta^{(0)} = \left(X^{T}\tilde{W}_e X\right)^{-1} X^{T}\tilde{W}_e z$$

and

$$\phi^{(0)} = \left(z - X\beta^{(0)}\right)^{T}\tilde{W}_e\left(z - X\beta^{(0)}\right).$$

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 $\mu_v = g^{-1}\left(x_v^{T}\beta + o_v\right).$

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:

Log-likelihood convergence: $\begin{cases} \dfrac{\left| \ell^{(i)} - \ell^{(i-1)} \right|}{\left| \ell^{(i-1)} \right| + 10^{-6}} < \epsilon_\ell \text{ if relative change} \\ \\ \left| \ell^{(i)} - \ell^{(i-1)} \right| < \epsilon_\ell \text{ if absolute change} \end{cases}$.

Parameter convergence: $\begin{cases} \max_j \left(\dfrac{\left| \beta_j^{(i)} - \beta_j^{(i-1)} \right|}{\left| \beta_j^{(i-1)} \right| + 10^{-6}} \right) < \epsilon_p \text{ if relative change} \\ \\ \max_j \left(\left| \beta_j^{(i)} - \beta_j^{(i-1)} \right| \right) < \epsilon_p \text{ if absolute change} \end{cases}$.

Hessian convergence: $\begin{cases} \dfrac{\left(\mathbf{s}^{(i)} \right)^{\mathrm{T}} \left(\mathbf{H}^{(i)} \right)^{-} \left(\mathbf{s}^{(i)} \right)}{\left| \ell^{(v)} \right| + 10^{-6}} < \epsilon_{\mathbf{H}} \text{ if relative change} \\ \\ \left(\mathbf{s}^{(i)} \right)^{\mathrm{T}} \left(\mathbf{H}^{(i)} \right)^{-} \left(\mathbf{s}^{(i)} \right) < \epsilon_{\mathbf{H}} \text{ if absolute change} \end{cases}$.

where $\epsilon_\ell, \epsilon_p$ 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 $\epsilon_{\mathbf{H}} = 1\text{E}{-}4$ 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

$$\Sigma_{\mathrm{m}} = -\mathbf{H}^{-} = -(-\mathbf{X}\mathbf{W}\mathbf{X})^{-}$$

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

$$\Sigma_r = \Sigma_m \left(\sum_{i=1}^{n} \left[\frac{\partial \ell_i}{\partial \beta} \right] \left[\frac{\partial \ell_i}{\partial \beta} \right]^T \right) \Sigma_m = \Sigma_m \left(\sum_{i=1}^{n} f_i \left(\frac{\omega_i (y_i - \mu_i)}{\phi V(\mu_i) g'(\mu_i)} \right)^2 \cdot x_i \cdot x_i^T \right) \Sigma_m$$

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 ith 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. 178. Then the estimate of the scale parameter is $\exp(\hat{\tau})$ and the standard error estimate is $(\exp(\hat{\tau}) \cdot \hat{\sigma}_\tau)$

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 100pth 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 $\exp(\beta_j)$ 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 − α)% 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 $100(1 − α)$% Wald confidence interval for ϕ is

$$\left(\exp\left(\hat{\tau} − z_{1−α/2}\hat{\sigma}_\tau\right), \exp\left(\hat{\tau} + z_{1−α/2}\hat{\sigma}_\tau\right)\right)$$

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 $p = 1 − G(T)$. For example, the *p*-value for the chi-square test of $H_{0i} : \beta_i = 0$ is $p_i = 1 − prob\left(\chi_1^2 \le c_i\right)$.

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 $A = -\left(\frac{\partial^2 \ell}{\partial \tau^2}\right) - \left(-\frac{\partial^2 \ell}{\partial \tau \partial \beta}\mathsf{T}\right)\left(-\frac{\partial^2 \ell}{\partial \beta \partial \beta}\mathsf{T}\right)^{-}\left(-\frac{\partial^2 \ell}{\partial \beta \partial \tau}\right)$ 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. 176 and see Table 18-9"The second derivative functions w.r.t. the scale parameter for probability distributions" on p. 178 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

$$\hat{\alpha} = \frac{\sum\limits_{i=1}^{n} f_i w_i z_i}{\sum\limits_{i=1}^{n} f_i w_i^2} \text{ and } \hat{\sigma}_\alpha = \sqrt{\frac{s_e^2}{\sum\limits_{i=1}^{n} f_i w_i^2}},$$

where $s_e^2 = \frac{1}{N-1}\sum\limits_{i=1}^{n} f_i e_i^2$ 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:

$$D = 2\phi(\ell(\mathbf{y}; \mathbf{y}) - \ell(\hat{\mu}; \mathbf{y})),$$

where $\ell\left(\hat{\mu}; \mathbf{y}\right)$ 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 $\ell\left(\mathbf{y}; \mathbf{y}\right)$ is the log-likelihood function computed by replacing $\hat{\mu}$ with $\mathbf{y}$. 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	$\omega_i\left(y_i - \mu_i\right)^2$
Inverse Gaussian	$\frac{\omega_i}{y_i \mu_i^2}\left(y_i - \mu_i\right)^2$
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 $\mathbf{y}$ 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 $\mathbf{r/m}$ 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	d_i
Negative Binomial	$2\omega_i \frac{\ln\left(1 + k\mu_i\right)}{k}$ if $y_i = 0$
Poisson	$2\omega_i \mu_i$ if $y_i = 0$
Binomial(m)	$\begin{cases} -2\omega_i^* \ln\left(1 - \mu_i\right) \text{ if } y_i = 0 \text{ or } r_i = 0 \\ -2\omega_i^* \ln\left(\mu_i\right) \text{ if } y_i = 1 \text{ or } r_i = m_i \end{cases}$

Pearson Chi-Square

$$\chi^2 = \sum_{i=1}^{n} f_i \gamma_i$$

where $\gamma_i = \frac{\omega_i^*\left(y_i - \mu_i\right)^2}{V\left(\mu_i\right)}$ for the binomial distribution and $\gamma_i = \frac{\omega_i\left(y_i - \mu_i\right)^2}{V\left(\mu_i\right)}$ 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.

$$\hat{\phi} = \frac{D}{N - p_x} \text{ or } \hat{\phi} = \frac{\chi^2}{N - p_x}.$$

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; $m_j = r_{j1} + r_{j0}$.
y_{j1}	The proportion of 1s in the jth subpopulation; $y_{j1} = r_{j1}/m_j$.
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

$$\hat{\phi} = \frac{D}{n_s - p_x} \text{ and } \hat{\phi} = \frac{\chi^2}{n_s - p_x}.$$

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 $d = p_x$ if only β is included; $d = p_x + 1$ 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 $\mathbf{L}_i$ is constructed and H_0: $\mathbf{L}_i\beta = \mathbf{0}$ is tested. Construction of matrix $\mathbf{L}_i$ is based on the generating matrix $\mathbf{H}_\omega = \left(\mathbf{X}^T\mathbf{\Omega}\mathbf{X}\right)^{-}\mathbf{X}^T\mathbf{\Omega}\mathbf{X}$, where $\mathbf{\Omega}$ is the scale weight matrix with *i*th diagonal element ω_i and such that $\mathbf{L}_i\beta$ 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 $\mathbf{L}_i$ is a $r{\times}p$ full row rank hypothesis matrix and $\mathbf{K}$ is a $r{\times}1$ 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 $r_C = rank\left(\mathbf{L\Sigma L}^{\mathrm{T}}\right)$. If $r_C < r$, then $\left(\mathbf{L\Sigma L}^{\mathrm{T}}\right)^{-}$ 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 $\mathbf{L}_i$ and $\mathbf{K}=\mathbf{0}$.

Type III Analysis

Wald statistics. See the discussion of Wald statistics for Type I analysis above. $\mathbf{L}_i$ 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

$$\hat{\sigma}_\eta = \sqrt{x_i^{\mathrm{T}}\mathbf{\Sigma}x_i}$$

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 $100(1-\alpha)$% confidence intervals for the mean can be computed as follows

$$g^{-1}\left(x_i^{\mathrm{T}}\hat{\beta} + o_i \pm z_{1-\alpha/2}\hat{\sigma}_\eta\right)$$

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 $p_{\mathrm{X}} \geq 1$.
K	Number of random effects.
y	$n \times 1$ target vector. The rows are records.
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.
m	$n \times 1$ trials vector for the binomial distribution. All elements are positive integers and $m_{\mathrm{i}} \geq r_{\mathrm{i}}$, $i=1,...,n$.
μ	$n \times 1$ expected target value vector.
η	$n \times 1$ linear predictor vector.
X	$n \times p$ design matrix. The rows represent the records and the columns represent the parameters. The ith row is $\mathrm{x}_i^{\mathrm{T}} = (x_{i1}, \ldots, x_{ip})$, 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.
Z	$n \times r$ design matrix of random effects.
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.
β	$p \times 1$ parameter vector. The first element is the intercept, if there is one.
γ	$r \times 1$ random effect vector.
ω	$n \times 1$ scale weight vector. If an element is less than or equal to 0 or missing, the corresponding record is not used.
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 *k*th random effect
θ_G	covariance parameters of the random effects, $\theta_G = \left[\theta_1^{\mathrm{T}}, \ldots, \theta_K^{\mathrm{T}}\right]^{\mathrm{T}}$
θ_R	covariance parameters of the residuals
$\boldsymbol{\theta}$	$\theta = \left[\theta_G^{\mathrm{T}}, \theta_R^{\mathrm{T}}\right]^{\mathrm{T}} = \left[\theta_1^{\mathrm{T}}, \ldots, \theta_K^{\mathrm{T}}, \theta_R^{\mathrm{T}}\right]^{\mathrm{T}}$
$V_{Y|\gamma}$	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 $(r \times 1)$ vector of random effects which are assumed to be normally distributed with mean 0 and variance matrix **G**, **X** is a $(n \times p)$ design matrix for the fixed effects; **Z** is a $(n \times r)$ 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. 194. The link functions offered are listed in Table 19-3 on p. 195. Different combinations of probability distribution and link function can result in different models.

See "Nominal multinomial distribution " on p. 212 for more information on the nominal multinomial distribution.

See "Ordinal multinomial distribution " on p. 219 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*	$V(\mu)$	Var(*y*)	$V'(\mu)$
Normal	$(-\infty, \infty)$	1	ϕ	0
Inverse Gaussian	$(0, \infty)$	μ^3	$\phi\mu^3$	$3\mu^2$

Distribution	Range of y	$V(\mu)$	Var(y)	$V'(\mu)$
Gamma	$(0,\infty)$	μ^2	$\phi\mu^2$	2μ
Negative binomial	$0(1)\infty$	$\mu+k\mu^2$	$\mu+k\mu^2$	$1+2k\mu$
Poisson	$0(1)\infty$	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	$\ln(\mu)$	$\exp(\eta)$	$\hat{\mu} \geq 0$

Link function	$\eta=g(\mu)$	Inverse $\mu=g^{-1}(\eta)$	Range of $\hat{\mu}$
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) $\begin{cases}\alpha\neq0\\\alpha=0\end{cases}$	$\begin{cases}\mu^{\alpha}\\\ln(\mu)\end{cases}$	$\begin{cases}\eta^{1/\alpha}\\\exp(\eta)\end{cases}$	$\begin{cases}\hat{\mu}\in R \text{ if } \alpha \text{ or} 1/\alpha \text{ is odd integer}\\\hat{\mu}\geq0 \text{ otherwise}\end{cases}$
Log-complement	$\ln(1-\mu)$	$1-\exp(\eta)$	$\hat{\mu}\leq1$
Negative log-log	$-\ln(-\ln(\mu))$	$\exp(-\exp(-\eta))$	$\hat{\mu}\in[0,1]$

Note: In the power link function, if $|\alpha|<$ 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	$\frac{1}{\phi(\Phi^{-1}(\mu))}$, 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) $\begin{cases}\alpha\neq0\\\alpha=0\end{cases}$	$\begin{cases}\alpha\mu^{\alpha-1}\\\frac{1}{\mu}\end{cases}$	$\begin{cases}\Delta\frac{\alpha-1}{\mu}\\-\Delta^2\end{cases}$
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

$$var\,(\mathbf{y}|\gamma) = \mathbf{A}^{1/2}\mathbf{R}\mathbf{A}^{1/2}$$

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, $\mathbf{A} = \mathrm{diag}(V(\mu_i)/\omega_i), i = 1, \ldots, n$. The variance functions, $V(\mu)$, 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. 198.

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 $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}}$, *i*=1,...,*n* with $x_{i1} = 1$ if the model has an intercept. Suppose $\mathbf{x}_i^*$ is the transformation of $\mathbf{x}_i$ then the *j*th entry of $\mathbf{x}_i^*$ is defined as

$$\mathbf{x}_{ij}^* = \frac{x_{ij} - c_j}{s_j}$$

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, $\overline{x}_j = \frac{1}{N} \sum_{i=1}^{n} f_i x_{ij}$ 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 $s_{x_j}^2 = \frac{1}{N-1} \sum_{i=1}^{n} f_i (x_{ij} - \overline{x}_j)^2$. 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

$$\mu \approx \tilde{\mu} + \left(g^{-1}\right)' \left(X\tilde{\beta} + Z\tilde{\gamma} + O\right) \left[X\left(\beta - \tilde{\beta}\right) + Z(\gamma - \tilde{\gamma})\right]$$

where $\left(g^{-1}\right)'\left(\mathbf{X}\tilde{\beta} + Z\tilde{\gamma} + O\right)$ is a diagonal matrix with elements consisting of evaluations of the 1st derivative of g^{-1}. Since $\left(g^{-1}\right)'\left(\mathbf{X}\tilde{\beta} + Z\tilde{\gamma} + O\right) = \left(g'(\tilde{\mu})\right)^{-1}$, this equation can be rearranged as

$$g'(\tilde{\mu})(\mu - \tilde{\mu}) + \mathbf{X}\tilde{\beta} + Z\tilde{\gamma} \approx \mathbf{X}\beta + Z\gamma$$

If we define a pseudo target variable as

$$v \equiv g'(\tilde{\mu})(y - \tilde{\mu}) + \mathbf{X}\tilde{\beta} + Z\tilde{\gamma} = g'(\tilde{\mu})(y - \tilde{\mu}) + g(\tilde{\mu}) - O,$$

then the conditional expectation and variance of v, based on $E\left(y|\gamma\right)$ and $var\left(y|\gamma\right) = A^{1/2}RA^{1/2}$, are

$$E(v|\gamma) = g'(\tilde{\mu})(\mu - \tilde{\mu}) + \mathbf{X}\tilde{\beta} + Z\tilde{\gamma}$$

$$var(v|\gamma) = g'(\tilde{\mu})A_{\tilde{\mu}}^{1/2}RA_{\tilde{\mu}}^{1/2}g'(\tilde{\mu})$$

where $A_{\tilde{\mu}}^{1/2} = \text{diag}\left[(V(\tilde{\mu}_i)/\omega_i)^{1/2}\right], i = 1, \ldots, n$.

Furthermore, we also assume $v|\gamma$ is normally distributed. Then we consider the model of v

$$v = \mathbf{X}\beta + Z\gamma + \varepsilon$$

as a weighted linear mixed model with fixed effects β, random effects $\gamma \sim N(0, G)$, error terms $\varepsilon \sim N\left(0, g'(\tilde{\mu})A_{\tilde{\mu}}^{1/2}RA_{\tilde{\mu}}^{1/2}g'(\tilde{\mu})\right)$, because $var(\varepsilon) = var(v|\gamma)$, and diagonal weight matrix $\tilde{W} = A_{\tilde{\mu}}\left[g'(\tilde{\mu})\right]^{-2}$. Note that the new target v (with $\mathbf{O}$ if an offset variable exists) is a Taylor series approximation of the linked target $g(y)$. The estimation method of unknown parameters of β and θ, which contains all unknowns in $\mathbf{G}$ and $\mathbf{R}$, 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:

$$\ell(\theta; v) = -\frac{1}{2}\ln|V(\theta)| - \frac{1}{2}r(\theta)^\mathrm{T}V(\theta)^{-1}r(\theta) - \frac{N}{2}\ln(2\pi)$$

$$\ell_R(\theta; v) = -\frac{1}{2}\ln|V(\theta)| - \frac{1}{2}r(\theta)^\mathrm{T}V(\theta)^{-1}r(\theta) - \frac{1}{2}\ln\left|X^\mathrm{T}V(\theta)^{-1}X\right| - \frac{N - p_x}{2}\ln(2\pi)$$

where
$V(\theta) = ZG(\theta)Z + \tilde{\mathbf{W}}^{-1/2}R(\theta)\tilde{\mathbf{W}}^{-1/2}, r(\theta) = v - X\left(X^\mathrm{T}V(\theta)^{-1}X\right)^- X^\mathrm{T}V(\theta)^{-1}v = v - X\hat{\beta}, N$
denotes the effective sample size, and p_x denotes the rank of the design matrix of $\mathbf{X}$ or the number of non-redundant parameters in $\mathbf{X}$. 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 $-2\ell(\theta; v)$ or $-2\ell_R(\theta; v)$. Upon obtaining $\hat{\theta}$, estimates for β and γ are computed as

$$\hat{\beta} = \left(X^T V\left(\hat{\theta}\right)^{-1} X \right)^{-} X^T V\left(\hat{\theta}\right)^{-1} v$$

$$\hat{\gamma} = \hat{G} Z^T V\left(\hat{\theta}\right)^{-1} \hat{r}$$

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 $-2\ell(\theta; v)$ and the above equation yields the PL estimation procedure and between $-2\ell_R(\theta; v)$ 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

   $$v = g(\tilde{\mu}) - O + g'(\tilde{\mu})(y - \tilde{\mu}) \text{ and } \tilde{W} = A_{\tilde{\mu}}^{-1}\left[g'(\tilde{\mu})\right]^{-2}.$$

   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

$$\tilde{\mu} = g^{-1}\left(\mathbf{X}\tilde{\beta} + Z\tilde{\gamma} + O\right),$$

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 $2H^{-1}$, where **H** is the Hessian matrix of the objective function ($-2\ell(\theta; v)$ or $-2\ell_R(\theta; v)$) 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 $2H^{-1}$.

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 $100(1 - \alpha)\%$ Wald confidence interval is given, assuming the variance parameter estimate is $\hat{\sigma}^2$ and its standard error is $\text{se}(\hat{\sigma}^2)$ from the corresponding diagonal element of $2H^{-1}$, by

$$\exp\left(\ln\left(\hat{\sigma}^2\right) \pm z_{1-\alpha/2} \cdot \hat{\sigma}^{-2} \cdot \text{se}\left(\hat{\sigma}^2\right)\right)$$

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 $100(1 - \alpha)\%$ Wald confidence interval is given, assuming the correlation parameter estimate is $\hat{\rho}$ and its standard error is $\text{se}(\hat{\rho})$ from the corresponding diagonal element of $2H^{-1}$, by

$$\tanh\left(\tanh^{-1}(\hat{\rho}) \pm z_{1-\alpha/2} \cdot \left(1 - \hat{\rho}^2\right)^{-1} \cdot \text{se}(\hat{\rho})\right)$$

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 $100(1 - \alpha)\%$ Wald confidence interval is simply, assuming the parameter estimate is $\hat{\sigma}_1$ and its standard error is $\text{se}(\hat{\sigma}_1)$ from the corresponding diagonal element of $2H^{-1}$,

$$\left(\hat{\sigma}_1 - z_{1-\alpha/2} \cdot \text{se}(\hat{\sigma}_1)\right), \hat{\sigma}_1 + z_{1-\alpha/2} \cdot \text{se}(\hat{\sigma}_1)\right)$$

The $100(1 - \alpha)\%$ 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

$$\hat{C} = \begin{bmatrix} X^T R^{*-1} X & X^T R^{*-1} Z \\ Z^T R^{*-1} X & Z^T R^{*-1} Z + G\left(\hat{\theta}\right)^{-1} \end{bmatrix}^{-} = \begin{bmatrix} C_{11} & C_{21}^T \\ C_{21} & C_{22} \end{bmatrix}$$

where $R^* \equiv \hat{var}(v|\gamma) = g'(\hat{\mu}) A_{\hat{\mu}}^{1/2} R A_{\hat{\mu}}^{1/2} g'(\hat{\mu})$ is evaluated at the converged estimates and

$$\hat{C}_{11} = \left(\mathbf{X}^T \hat{V}^{-1} \mathbf{X}\right)^{-}$$

$$\hat{C}_{21} = -\hat{G} Z^T \hat{V}^{-1} \mathbf{X} \hat{C}_{11}$$

$$\hat{C}_{22} = \left(Z^T \hat{R}^{-1} Z + \hat{G}^{-1}\right)^{-1} - \hat{C}_{21} \mathbf{X}^T \hat{V}^{-1} Z \hat{G}$$

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 $-2\ell_R^*(\theta; v)$ and $-2\ell_R(\theta; v)$, respectively, then $-2\ell_R(\theta; v)$ can be obtained from $-2\ell_R^*(\theta; v)$ as follows:

$$-2\ell_R(\theta; v) = -2\ell_R^*(\theta; v) - 2\ln|A|$$

Because REPL has the following extra term involved the X matrix

$$
\begin{aligned}
-\tfrac{1}{2}\ln\left|X^{*\mathrm{T}}V(\theta)^{-1}X^*\right| &= -\tfrac{1}{2}\ln\left|(XA)^{\mathrm{T}}V(\theta)^{-1}XA\right| \\
&= -\tfrac{1}{2}\ln\left(\left|A^{\mathrm{T}}\right| \times \left|XV(\theta)^{-1}X\right| \times |A|\right) \\
&= -\tfrac{1}{2}\left(\ln\left|XV(\theta)^{-1}X\right| + \ln|A| + \ln\left|A^{\mathrm{T}}\right|\right) \\
&= -\tfrac{1}{2}\ln\left|XV(\theta)^{-1}X\right| - \ln|A|
\end{aligned}
$$

then $-\tfrac{1}{2}\ln\left|XV(\theta)^{-1}X\right| = -\tfrac{1}{2}\ln\left|X^{*\mathrm{T}}V(\theta)^{-1}X^*\right| + \ln|A|$ and $\ell_R(\theta; v) = \ell_R^*(\theta; v) + \ln|A|$. 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:

$$\hat{\beta} = A\hat{\beta}^*,$$

$$\hat{C}_{11} = A\hat{C}_{11}^*A^{\mathrm{T}},$$

$$\hat{C}_{21} = \hat{C}_{21}^*A^{\mathrm{T}},$$

$$\hat{C}_{22} = \hat{C}_{22}^*.$$

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

$$\Sigma_\mathrm{r} = \Sigma_\mathrm{m} \left(\sum_{j=1}^{S} \mathbf{X}_j^\mathrm{T} \hat{V}_j^{-1} \hat{r}_j \hat{r}_j^\mathrm{T} \hat{V}_j^{-1} \mathbf{X}_j \right) \Sigma_\mathrm{m}$$

where $\hat{r}_j = \mathbf{v}_j - \mathbf{X}_j \hat{\beta}$.

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 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.

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 v degrees of freedom. See "Method for computing degrees of freedom " on p. 209 for details on computing the degrees of freedom.

Wald confidence intervals for estimates in fixed effects and predictions in random effects

The $100(1 - \alpha)\%$ Wald confidence interval for β_i is given by

$$\left(\hat{\beta}_i - t_{v,\alpha/2} \hat{\sigma}_{\beta_i}, \hat{\beta}_i + t_{v,\alpha/2} \hat{\sigma}_{\beta_i} \right)$$

where $t_{v,\alpha/2}$ is the $(1 - \alpha/2)\,100$th percentile of the t_v 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 $100(1 - \alpha)\%$ Wald confidence interval for $\exp(\beta_i)$ is

$$\left(\exp\left(\hat{\beta}_i - t_{v,\alpha/2} \hat{\sigma}_{\beta_i} \right), \exp\left(\hat{\beta}_i + t_{v,\alpha/2} \hat{\sigma}_{\beta_i} \right) \right).$$

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 $\mathbf{L}$ is constructed and H_0: $\mathbf{L_i\beta} = \mathbf{0}$ is tested. Construction of $\mathbf{L}$ and the generating estimable function (GEF) is based on the generating matrix $H_\omega = \left(X^T \Psi X\right)^{-} X^T \Psi X$, where $\Psi = \mathrm{diag}(f_1\omega_1, \ldots f_n\omega_n)$, such that $\mathbf{L_i\beta}$ is estimable; that is, $L_i = L_i H_\omega$. It involves parameters only for the given effect and the effects containing the given effect. For type III analysis, $\mathbf{L}$ 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 $\mathbf{L}$ 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 v. See "Method for computing degrees of freedom " on p. 209 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

$$V(\hat{\mathbf{u}}) = L\Sigma L^{\mathrm{T}}$$

The standard error for the *j*th element of $\hat{\mathbf{u}}$ is the square root of the *j*th diagonal element of $V(\hat{\mathbf{u}})$. 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 $100(1-\alpha)\%$ confidence interval for $u_j, j = 1, \ldots, r,$ is given by

$$\hat{u}_j \pm t_{v^j, \alpha/2} \hat{\sigma}_{u_j}$$

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. 209 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 $\mathbf{C}$ 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:

$$F = \frac{(\mathrm{C}\hat{u})^{\mathrm{T}} \left(\mathrm{C}\mathrm{V}(\hat{u})\mathrm{C}^{\mathrm{T}} \right)^{-} (\mathrm{C}\hat{u})}{r_I}$$

which has an asymptotic F distribution with r_I degrees of freedom, where $r_I = \mathrm{rank}\left(\mathrm{C}\mathrm{V}(\hat{u})\mathrm{C}^{\mathrm{T}} \right)$. See "Method for computing degrees of freedom " on p. 209 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 $\mathbf{c}_i^{\mathrm{T}}$ of matrix $\mathbf{C}$ is also tested separately. The estimate for the ith row is given by $\mathbf{c}_i^{\mathrm{T}}\hat{u}$ and its standard error by $\sqrt{\mathbf{c}_i^{\mathrm{T}}\mathrm{V}(\hat{u})\mathbf{c}_i}$. The corresponding $100(1 - \alpha)\%$ confidence interval is given by

$$\mathbf{c}_i^{\mathrm{T}}\hat{u} \pm t_{v^i, \alpha/2}\hat{\sigma}_{cu_i}$$

The test statistic for $H_0 : \mathbf{c}_i^{\mathrm{T}}\mathbf{u} = 0$ is

$$t_i = \frac{\mathbf{c}_i^{\mathrm{T}}\hat{u}}{\hat{\sigma}_{cu_i}}$$

It has an asymptotic t distribution. See "Method for computing degrees of freedom " on p. 209 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 $r \times r$ 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 jth 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. 196.

The $100(1 - \alpha)$% 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. 206.

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 ith row vector of matrix $\mathbf{C}$. The ith 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 ith 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:

$$p_{(i)}^* = \begin{cases} R p_{(1)} & i = 1 \\ \max\left((R - i + 1)\, p_{(i)}, p_{(i-1)}^*\right) & i \geq 2 \end{cases}$$

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 $x = \dfrac{\ln\left(1 - p_{(i-1)}^*\right)}{\ln\left(1 - p_{(i)}\right)}$.

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 $\mathbf{X}$ is the design matrix of fixed effects.

Satterthwaite's approximation

First perform the spectral decomposition $\mathbf{L\hat{C}L}^T = \mathbf{\Gamma}^T \mathbf{D\Gamma}$ where $\mathbf{\Gamma}$ is an orthogonal matrix of eigenvectors and $\mathbf{D}$ is a diagonal matrix of eigenvalues. If l_m is the mth row of $\mathbf{\Gamma L}$, d_m is the mth eigenvalues and

$$\nu_m = \frac{2d_m}{\mathbf{g_m}\Sigma\left(\hat{\theta}\right)^{-1}\mathbf{g_m}}$$

where $\mathbf{g}_m = \frac{\partial l_m \mathbf{C}l_m^T}{\partial \theta}\big|_{\theta=\hat{\theta}}$ 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

$$\mathbf{x}_i^{\mathrm{T}}\hat{\beta} + \mathbf{z}_i^{\mathrm{T}}\hat{\gamma} + o_i$$

Standard error of the linear predictor

$$\hat{\sigma}_\eta = \sqrt{\mathbf{x}_i^{\mathrm{T}}\Sigma\mathbf{x}_i + \mathbf{z}_i^{\mathrm{T}}\hat{C}_{22}\mathbf{z}_i + 2\mathbf{z}_i^{\mathrm{T}}\hat{C}_{21}\mathbf{x}_i},$$

Predicted value of the mean

$$g^{-1}\left(\mathbf{x}_i^{\mathrm{T}}\hat{\beta} + \mathbf{z}_i^{\mathrm{T}}\hat{\gamma} + o_i\right)$$

For the binomial distribution with 0/1 binary target variable, the predicted category $c(\mathbf{x}_i)$ is

$$c(\mathbf{x}_i) = \begin{cases} 1 \text{ (or sucess)} & \text{if } \hat{\mu}_i \geq 0.5 \\ 0 \text{ (or failure)} & \text{otherwise} \end{cases}.$$

Approximate $100(1-\alpha)\%$ confidence intervals for the mean

$$g^{-1}\left(\mathbf{x}_i^{\mathrm{T}}\hat{\beta} + \mathbf{z}_i^{\mathrm{T}}\hat{\gamma} + o_i \pm t_{v,\alpha/2}\hat{\sigma}_\eta\right)$$

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 $v\hat{a}r(\mathbf{v}|\gamma)$ and $v\hat{a}r(\mathbf{v}|\gamma) = g'(\hat{\mu})A_{\hat{\mu}}^{1/2}\hat{R}A_{\hat{\mu}}^{1/2}g'(\hat{\mu})$ 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 $v\hat{a}r(\mathbf{y}) = A_{\hat{\mu}_m}^{1/2}\hat{R}A_{\hat{\mu}_m}^{1/2}$ 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

$$c\left(j,j'\right) = \sum_{i=1}^n f_i I\left(y_i = j, c\left(x_i\right) = j'\right)$$

where $I\left(\cdot\right)$ is indicator function.

Suppose that $p\left(j,j'\right)$ is the $\left(j,j'\right)^{\text{th}}$ element of the classification table, which is the row percentage, then

$$p_{j,j'} = \left(\cfrac{c\left(j,j'\right)}{\displaystyle\sum_{j'=1}^{J} c\left(j,j'\right)} \right) \times 100\%$$

The percentage of total correct predictions of the model (or "overall percent correct") is

$$p_{total} = \left(\cfrac{\displaystyle\sum_{j=1}^{J} c\left(j,j\right)}{\displaystyle\sum_{j=1}^{J}\sum_{j'=1}^{J} c\left(j,j'\right)} \right) \times 100\%$$

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 sth super subject.
y_{st}	Nominal categorical target for the tth case in the sth 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}, $y_{st} = (y_{st,1}, \cdots, y_{st,J-1})^{\mathrm{T}}$, 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	$y_s = \left(y_{s1}^{\mathrm{T}}, \ldots, y_{sT_s}^{\mathrm{T}} \right)^{\mathrm{T}}, s = 1, \cdots, S.$
y	$y = \left(y_1^{\mathrm{T}}, \cdots, y_S^{\mathrm{T}} \right)^{\mathrm{T}}$
$\pi_{st,j}$	Probability of category j for the tth case in the sth super subject; that is, $\pi_{st,j} = P\left(y_{st} = j\right).$
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 tth case in the sth super subject.
η_{st}	$\eta_{st} = \left(\eta_{st,1}, \cdots, \eta_{st,J-1} \right)^{\mathrm{T}}$
η_s	$\eta_s = \left(\eta_{s1}^{\mathrm{T}}, \ldots, \eta_{sT_s}^{\mathrm{T}} \right)^{\mathrm{T}}, s = 1, \cdots, S$
η	$(n(J-1)) \times 1$ vector of linear predictor. $\eta = \left(\eta_1^{\mathrm{T}}, \cdots, \eta_S^{\mathrm{T}} \right)^{\mathrm{T}}$
x_{st}	$p \times 1$ vector of predictor variables for the tth case in the sth super subject. The first element is 1 if there is an intercept.
X_s	$X_s = \left(I_{J-1} \otimes x_{s1}, \cdots, I_{J-1} \otimes x_{sT_s} \right)^T, s = 1, \cdots, S$
X	$(n(J-1)) \times (J-1)p$ design matrix of fixed effects, $X = \left(X_1^T, \cdots, X_S^T \right)^T$
z_{st}	$r \times 1$ vector of coefficients for the random effect corresponding to the tth case in the sth super subject.
Z_s	$Z_s = \left(I_{J-1} \otimes z_{s1}, \cdots, I_{J-1} \otimes z_{sT_s} \right)^T, s = 1, \cdots, S$
Z	Design matrix of random effects, $Z = \overset{S}{\underset{s=1}{\oplus}} Z_s$, where $\oplus$ is the direct sum of matrices.
O	$n \times 1$ vector of offsets, $O = \left(o_{11}, \cdots, o_{1T_1}, \cdots, o_{ST_S} \right)^T$, where o_{st} is the offset value of the tth case in the sth 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	$p \times 1$ vector of unknown parameters for category j, $\beta_j = \left(\beta_{j1}, \cdots, \beta_{jp} \right)^T, j = 1, \cdots, J$. 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}	$r \times 1$ vector of random effects for category j in the sth super subject, $j = 1, \cdots, J-1$.
γ_s	Random effects for the sth 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 tth case in the sth 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	$n \times 1$ vector of scale weight variable, $\omega = \left(\omega_{11}, \cdots, \omega_{1T_1}, \cdots, \omega_{S1}, \cdots, \omega_{ST_S} \right)^{\mathrm{T}}$.
f_{st}	Frequency weight of the tth case in the sth 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	$n \times 1$ vector of frequency count variable, $f = \left(f_{11}, \cdots, f_{1T_1}, \cdots, f_{S1}, \cdots, f_{ST_S} \right)^{\mathrm{T}}$.
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; $\mathbf{X}$ is the design matrix for fixed effects; $\mathbf{Z}$ is the design matrix for random effects; γ is a vector of random effects which are assumed to be normally distributed with mean $\mathbf{0}$ and variance matrix $\mathbf{G}$; $g\left(.\right)$ is the logit link function such that

$$\eta_{st,j} = g\left(\pi_{st,j}\right) = \log\left(\frac{\pi_{st,j}}{\pi_{st,J}}\right)$$

And its inverse function is

$$\pi_{st,j} = g^{-1}\left(\eta_{st,j}\right) = \begin{cases} \dfrac{\exp\left(\eta_{st,j}\right)}{1 + \displaystyle\sum_{k=1}^{J-1}\exp\left(\eta_{st,k}\right)}, j = 1,\cdots,J-1, \\ \dfrac{1}{1 + \displaystyle\sum_{k=1}^{J-1}\exp\left(\eta_{st,k}\right)}, j = J. \end{cases}$$

The variance of $\mathbf{y}$, conditional on the random effects is

$$Var\left(y|\gamma\right) = A_\mu^{1/2} R A_\mu^{1/2}$$

where $A_\mu = \displaystyle\bigoplus_{s=1}^{S}\bigoplus_{t=1}^{T_s}\left(diag\left(\pi_{st}\right) - \pi_{st}\pi_{st}^{\mathrm{T}}\right)/\omega_{st}$ and $\mathrm{R} = \phi\mathrm{I}$ 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. 198, we can obtain a weighted linear mixed model

$$v = X\beta + Z\gamma + \epsilon$$

where $\mathrm{v} \equiv \mathrm{D}^{-1}(\mathrm{y} - \tilde{\pi}) + g(\tilde{\pi}) - \mathrm{O}^*$ and error terms $\varepsilon \sim N\left(0, \mathrm{D}^{-1}A_{\tilde{\pi}}^{1/2} R A_{\tilde{\pi}}^{1/2}\mathrm{D}^{-1}\right)$ with

$$\mathrm{D} = \bigoplus_{s=1}^{S}\bigoplus_{t=1}^{T_s}\mathrm{D}_{st} = \bigoplus_{s=1}^{S}\bigoplus_{t=1}^{T_s}\frac{\mathrm{d}g^{-1}(\tilde{\eta}_{st})}{\mathrm{d}\tilde{\eta}_{st}} = \bigoplus_{s=1}^{S}\bigoplus_{t=1}^{T_s}\left(diag\left(\tilde{\pi}_{st}\right) - \tilde{\pi}_{st}\tilde{\pi}_{st}^{\mathrm{T}}\right)$$

and

$$A_{\tilde{\mu}} = \bigoplus_{s=1}^{S}\bigoplus_{t=1}^{T_s}\left(diag\left(\tilde{\pi}_{st}\right) - \tilde{\pi}_{st}\tilde{\pi}_{st}^{\mathrm{T}}\right)/\omega_{st}.$$

And block diagonal weight matrix is

$$\tilde{W} = \mathrm{D}A_{\tilde{\mu}}^{-1}\mathrm{D} = \mathop{\oplus}_{s\,=\,1}^{S}\;\mathop{\oplus}_{t\,=\,1}^{T_s}\;\omega_{st}\mathrm{D}_{st}.$$

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:

$$\ell(\theta; v) = -\frac{1}{2}\ln|\mathrm{V}(\theta)| - \frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta) - \frac{N}{2}\ln(2\pi)$$

$$\ell_R(\theta; v) = -\frac{1}{2}\ln|\mathrm{V}(\theta)| - \frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta) - \frac{1}{2}\ln\left|\mathrm{X}^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}\right| - \frac{N - p_x}{2}\ln(2\pi)$$

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 $-2\ell(\theta; v)$ or $-2\ell_R(\theta; v)$, 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 $X\tilde{\beta} + Z\tilde{\gamma} + O^*$ 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. 200.

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. 201.

Statistics for estimates of fixed and random effects

Similarly to "Statistics for estimates of fixed and random effects " on p. 202, 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 $R^* = v\hat{a}r\left(v|\gamma\right) = \hat{D}^{-1} A_{\hat{\pi}}^{1/2} R A_{\hat{\pi}}^{1/2} \hat{D}^{-1}$ with $\hat{D} = \overset{S}{\underset{s=1}{\oplus}} \overset{T_s}{\underset{t=1}{\oplus}} \left(diag\left(\hat{\pi}_{st}\right) - \hat{\pi}_{st}\hat{\pi}_{st}^T\right)$, 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:

$$\hat{\beta} = T\hat{\beta}^*$$

$$\hat{C}_{11} = T\hat{C}_{11}^* T^T$$

$$\hat{C}_{21} = \hat{C}_{21}^* T^T$$

$$\hat{C}_{22} = \hat{C}_{22}^*$$

where $T = \overset{J-1}{\underset{j=1}{\oplus}} A_j$.

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 sth 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 ith random effects prediction. Its standard error is the square root of the ith 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. 202.

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. 202.

Testing

Information criteria

These are as described in "Goodness of fit " on p. 205.

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 υ. For more information, see the topic "Method for computing degrees of freedom " on p. 209.

Scoring

PQL-type predicted values and relevant statistics

$(J-1) \times 1$ predicted vector of the linear predictor

$$\hat{\eta}_{st} = (I_{J-1} \otimes x_{st})^{\mathrm{T}} \hat{\beta} + (I_{J-1} \otimes z_{st})^{\mathrm{T}} \hat{\gamma}_s + 1_{J-1} \otimes o_{st}$$

Estimated covariance matrix of the linear predictor

$$\begin{aligned}
\Sigma_{\hat{\eta}_{st}} = \ & (I_{J-1} \otimes x_{st})^T \Sigma (I_{J-1} \otimes x_{st}) + (I_{J-1} \otimes z_{st})^T \hat{C}_{22}^s (I_{J-1} \otimes z_{st}) \\
& + (I_{J-1} \otimes z_{st})^T \hat{C}_{21}^s (I_{J-1} \otimes x_{st}) + (I_{J-1} \otimes x_{st})^T \left(\hat{C}_{21}^s \right)^T (I_{J-1} \otimes z_{st})
\end{aligned}$$

where $\hat{C}_{22}^s$ is a diagonal block corresponding to the sth 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 sth super subject.

The estimated standard error of the jth element in $\hat{\eta}_{st}$, $\hat{\eta}_{st,j}$, is the square root of the jth 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

$$\hat{\pi}_{st,j} = g^{-1}(\hat{\eta}_{st,j}) = \begin{cases} \dfrac{\exp\left(\hat{\eta}_{st,j}\right)}{1 + \displaystyle\sum_{k=1}^{J-1} \exp\left(\hat{\eta}_{st,k}\right)}, & j = 1, \cdots, J-1, \\[2em] \dfrac{1}{1 + \displaystyle\sum_{k=1}^{J-1} \exp\left(\hat{\eta}_{st,k}\right)}, & j = J. \end{cases}$$

Predicted category

$$c(\mathbf{x}_{st}) = arg \max_j \hat{\pi}_{st,j},$$

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 $\hat{\sigma}_{\pi_{st,j}}^2$ is the jth 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 sth super subject.
y_{st}	Ordinal categorical target for the tth case in the sth 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}, $y_{st} = (y_{st,1}, \cdots, y_{st,J-1})^{\mathrm{T}}$, 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	$y_s = \left(y_{s1}^{\mathrm{T}}, \ldots, y_{sT_s}^{\mathrm{T}}\right)^{\mathrm{T}}, s = 1, \cdots, S.$
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 tth case in the sth 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 $\lambda_{st}^{\mathrm{T}} = (\lambda_{st,1}, \cdots, \lambda_{st,J-1})$, $s = 1, \ldots, S$ and $t = 1, \ldots, T_s.$
$\pi_{st,j}$	Probability of category j for the tth case in the sth 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 tth case in the sth 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	$(n\,(J{-}1)) \times 1$ vector of linear predictor. $\eta = \left(\eta_1^{\mathrm{T}}, \cdots, \eta_S^{\mathrm{T}}\right)^{\mathrm{T}}$
x_{st}	$p \times 1$ vector of predictors for the tth case in the sth super subject.
z_{st}	$r \times 1$ vector of coefficients for the random effect corresponding to the tth case in the sth super subject.
O	$n \times 1$ vector of offsets, $O = (o_{11}, \cdots, o_{1T_1}, \cdots, o_{ST_S})^{T}$, where o_{si} is the offset value of the tth case in the sth 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	$J{-}1 \times 1$ 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	$p \times 1$ vector of unknown parameters.
B	$(J{-}1{+}p) \times 1$ vector of all parameters B=$\left(\psi^{\mathrm{T}}, \beta^{\mathrm{T}}\right)^{\mathrm{T}}$
ω_{st}	Scale weight of the tth case in the sth 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	$n \times 1$ 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 ith case in the sth 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	$n \times 1$ 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, $N = n$.
$A \otimes B$	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	$m \times 1$ 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

$$\eta = g\,(\lambda) = XB + Z\gamma + O^*$$

where η is the expanded linear predictor vector; λ is the expanded cumulative target probability vector; $g\left(.\right)$ is a cumulative link function; $\mathbf{X}$ is the expanded design matrix for fixed effects arranged as follows

$$\mathrm{X} = \begin{pmatrix} \mathrm{X}_1 \\ \vdots \\ \mathrm{X}_S \end{pmatrix},$$

$$\mathrm{X}_s = \begin{pmatrix} \mathrm{X}_{s1} \\ \vdots \\ \mathrm{X}_{sT_s} \end{pmatrix}_{T_s(J-1)\times(J-1+p)},$$

$$\mathrm{X}_{st} = \left(\mathrm{I}_{J-1} \quad 1_{J-1} \otimes -\mathrm{x}_{st}^{\mathrm{T}} \right)_{(J-1)\times(J-1+p)}$$

$$= \begin{pmatrix} 1 & \cdots & 0 & -\mathrm{x}_{st}^{\mathrm{T}} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -\mathrm{x}_{st}^{\mathrm{T}} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \cdots & 0 & -x_{st,1} & \cdots & -x_{st,p} \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 1 & -x_{st,1} & \cdots & -x_{st,p} \end{pmatrix}$$

$\mathrm{B} = \left(\psi^{\mathrm{T}},\beta^{\mathrm{T}}\right)^{\mathrm{T}} = \left((\psi_1,\ldots,\psi_{J-1}),\beta^{\mathrm{T}} \right)^{\mathrm{T}}$; $\mathbf{Z}$ is the expanded design matrix for random effects arranged as follows

$$\mathrm{Z} = \begin{pmatrix} \mathrm{Z}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathrm{Z}_S \end{pmatrix},$$

$$\mathrm{Z}_s = \begin{pmatrix} \mathrm{Z}_{s1} \\ \vdots \\ \mathrm{Z}_{sT_s} \end{pmatrix}_{T_s(J-1)\times r},$$

$$\mathrm{Z}_{st} = \left(1_{J-1} \otimes -\mathrm{z}_{st}^{\mathrm{T}} \right)_{(J-1)\times r},$$

γ is a vector of random effects which are assumed to be normally distributed with mean $\mathbf{0}$ and variance matrix $\mathbf{G}$.

The variance of $\mathbf{y}$, conditional on the random effects is

$$Var\left(y|\gamma\right) = A_\mu^{1/2} R A_\mu^{1/2}$$

where $A_\mu = \overset{S}{\underset{s=1}{\oplus}} \overset{T_s}{\underset{t=1}{\oplus}} \left(diag\left(\pi_{st}\right) - \pi_{st}\pi_{st}^{\mathrm{T}} \right) /\omega_{st}$ and $\mathrm{R} = \phi\mathrm{I}$ 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. 198, 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 $\varepsilon \sim N\left(0, \mathrm{D}^{-1}A_{\tilde{\pi}}^{1/2}RA_{\tilde{\pi}}^{1/2}\left(\mathrm{D}^{-1}\right)^{\mathrm{T}}\right)$ with

$$\mathrm{D} = \overset{S}{\underset{s=1}{\oplus}}\ \overset{T_s}{\underset{t=1}{\oplus}}\ \mathrm{D}_{st} = \overset{S}{\underset{s=1}{\oplus}}\ \overset{T_s}{\underset{t=1}{\oplus}}\ \frac{\mathrm{d}g^{-1}(\tilde{\eta}_{st})}{\mathrm{d}\tilde{\eta}_{st}} = \overset{S}{\underset{s=1}{\oplus}}\ \overset{T_s}{\underset{t=1}{\oplus}}\ \frac{\mathrm{d}\tilde{\lambda}_{st}}{\mathrm{d}\tilde{\eta}_{st}}$$

$$\mathrm{D}_{st} = \begin{bmatrix} \frac{\partial \tilde{\lambda}_{st,1}}{\partial \tilde{\eta}_{st,1}} & 0 & \cdots & 0 & 0 \\ -\frac{\partial \tilde{\lambda}_{st,1}}{\partial \tilde{\eta}_{st,1}} & \frac{\partial \tilde{\lambda}_{st,2}}{\partial \tilde{\eta}_{st,2}} & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & \frac{\partial \tilde{\lambda}_{st,J-2}}{\partial \tilde{\eta}_{st,J-2}} & 0 \\ 0 & 0 & \cdots & -\frac{\partial \tilde{\lambda}_{st,J-2}}{\partial \tilde{\eta}_{st,J-2}} & \frac{\partial \tilde{\lambda}_{st,J-1}}{\partial \tilde{\eta}_{st,J-1}} \end{bmatrix}$$

and

$$A_{\tilde{\mu}} = \overset{S}{\underset{s=1}{\oplus}}\ \overset{T_s}{\underset{t=1}{\oplus}}\ \left(diag\left(\tilde{\pi}_{st}\right) - \tilde{\pi}_{st}\tilde{\pi}_{st}^{\mathrm{T}}\right)/\omega_{st}.$$

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:

$$\ell(\theta; \mathrm{v}) = -\frac{1}{2}\ln|\mathrm{V}(\theta)| - \frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta) - \frac{N}{2}\ln(2\pi)$$

$$\ell_R(\theta; \mathrm{v}) = -\frac{1}{2}\ln|\mathrm{V}(\theta)| - \frac{1}{2}\mathrm{r}(\theta)^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{r}(\theta) - \frac{1}{2}\ln\left|\mathrm{X}^{\mathrm{T}}\mathrm{V}(\theta)^{-1}\mathrm{X}\right| - \frac{N - p_x}{2}\ln(2\pi)$$

where $\mathrm{V}(\theta) = Z\mathrm{G}(\theta)Z^T + \tilde{W}^{-1/2}\mathrm{R}(\theta)\tilde{W}^{-1/2}$, $r(\theta) = v - X\hat{\mathbf{B}}$, 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 $X\tilde{\mathbf{B}} + Z\tilde{\gamma} + O^*$ 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. 200.

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. 201.

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

$$R^* = v\hat{a}r\left(v|\gamma\right) = \mathrm{D}^{-1} A_{\tilde{\pi}}^{1/2} R A_{\tilde{\pi}}^{1/2} \left(\mathrm{D}^{-1} \right)^{\mathrm{T}}.$$

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:

$$\mathrm{B} = \begin{pmatrix} \psi \\ \beta \end{pmatrix} = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_{J-1} \\ \beta \end{pmatrix} = \mathrm{A} \begin{pmatrix} \psi^* \\ \beta^* \end{pmatrix} = \mathrm{A}\mathrm{B}^*$$

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. 202.

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 ith 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. 202.

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. 202.

Wald confidence intervals for estimates in fixed effects and random effects predictions

The $100(1 - \alpha)$% 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. 202.

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. 209.

Testing

Information criteria

These are as described in "Goodness of fit " on p. 205, 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 $\mathbf{L_i}$ is constructed and H_0: $\mathbf{L_i B} = \mathbf{0}$ is tested. Construction of matrix $\mathbf{L_i}$ is based on matrix $H_\omega = \left(X_1^T \Omega X_1 \right)^{-} X_1^T \Omega X_1$, where $X_1 = (1, -X)$ and such that $\mathbf{L_i B}$ is estimable. Note that $\mathbf{L_i B}$ is estimable if and only if $L_0 = L_0 H_\omega$, where $L_0 = (l_0, L(\beta))$. Construction of L_0 considers a partition of the more general test matrix $L_i = (L_i(\psi), L_i(\beta))$ first, where $L_i(\psi) = (l_1, \ldots, l_{J-1})$ consists of columns corresponding to the threshold parameters and $L_i(\beta)$ is the part of $\mathbf{L_i}$ corresponding to regression parameters, then replace $L_i(\psi)$ with their sum $l_0 = \sum_{j=1}^{J-1} l_j$ to get L_0.

Note that the threshold-parameter effect is not tested for both type I and III analyses and construction of $\mathbf{L_i}$ is the same as in GENLIN. For more information, see the topic "Default Tests of Model Effects " in Chapter 18 on p. 188. Similarly, if the fixed effects are transformed when constructing matrix $\mathbf{X}$, 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

$$\Sigma_{\hat{\eta}_{st}} = X_{st}\Sigma X_{st}^T + Z_{st}\hat{C}_{22}^s Z_{st}^T + Z_{st}\hat{C}_{21}^s X_{st}^T + X_{st}\left(\hat{C}_{21}^s\right)^T Z_{st}^T$$

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

$$c(\mathbf{x}_{st}) = arg \max_{j} \hat{\pi}_{st,j},$$

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 $\left(\mathbf{X}_{st,j}, \mathbf{Z}_{st,j}\right)$, where $\mathbf{X}_{st,j}$ and $\mathbf{Z}_{st,j}$ are the *j*th rows of $\mathbf{X}_{st}$ and $\mathbf{Z}_{st}$, respectively, corresponding to the *j*th category. For example, the **L** matrix is $\left(1, 0, \ldots, 0, -\mathbf{x}_{st}^{\mathrm{T}}, -\mathbf{z}_{st}^{\mathrm{T}}\right)_{1 \times (J-1+p+r)}$ for the 1st category. The computation should then be similar to that described in "Method for computing degrees of freedom" on p. 209.

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.

229

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" in Chapter 9 on p. 65.

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_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.

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 qth encoded input field for the ith record, and c_{qj} is the value of the qth encoded input field for the jth 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, $x_{qi}(j)$ is the qth 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 jth 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. 234.

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. 236.

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. 241.

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. 241.

▶ 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. 241.

▶ 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. For more information, see the topic "Overview" in Chapter 26 on p. 291.

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 jth 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. 242.

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. 257.

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

Record #	X	X_1'	X_2'
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 ith 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 ith 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 jth 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 $j = 1, \ldots, J\text{-}1$.

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

$$
\begin{aligned}
l(\mathbf{B}) &= \sum_{i=1}^{m} \sum_{j=1}^{J} n_{ij} \log\left(\pi_{ij}\right) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{J} n_{ij} \log\left(\frac{\exp\left(\mathbf{x}'_i \beta_j\right)}{1 + \sum_{k=1}^{J-1} \exp\left(\mathbf{x}'_i \beta_l\right)}\right)
\end{aligned}
$$

A constant that is independent of parameters has been excluded here. The value of the constant is $c = \sum_{i=1}^{m} \log\left(n_i! / (n_{i1}! \ldots n_{iJ}!)\right)$.

Model Parameters

Derivatives of the Log-Likelihood

For any $j = 1, ..., J\text{-}1$, $s = 1, ..., p$, 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 $j, j' = 1, ..., J\text{-}1$ 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

$$\Delta_i = n_i \left(Diag\left(\pi_i^{(-J)}\right) - \pi_i^{(-J)} \pi_i^{(-J)\prime} \right)$$

in which $\pi_i^{(-J)} = (\pi_{i1}, ..., \pi_{iJ-1})'$ and $Diag\left(\pi_i^{(-J)}\right)$ 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 $l\left(\mathbf{B}^{(\nu+1)}\right) - l\left(\mathbf{B}^{(\nu)}\right) \geq 0$, $\mathbf{X}^*$ 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 $l\left(\mathbf{B}^{(\nu+1)}\right) - l\left(\mathbf{B}^{(\nu)}\right) < 0$. 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 $\beta_j^{(0)} = \left(\beta_{j1}^{(0)}, 0, ..., 0\right)'$ where

$$\beta_{j1}^{(0)} = \log\left(\frac{\tilde{\pi}_{ij}}{\tilde{\pi}_{iJ}}\right) = \log\left(\frac{\sum\limits_{i=1}^{m} n_{ij}}{\sum\limits_{i=1}^{m} n_{iJ}}\right)$$

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

$$\tilde{\pi}_{ij} = \frac{\sum\limits_{i=1}^{m} n_{ij}}{N}$$

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

$$-2l(\tilde{\pi}) - \left\{-2l(\hat{\pi})\right\}$$

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:

$$U\left(B\right) = \frac{\partial l(B)}{\partial B}$$

The (*j,s*)th element of the score function can be written as

$$
\begin{aligned}
\left[U\left(B\right)\right]_{js} &= \frac{\partial l(B)}{\partial \beta_{js}} \\
&= \sum_{i=1}^{m} x_{is}\left(n_{ij} - n_i \pi_{ij}\right)
\end{aligned}
$$

Similarly, elements of the information matrix are given by

$$
\begin{aligned}
\left[I\left(B\right)\right]_{js,j't} &= \frac{\partial^2 l(B)}{\partial \beta_{js}\partial \beta_{jt}} \\
&= -\sum_{i=1}^{m} n_i x_{is} x_{it} \pi_{ij}\left(\delta_{jj'} - \pi_{ij'}\right)
\end{aligned}
$$

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:

$$
\begin{aligned}
U\left(B_1, B_2\right) &= \begin{pmatrix} U_1\left(B_1, B_2\right) \\ U_2\left(B_1, B_2\right) \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial l(B_1, B_2)}{\partial B_1} \\ \frac{\partial l(B_1, B_2)}{\partial B_2} \end{pmatrix}
\end{aligned}
$$

where $l\left(B_1, B_2\right) = l\left(B\right)$

$$
\begin{aligned}
I\left(B\right) &= I\left(B_1, B_2\right) \\
&= \begin{pmatrix} I_{11}\left(B_1, B_2\right) & I_{12}\left(B_1, B_2\right) \\ I_{21}\left(B_1, B_2\right) & I_{22}\left(B_1, B_2\right) \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial^2 l(B_1, B_2)}{\partial B_1 \partial B_1} & \frac{\partial^2 l(B_1, B_2)}{\partial B_1 \partial B_2} \\ \frac{\partial^2 l(B_1, B_2)}{\partial B_2 \partial B_1} & \frac{\partial^2 l(B_1, B_2)}{\partial B_2 \partial B_2} \end{pmatrix}
\end{aligned}
$$

$$J\left(B\right) = I(B_1, B_2)^- = \begin{pmatrix} J_{11}\left(B_1, B_2\right) & J_{12}\left(B_1, B_2\right) \\ J_{21}\left(B_1, B_2\right) & J_{22}\left(B_1, B_2\right) \end{pmatrix}$$

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

$$H_0 : B_E = 0 \text{ v.s. } H_1 : B_E \neq 0$$

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:

$$\left[U_E\left(\hat{B}_{base},0\right)\right]_{kt} = \sum_{i=1}^{m} x_{it}\left(n_{ik} - n_i\hat{\pi}_{ik}\right)$$

$$\left[I_{E,E}\left(\hat{B}_{base},0\right)\right]_{kt,k't'} = -\sum_{i=1}^{m} n_i x_{it} x_{it'} \hat{\pi}_{ik}\left(\delta_{kk'} - \hat{\pi}_{ik'}\right)$$

$$\left[I_{base,E}\left(\hat{B}_{base},0\right)\right]_{js,kt} = -\sum_{i=1}^{m} n_i x_{is} x_{it} \hat{\pi}_{ij}\left(\delta_{jk} - \hat{\pi}_{ik}\right)$$

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

$$H_0 : B_F = 0 \text{ vs. } H_1 : B_F \neq 0$$

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^*,

$$\hat{\pi}_{ij} = \frac{\exp\left(r_{ij'}\right)}{1 + \displaystyle\sum_{k=1}^{J-1} \exp\left(r_{ij'}\right)} = \frac{\exp\left(\mathbf{x}'_i\beta_j\right)}{1 + \displaystyle\sum_{k=1}^{J-1} \exp\left(\mathbf{x}'_i\beta_k\right)}$$

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 ith case of the dichotomous dependent variable
$\mathbf{X}$	$n \times p$ matrix with element x_{ij}, the observed value of the ith case of the jth parameter
β	$p \times 1$ vector with element β_j, the coefficient for the jth parameter
$\mathbf{w}$	$n \times 1$ vector with element w_i, the weight for the ith case
l	Likelihood function
L	Log-likelihood function
$\mathbf{I}$	Information matrix

Model

The linear logistic model assumes a dichotomous dependent variable Y with probability π, where for the ith 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

$$L = \ln(l) = \sum_{i=1}^{n} \left(w_i y_i \ln(\pi_i) + w_i(1-y_i)\ln(1-\pi_i)\right)$$

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

$$\mathbf{A}_{11} = \underset{\sim}{\alpha}' \hat{\mathbf{V}} \underset{\sim}{\alpha},$$
$$\mathbf{A}_{12,i} = \underset{\sim}{\alpha}' \hat{\mathbf{V}} \underset{\sim}{\gamma}_i,$$
$$\mathbf{A}_{22,i} = \underset{\sim}{\gamma}'_i \hat{\mathbf{V}} \underset{\sim}{\gamma}_i$$

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 ith variable is not categorical, the Wald statistic is defined by

$$Wald_i = \frac{\hat{\beta}_i^2}{\hat{\sigma}_{\hat{\beta}_i}^2}$$

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

$$LR = -2 \ln \left(\frac{l(reduced)}{l(full)} \right) = -2(L(reduced) - L(full))$$

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 $\mathbf{C}$ 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

$$L(0) = W \ln{(0.5)} = -0.6931472 W$$

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

$$L(0) = W \left[\hat{\pi}_0 \ln{\left(\frac{\hat{\pi}_0}{1 - \hat{\pi}_0} \right)} + \ln{(1 - \hat{\pi}_0)}. \right]$$

Model Information

The following statistics are computed if a stepwise method is specified.

–2 Log-Likelihood

$$-2 \sum_{i=1}^{n} \left(w_i y_i \ln{(\hat{\pi}_i)} + w_i (1 - y_i) \ln{(1 - \hat{\pi}_i)} \right)$$

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)

$$R_{CS}^2 = 1 - \left(\frac{l(0)}{l(\hat{\beta})} \right)^{\frac{2}{W}}$$

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)

$$R_N^2 = R_{CS}^2 / \max \left(R_{CS}^2 \right)$$

where $\max \left(R_{CS}^2 \right) = 1 - \{l(0)\}^{2/W}$.

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 qth block has m_q number of observations, $q = 1, \ldots, Q$. Moreover, suppose that the kth group ($k = 1, \ldots, g$) is composed of the q_1th, ..., q_kth blocks of observations. Then the total number of observations in the kth group is $s_k = \Sigma_{q_1}^{q_k} m_j$. The total observed frequency of events (that is, Y=1) in the kth group, call it O_{1k}, is the total number of observations in the kth group with Y=1. Let E_{1k} be the total expected frequency of the event in the

kth group; then E_{1k} is given by $E_{1k} = s_k \xi_k$, where ξ_k is the average predicted event probability for the kth 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 $\Pr(\chi^2 \geq \chi^2_{HL})$ 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

$$R_{CS} = \left(L_{\mathbf{g}}^*\right)' B_{22} L_{\mathbf{g}}^*$$

where $L_{\mathbf{g}}^* = \left(L_{\gamma_1}^*, \ldots, L_{\gamma_{r_2}}^*\right)'$

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

$$\hat{\pi}_i = \frac{\exp\left(\hat{\eta}_i\right)}{1 + \exp\left(\hat{\eta}_i\right)}$$

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 $null$.

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:

Y	Optional $1 \times N$ vector of responses with element y_n, where $n=1,...,N$ indexes the cases.
$\mathbf{X}^0$	$P^0 \times N$ matrix of features with element x_{pn}^0, where $p=1,...,P^0$ indexes the features and $n=1,...,N$ indexes the cases.
X	$P \times N$ 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 $y_{(j_1)} \leq y_{(m_2)} \leq \cdots \leq y_{(m_k)}$, 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.

Chapter

25

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. 131.

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 **y**.
R	$(p+1) \times (p+1)$ weighted sample correlation matrix for **X** (excluding the intercept, if it exists) and **y**.
$\tilde{\mathbf{R}}$	The resulting matrix after a sweep operation whose elements are $\tilde{r}_{ij}$.

Model

Linear regression has the form

$$\mathrm{y} = \mathrm{X}\beta + \varepsilon$$

where ε follows a normal distribution with mean 0 and variance $\sigma^2 \mathrm{D}^{-1}$, where $\mathrm{D}^{-1} = diag(1/g_1, \ldots, 1/g_n)$. 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 $\mathrm{D}^{1/2}$ as follows:

$$\mathrm{D}^{1/2}\mathrm{y} = \mathrm{D}^{1/2}\mathrm{X}\beta + \mathrm{D}^{1/2}\varepsilon$$

so that the new unobserved error $\mathrm{D}^{1/2}\varepsilon$ follows a normal distribution $N_n(0, \sigma^2\mathbf{I})$, where **I** is an identity matrix and $\mathrm{D}^{1/2} = diag(\sqrt{g_1}, \ldots, \sqrt{g_n})$. Then the least squares estimates of β can be obtained from the following formula

$$\hat{\beta} = arg \min_{\beta} \left(\mathrm{D}^{1/2}\mathrm{y} - \mathrm{D}^{1/2}\mathbf{X}\beta\right)^{\mathrm{T}} F\left(\mathrm{D}^{1/2}\mathrm{y} - \mathrm{D}^{1/2}\mathbf{X}\beta\right)$$

where $\mathrm{F} = diag(f_1, \ldots, f_n)$. Note that

$$\left(\mathrm{D}^{1/2}\mathbf{y} - \mathrm{D}^{1/2}\mathbf{X}\beta\right)^{\mathrm{T}} F\left(\mathrm{D}^{1/2}\mathbf{y} - \mathrm{D}^{1/2}\mathbf{X}\beta\right)$$
$$= (\mathbf{y} - \mathbf{X}\beta)^{\mathrm{T}}\mathrm{D}^{1/2}F\mathrm{D}^{1/2}(\mathbf{y} - \mathbf{X}\beta)$$
$$= (\mathbf{y} - \mathbf{X}\beta)^{\mathrm{T}}W(\mathbf{y} - \mathbf{X}\beta)$$

where $W = \text{diag}(w_1, \ldots, w_n) = \text{diag}(g_1 f_1, \ldots, g_n f_n)$, so the closed form solution of $\hat{\beta}$ is

$$\hat{\beta} = \left(\mathbf{X}^T \mathbf{W} \mathbf{X} \right)^{-} \mathbf{X}^T \mathbf{W} \mathbf{y}$$

$\hat{\beta}$ is computed by applying sweep operations instead of the equation above. In addition, sweep operations are applied to the transformed scale of $\mathbf{X}$ and $\mathbf{y}$ to achieve numerical stability. Specifically, we construct the weighted sample correlation matrix $\mathbf{R}$ then apply sweep operations to it. The $\mathbf{R}$ matrix is constructed as follows.

First, compute weighted sample means, variances and covariances among $\mathbf{X_i}$, $\mathbf{X_j}$, $i, j = 1, \ldots, p$, and $\mathbf{y}$:

Weighted sample means of $\mathbf{X_i}$ and $\mathbf{y}$ 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 $\mathbf{X_i}$ and $\mathbf{X_j}$ 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 $\mathbf{X_i}$ and $\mathbf{y}$ 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 $\mathbf{y}$ 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 $\mathbf{R}$ is

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} & r_{1y} \\ r_{21} & r_{22} & \cdots & r_{2p} & r_{2y} \\ \vdots & \vdots & \ddots & \vdots & \\ r_{p1} & r_{2p} & \cdots & r_{pp} & r_{py} \\ r_{y1} & r_{y2} & \cdots & r_{yp} & r_{yy} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{12}^T & R_{22} \end{bmatrix}$$

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

$$\tilde{\mathbf{R}} = \begin{bmatrix} R_{11}^{-1} & R_{11}^{-1} R_{12} \\ -R_{12}^T R_{11}^{-1} & R_{22} - R_{12}^T R_{11}^{-1} R_{12} \end{bmatrix}$$

The last column $R_{11}^{-1} R_{12}$ 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 $\mathbf{X}_j$ 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 $\mathbf{R}$ 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:

$$F_{enter_j} = \frac{\left(SSe_p - SSe_{p+\ell}\right)/\ell^*}{SSe_{p+\ell}/\left(N - p^r\right)} = \frac{\left(r_{yy} - \tilde{r}_{yy}\right)\left(N - p^r\right)}{\tilde{r}_{yy} \times \ell^*}$$

$$F_{remove_j} = \frac{\left(SSe_{p-\ell} - SSe_p\right)/\ell^*}{SSe_p/\left(N - p^c\right)} = \frac{\left(\tilde{r}_{yy} - r_{yy}\right)\left(N - p^c\right)}{r_{yy} \times \ell^*}$$

and their corresponding *p*-values are:

$$p_{enter_j} = P\left(F_{\ell^*, N-p^r} \geq F_{enter_j}\right) = 1 - P\left(F_{\ell^*, N-p^r} \leq F_{enter_j}\right)$$

$$p_{remove_j} = P\left(F_{\ell^*, N-p^c} \geq F_{remove_j}\right) = 1 - P\left(F_{\ell^*, N-p^c} \leq F_{remove_j}\right)$$

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 $\hat{y}_t = x_t\hat{\beta}$ 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. $FLAG_i = 1$ means the effect i is in the current model, $FLAG_i = 0$ means it is not. $|\{i | FLAG_i = 1\}|$ denotes the number of effects with $FLAG_i = 1$.
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* $\leq$ *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*}, $j^* = arg\min_j \{p_{enter_j}\}$ and if $p_{enter_{j*}} < P_{in}$, 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 $\{X_{j_s}\}_{s=1}^{\ell}$, $max\{r_{j_1 j_1}, r_{j_2 j_2}, \ldots, r_{j_\ell j_\ell}\} \geq t$;

where *t* is the singularity tolerance, and r_{jj} and $r_{j_s j_s}, s = 1, \ldots, \ell$, 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, $max\{\tilde{r}_{k_1 k_1}, \tilde{r}_{k_2 k_2}, \ldots, \tilde{r}_{k_{\ell'} k_{\ell'}}\} t \leq 1$.

where $\tilde{r}_{kk}$ and $\tilde{r}_{k_s k_s}, s = 1, \ldots, \ell'$, 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,
$$S_{p^e} = \left(\underline{S_{p^e-1}}, k, \underline{S_{p^e-1}} \right) = \left(\underline{S_{p^e-2}, k-1, S_{p^e-2}}, k, \underline{S_{p^e-2}, k-1, S_{p^e-2}} \right) = \ldots, \text{ and so on.}$$

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. 285) 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 $\mathbf{L_i}$ needs to be constructed first. Construction of $\mathbf{L_i}$ is based on the generating matrix $\mathrm{H}_\omega = \left(\mathrm{X^T DX}\right)^{-} \mathrm{X^T DX}$, where $\mathrm{D} = diag\left(g_1, \ldots, g_n\right)$, such that $\mathbf{L_i}\beta$ is estimable. It involves parameters only for the given effect and the effects containing the given effect. For type III analysis, $\mathbf{L_i}$ 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: $\mathbf{L_i}\beta$ = $\mathbf{0}$ 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

$$\mathrm{adj}.R^2 = 1 - \frac{SS_e / df_e}{SS_t / df_t} = R^2 - \frac{\left(1 - R^2\right) p^*}{df_e} = 1 - \frac{df_t \times \tilde{r}_{yy}}{df_e}$$

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)

$$AICC = N \ln\left(\frac{SS_e}{N}\right) + \frac{2p^c N}{N - p^c - 1}$$

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

$$\begin{aligned}\hat{\sigma}^2_{\hat{\beta}_0} &= \frac{(N-1)\tilde{r}_{yy}S_{yy}}{N(N-p^*-1)} + \sum_{j=1}^{p} \overline{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \overline{X}_k \overline{X}_j cov\left(\hat{\beta}_k, \hat{\beta}_j\right) \\ &= \frac{SS_e}{N \times df_e} + \sum_{j=1}^{p} \overline{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \overline{X}_k \overline{X}_j \frac{\tilde{r}_{kj} \times SS_e}{\sqrt{S_{kk}S_{jj}} \times (N-1)df_e}. \end{aligned}$$

$\hat{\sigma}^2_{\hat{\beta}_0} = \frac{(N-1)\tilde{r}_{yy}S_{yy}}{N(N-p^*-1)} + \sum_{j=1}^{p} \overline{X}_j^2 \hat{\sigma}^2_{\hat{\beta}_j} + 2\sum_{j=1}^{p-1}\sum_{k=j+1}^{p} \overline{X}_k \overline{X}_j cov\left(\hat{\beta}_k, \hat{\beta}_j\right)$ and $cov\left(\hat{\beta}_k, \hat{\beta}_j\right)$ is the kth row and jth 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 kth 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"

$$h_k = g_k \mathbf{x}_k \mathbf{G} \mathbf{x}_k^{\mathrm{T}}$$

is the kth 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.

Neural Network Algorithms

Overview

Note: A newer version of the Neural Net modeling node, with enhanced features, is available in this release. For more information, see the topic "Neural Networks Algorithms" in Chapter 27 on p. 305. Although you can still build and score a model with the previous version, we recommend using the new version from now on. Details of the previous version are retained here for reference, but support for it will be removed in a future release.

The basic element of a neural network is a **neuron**. This is a simple virtual device that accepts many inputs, sums them, applies a (usually nonlinear) transfer function, and generates the result, either as a model prediction or as input to other neurons.

A neural network is a structure of many such neurons connected in a systematic way. In IBM® SPSS® Modeler, the neural networks used are **feed-forward** neural networks, also known as **multilayer perceptrons**. The neurons in such networks (sometimes called **units**) are arranged in layers. Typically, there is one layer for input neurons (the **input layer**), one or more layers of internal processing units (the **hidden layers**), and one layer for output neurons (the **output layer**). Each layer is fully interconnected to the preceding layer and the following layer. For example, in a network with an input layer, a single hidden layer, and an output layer, each neuron in the input layer is connected to every neuron in the hidden layer, and each neuron in the hidden layer is connected to every neuron in the output layer.

The connections between neurons have weights associated with them, which determine the strength of influence one neuron has on another. Information flows from the input layer through the processing layer(s) to the output layer to generate predictions. By adjusting the connection weights during training to match predictions to target values for specific records, the network "learns" to generate better and better predictions.

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	X₁'	X₂'	X₃'
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.

Binary Set Encoding

An alternate encoding for binary sets is available for neural networks that may help them deal with large set fields. The default encoding creates one input for each possible value of a set field. For large sets this can create a burdensome number of inputs which can bog down the network and increase memory requirements.

The binary set encoding options use an encoding based on binary arithmetic to encode each set field as a group of numeric inputs. Instead of k input units for a set field, where k is the number of possible values for the set, binary encoding uses $\log_2(k + 1)$ input units (rounded up). To get the encoded values, enumerate the possible set values in ascending order and convert the number of the value to be encoded into its binary (base-2) representation. This gives a set of ones and zeros uniquely representing the input value. The binary representation sets the values of the recoded inputs.

For example, consider a set field with three possible values, A, B, and C. The field would be recoded as $\log_2(3) \approx 1.58 \rightarrow 2$ derived units, as illustrated in the table.

Record No.	X	X$_1$	X$_2$
1	A	0	1
2	B	1	0
3	C	1	1

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.

Multilayer Perceptron

The training of a multilayer perceptron uses a method called **back propagation of error**, based on the generalized delta rule (Rumelhart, McClelland, and The PDP Research Group, 1986). For each record presented to the network during training, information (in the form of input fields) feeds forward through the network to generate a prediction from the output layer. This prediction is compared to the recorded output value for the training record, and the difference between the predicted and actual output(s) is propagated backward through the network to adjust the connection weights to improve the prediction for similar patterns.

Feed-forward Calculations

Information flows through the network as follows:

Input neurons have their activations set to the values of the encoded input fields. The activation of each neuron in a hidden or output layer is calculated as

$$a_i = \sigma \left(\Sigma_j w_{ij} o_j \right),$$

where a_i is the activation of neuron i, j is the set of neurons in the preceding layer, w_{ij} is the weight of the connection between neuron i and neuron j, o_j is the output of neuron j, and $\sigma(x)$ is the **sigmoid** or **logistic** transfer function

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Back-propagation Calculations

At the beginning of training, all weights in the network are set to random values in the interval $-0.5 \le w_{ij} \le 0.5$.

Records are presented in **cycles** (also called **epochs**), where each cycle involves presenting *n* randomly selected training records to the network, where *n* is the number of records in the training data. Because of the random selection process, in any particular cycle some training records may be presented more than once and others may not be presented at all.

For each record, information flows through the network to generate a prediction, as described above. The prediction is compared to the target value found in the training data for the current record, and that difference is propagated back through the network to update the weights. To be more precise, the change value Δw for updating the weights is calculated as

$$\Delta w_{ij}(n+1) = \eta \delta_{pj} o_{pi} + \alpha \Delta w_{ij}(n),$$

where η is the learning rate parameter, δ_{pj} is the propagated error (described below), o_{pi} is the output of neuron *i* for record *p*, α is the momentum parameter, and $\Delta w_{ji}(n)$ is the change value for w_{ji} at the previous cycle.

The value of α is fixed during training, but the value of η varies across cycles of training. η starts at the user-specified *initial eta*, decreases logarithmically to the value of *low eta*, reverts to the *high eta* value, and then decreases again to *low eta*. The value of η is calculated as

$$\eta(t) = \eta(t-1) \cdot \exp\left(\log\left(\frac{\eta_{low}}{\eta_{high}}\right)/d\right),$$

where *d* is the user-specified number of *eta decay* cycles. If $\eta(t-1) < \eta_{low}$, then $\eta(t)$ is set to η_{high}. η and continues to cycle thusly until training is complete.

The back-propagated error value δ_{pj} is calculated based on where the connection lies in the network. For connections to output neurons it is calculated as

$$\delta_{pj} = (t_{pj} - o_{pj})o_{pj}(1 - o_{pj}),$$

where t_{pj} is the target value of output *j* for record *p*.

For weights that do not connect to the output neurons, δ_{pj} calculations take account of upstream error propagation,

$$\delta_{pj} = o_{pj}(1 - o_{pj})\sum_{k} \delta_{pk} w_{kj},$$

where *k* is the set of neurons to which neuron *j*'s output is connected, w_{kj} is the weight between the current neuron and neuron *k*, and δ_{pk} is the propagated error for that weight for the current input record.

Weights are updated immediately as each record is presented to the network during training.

RBFN

A **radial basis function network (RBFN)** is a special kind of neural network. It consists of three layers: an input layer, a hidden layer (also called a **receptor layer**), and an output layer. The input and output layers are similar to those of a multilayer perceptron. However, the hidden or receptor layer consists of neurons that represent clusters of input patterns, similar to the clusters in a *k*-means model. These clusters are based on **radial basis functions**, or functions of the distance between the RBF's center and a vector of input values. The connections between the

input neurons and the receptor neurons (receptor weights) are trained in essentially the same manner as a *k*-means model. (For more information, see the topic "Overview" in Chapter 21 on p. 233.) Specifically, the receptor weights are trained with only the input fields; the output field(s) are ignored for the first phase of training. Only after the receptor weights are optimized to find clusters in the input data are the connections between the receptors and the output neurons trained to generate predictions.

Estimating the RBF Centers

The RBF (receptor) centers are trained using the *k*-means algorithm as implemented in the *k*-means node. For more information, see the topic "Overview" in Chapter 21 on p. 233. The modeling parameters are fixed, however:

- The number of iterations is set to 10
- The change tolerance is set to 0.000001
- No set encoding value is used for set fields (or, equivalently, the encoding value is set to 1.0)

Assigning the Basis Function Widths

Each receptor neuron has a radial basis function associated with it. The basis function used in IBM® SPSS® Modeler is a multidimensional Gaussian,

$$\exp\left(-\frac{d_i{}^2}{2\sigma_i^2}\right),$$

where d_i is the distance from cluster center i, and σ_i is a scale parameter describing the size of the RBF for cluster/neuron i. You can think of the size of the RBF as the **receptive field** of the neuron, or how wide a range of inputs it will respond to.

The scale parameter σ_i is calculated based on the distances of the two closest clusters,

$$\sqrt{\frac{d_1 + d_2}{2}},$$

where d_1 is the distance between the cluster center and the center of the closest other cluster, and d_2 is the distance to the next closest cluster center. Thus, clusters that are close to other clusters will have a small receptive field, while those that are far from other clusters will have a larger receptive field.

Training the Output Weights for an RBFN

During output weight training, records are presented to the network as with a multilayer perceptron. The receptor neurons compute their activation as a function of their RBF size and the user-specified overlapping value *h*. The activation for receptor neuron *j* is calculated as

$$a_j = \exp\left(-\frac{||r - c||^2}{2\sigma_j^2 h}\right),$$

where *r* is the vector of record inputs and *c* is the cluster center vector.

The output neuron(s) are fully interconnected with the receptor or hidden neurons. The receptor neurons pass on their activation values, which are weighted and summed by the output neuron(s),

$$o_k = \sum_j W_{jk} a_j.$$

The output weights W_{jk} are trained in a manner similar to the training of a two-layer back-propagation network. The weights are initialized to small random values in the range $-0.001 \leq w_{ij} \leq 0.001$, and then they are updated at each cycle *t* by the formula

$$w_{jk}(t) = w_{jk}(t-1) + \Delta w_{jk}(t).$$

The change value $\Delta w_{jk}(t)$ is calculated as

$$\Delta w_{jk}(t) = \eta(r_k - o_k)a_i + \alpha \Delta w_{jk}(t-1),$$

which is analogous to the formula used in the back-propagation method. (For more information, see the topic "Back-propagation Calculations" on p. 293.) There is one important difference, however; in training the output layer of an RBFN, the value of η is fixed throughout training.

Effect of Options

Prevent Overtraining

This option splits the input data into two sets, a training set and a test set. Networks are trained on the training set, and error is evaluated on the test set at each cycle. The user can optionally set the seed for the random number generator to create a reproducible split between test and training sets. (Splits using the same seed will always assign the same records to the same subset, assuming identical ordering of records on input.)

Stop On

This option determines the criterion for terminating training cycles.

- **Default.** The default criterion uses the persistence parameter described below to determine when to stop training the network.
- **Accuracy.** At the end of each cycle the accuracy of the network is evaluated as described below. When the accuracy exceeds the specified value, training stops. Note that it is possible to set an accuracy criterion that will never be met for a particular modeling problem, in which case training will continue until interrupted by the user.
- **Cycles.** Training ceases after the specified number of cycles.
- **Time.** Training ceases after the specified amount of time has elapsed.

Persistence

Persistence determines how long a network will continue training when there is no improvement between cycles. The network will continue training until it has trained for the user-specified persistence number of cycles without any improvement.

Quick Method

When the quick method is selected, a single neural network is trained. By default, the network has one hidden layer containing $\max\left(3, \left(n_i + n_o\right)/20\right)$ neurons, where n_i is the number of input neurons and n_o is the number of output neurons. The network is trained using the back-propagation method described above.

Dynamic Method

When the dynamic method is selected, the topology of the network changes during training, with neurons added to improve performance until the network achieves the desired accuracy. There are two stages to dynamic training: finding the topology and training the final network.

Finding the Topology

Finding the topology follows these steps:

▶ Set the training parameters:
  - Persistence: 5
  - Alpha: 0.9
  - Initial eta: 0.05
  - Stop tolerance: 0.02

▶ Build a network with two hidden layers, each with two neurons. Train the initial network as usual through one cycle.

▶ Create two copies of the initial network, a *left* and a *right* network. To the right network, add one neuron to the second hidden layer.

▶ Train both augmented networks through one cycle, and determine the overall error for each network, calculated as the sum of the δ_{pj} across the j outputs and the p records in the cycle.

▶ If the left network has lower error, keep it and add one neuron to the right network's *first* hidden layer.

▶ If the right network has lower error, replace the left network with a copy of the right network, and add a neuron to the *second* hidden layer of the right network.

▶ Train both networks through another cycle, and repeat the training/augmentation cycle until the stopping criteria are met. For more information, see the topic "Effect of Options" on p. 296.

Adjusting Eta

With the dynamic training method, changes to eta take the performance of the networks so far into account. At each cycle, two vectors are computed: **movement**, based on the changes to the weights over the cycle,

$$M(t) = 2[W(t) - W(t-1)],$$

where $W(t)$ is the vector of weights at cycle t and $W(t-1)$ is the vector of weights at the previous cycle, and **change**, based on the momentum at the current cycle,

$$C(t) = 0.8 \cdot C(t-1) + M(t).$$

The ratio of the magnitudes of these vectors,

$$m(t) = \frac{||M(t)||}{||C(t)||},$$

is an index of the acceleration of training. If the index is less than $1 + \frac{||C(t)||}{10}$, training is slowing and eta is increased by a factor of 1.2. If the index is greater than 5.0, training is accelerating, and eta is decreased by a factor of $\frac{4}{m(t)}$.

Training the Final Network

After a good topology has been found, the final network is trained in the normal back-propagation manner, with the following settings:

- Persistence: 5
- Alpha: 0.9
- Initial eta: 0.02
- Stop tolerance: 0.005

Multiple Method

When the multiple method is selected, multiple networks are trained in pseudoparallel fashion. Each specified network is initialized, and all networks are trained. When the stopping criterion is met for all networks, the network with the highest accuracy is returned as the final model. For more information, see the topic "Effect of Options" on p. 296.

   The network topologies are taken from the user settings. If no set of networks is specified (e.g. when using "simple" options), IBM® SPSS® Modeler uses the following algorithm to generate network topologies:

- Single-layer networks are generated with different numbers of hidden units, from 3 up to the number of input neurons. These networks will always go up to 12 neurons (even if there are fewer than 12 input neurons) and will never go larger than 60 neurons.

A network is generated for each number of input neurons in the sequence 3, 4, 7, 12, and so on, with the increment at each step being two larger than the previous increment.

■ For each single-layer network, a set of two-layer networks is also created. The first layer has the same number of hidden neurons as the single-layer network, and the number of neurons in the second layer varies across networks.

A network is generated for each number of second-layer neurons in the sequence 2, 5, 10, 17, and so on, up to the number of neurons in the first hidden layer, again with the increment at each step being two larger than the previous increment.

Prune Method

The prune method is, conceptually, the opposite of the dynamic method. Rather than starting with a small network and building it up, the prune method starts with a large network and gradually **prunes** it by removing unhelpful neurons from the input and hidden layers.

Pruning proceeds in two stages: pruning the hidden neurons and pruning the input neurons. These stages are described below. The two-stage process repeats until the overall stopping criteria are met. For more information, see the topic "Effect of Options" on p. 296.

Hidden Neuron Pruning

In the first stage, the hidden neuron pruning proceeds as follows:

▶ Train the network on the training data.

▶ If any of the hidden layer stopping criteria are met, proceed to input neuron pruning. Hidden neuron pruning stops if:

■ Overall stopping criteria are satisfied

■ The hidden persistence limit is exceeded

■ The error of the current network is three times greater than the error of the best network so far

If none of the hidden layer stopping criteria are met, then a sensitivity analysis is performed on the hidden neurons to find the weakest neurons. (For more information, see the topic "Sensitivity Analysis" on p. 301.) The user-specified proportion of hidden neurons (specified as *hidden rate*) is removed, and another iteration of hidden neuron pruning proceeds.

Input Neuron Pruning

In the second stage, the input neuron pruning proceeds as follows:

▶ Train the network on the training data.

▶ If any of the input layer stopping criteria are met, check overall stopping criteria and repeat the two-stage process if necessary. Input neuron pruning stops if:

■ Overall stopping criteria are satisfied

■ The input persistence limit is exceeded

■ The error of the current network is three times greater than the error of the best network so far

If none of the input layer stopping criteria are met, then a sensitivity analysis is performed on the input neurons to find the weakest neurons. The user-specified proportion of input neurons (specified as *input rate*) is removed, and another iteration of input neuron pruning proceeds.

Overall Stopping Criteria

The overall stopping criteria are interpreted similarly to the stopping criteria for other training methods. The default values for the prune method are:

- Number of hidden layers: 1
- Number of units in hidden layer: $\min\left(50, round\left(\log\left(n_r\right)\log\left(k_i + k_o\right)\right)\right)$, where n_r is the number of records in the training data, k_i is the number of input units in the network, and k_o is the number of output units.
- Alpha: 0.9
- Initial eta: 0.4
- High eta: 0.15
- Low eta: 0.01
- Persistence: 100
- Overall persistence: 4
- Hidden persistence: $\min\left(10, \max\left(1, \frac{k_i + k_h}{10}\right)\right)$, where k_h is the number of hidden units
- Hidden rate: 0.15
- Input persistence: $\min\left(10, \max\left(2, \frac{k_i - k_o}{5}\right)\right)$
- Input rate: 0.15

Exhaustive Prune Method

The exhaustive prune method is a special case of the prune method, with the following parameter values:

- Number of hidden layers: 2
- Number of units in hidden layer 1: 30
- Number of units in hidden layer 2: 20
- Persistence: 200
- Overall persistence: 4
- Hidden persistence: 100
- Hidden rate: 0.02
- Input persistence: 100
- Input rate: 0.01

Blank Handling

In neural 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.

Secondary Calculations

Network Accuracy

If the *prevent overtraining* option is selected, network accuracy is based on the test set; otherwise, it is based on the training data. The calculation depends on the output field type.

Symbolic output fields. The percentage of records for which the prediction of the network matches the observed value in the data.

Numeric output fields. Accuracy is calculated as

$$accuracy = \sum_{R} \frac{1 - |t_r - o_r|}{\max{(t)} - \min{(t)}} / n_r,$$

where R is the set of records, t_r is the target output value for record r, o_r is the prediction generated by the network for record r, and n_r is the number of records.

For models with multiple outputs, the accuracy is the simple average of the accuracies for the individual outputs.

Sensitivity Analysis

Sensitivity analysis involves varying network activations and observing the resulting changes in other parts of the network (such as the output activations) to determine which parts of the network are most important and which are least important.

Input Units

When the user selects a sensitivity analysis, and internally when using the prune training method, the input units are subjected to sensitivity analysis to rank their importance. When analyzing the inputs, input fields are considered as the units of analysis, rather than individual input neurons. So, for set input fields, the entire group of input neurons representing the set field is analyzed together.

The sensitivity of an input field is calculated by varying the value of that input field for each record in the test set. As the value is varied, the maximum and minimum outputs are stored and the maximum difference in the outputs is calculated. This maximum difference is calculated for every record and then averaged. Values are varied as follows:

Flag fields. Values of *true* (1.0) and *false* (0.0) are used.

Set fields. Each possible value of the set field is tested (each input neuron for the set is turned on in turn for the default encoding; all combinations representing permissible input values are used for binary set encoding).

Numeric fields. The rescaled values 0.0, 0.25, 0.5, 0.75, and 1.0 are used, representing five equally spaced values covering the range of the original input field.

Hidden Units

When the prune training method is selected, the hidden units are also subjected to sensitivity analysis. Analysis of the hidden units proceeds as follows:

- The test data are passed through the network, and the results recorded as a baseline.
- The first hidden unit is turned off by temporarily setting its weights to zero. The test data are presented to the modified network, and the results compared to the baseline results. For each record, the absolute difference between the full network output and the modified network output is recorded, and the standard deviation of these values across all test records is calculated. For multiple outputs, the value is calculated separately for each output and then averaged across output units.
- The process is repeated for each hidden neuron, and the neurons are ranked according to this value, with large values indicating important neurons and small values indicating unimportant neurons.

Generated Model/Scoring

Predicted Values

Predicted values are generated by passing the record to be scored as input to the network and taking the output activation(s) as follows, based on output field type:

Flag fields. If the output activation $o < 0.5$, *false* (0.0) is the predicted value. If $o \geq 0.5$, *true* (1.0) is the predicted value.

Set fields. With the standard encoding, the output unit in the group representing the input field with the highest activation determines which value is set as the predicted value. With binary set encoding, the output activations for the output unit group are compared to each valid encoded value, and the encoded value with the smallest sum of errors across the group is selected as the predicted value.

Numeric fields. The predicted value is derived by reversing the initial numeric field encoding formula, which gives

$$p = o \cdot (x_{\max} - x_{\min}) + x_{\min}.$$

Confidence

Confidence values for neural network predictions are calculated based on the type of output field being predicted. The method for calculating confidence depends on the settings in the generated model node. 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.

Softmax

Softmax is equivalent to a multinomial logistic transformation, giving a probabilistic interpretation to the confidence values.

- **Set fields.** The confidence of each output category is

$$c_i = \frac{e^{o_i}}{\sum_{i=1}^{k} e^{o_i}},$$

  where o_i is the activation for output unit corresponding to category i, and k is the number of output categories.

Blank Handling

Blank handling for scoring is the same as during model building. For more information, see the topic "Blank Handling" on p. 301.

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. 131.

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. $J_0 = P$, $J_i = R$, 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 hth 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.
$c_{i:k}^m$	$\displaystyle\sum_{j=0}^{J_{i-1}} w_{i:j,k} a_{i-1:j}^m$, i=1,...,I.
$\gamma_i\left(c \right)$	Activation function for layer i.
$\mathbf{w}$	Weight vector containing all weights $\left(w_{1:0,1}, w_{1:0,2}, ..., w_{I:J_{I-1}, J_I} \right)$.

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

$$E_T\left(w\right) = \sum_{m=1}^{M} E_m\left(w\right)$$

where

$$E_m\left(w\right) = -\sum_{r\in\Gamma^c} y_r^{(m)} \log\left(\frac{a_{I:r}^m}{y_r^{(m)}}\right)$$

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

    $$g\left(R,P\right) = \begin{cases} \lfloor 4.5 + \sqrt{P+R}\rfloor & R < 5, P \geq 8 \\ \lfloor 0.5 + 0.5\left(P+R\right)\rfloor & \text{otherwise} \end{cases}$$

    where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. $h\left(R,P\right) = \left[\frac{M-R}{P+R+1}\right]$ 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 $a(k_1) = (0.5)^{k_1} a$. Let $\mathbf{w}_{\min}$ be the weights that give the minimum training error among all the perturbed weights. If $E_T(\mathbf{w}_{\min}) < E_T(\mathbf{w})$, 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 $\gamma'(c) = \begin{cases} 1 - [\gamma(c)]^2 & \text{tanh} \\ 1 & \text{identity} \end{cases}$

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

$$\delta_{I:p}^m = \frac{\partial E_m}{\partial c_{I:p}^m} = \begin{cases} a_{I:p}^m - y_p^{(m)} & \text{if cross-entropy error is used} \\ \gamma_I'(c_{I:p}^m)\left(a_{I:p}^m - y_p^{(m)}\right) & \text{otherwise} \end{cases}$$

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 $\frac{\partial E_m}{\partial w_{i:k,j}} = \delta_{i:j}^m a_{i-1:k}^m$, where $\delta_{i:j}^m = \frac{\partial E_m}{\partial c_{i:j}^m}$

▶ 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 $\delta_{i-1:k}^m = \gamma_{i-1}'(c_{i-1:k}^m) \sum_{j=1}^{J_i} \delta_{i:j}^m w_{i:k,j}$

This gives us a vector of $\sum_{i=0}^{I-1} (J_i + 1) J_{i+1}$ elements that form the gradient of $E_T(w_k)$.

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 $E_T(w_k)$ and its gradient $g_k = \nabla E_T(w_k)$. If $|g_k| < 10^{-6}$, stop and report the current network.

3. If $\eta_k|g_k| \leq \alpha|\Delta w_k|$, $\alpha = 0.9\eta_k \frac{|g_k|}{|\Delta w_k|}$. 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 $E_T(v) < E_T(w_k)$, 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 $\beta = (1/pK)*\ln(\eta_0/\eta_{low})$, 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 $E_{T_k}(w_k)$ and its gradient $g_k = \nabla E_{T_k}(w_k)$.

3. If $\eta_k|g_k| \leq \alpha|\Delta w_k|$, $\alpha = 0.9\eta_k \frac{|g_k|}{|\Delta w_k|}$. 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 $E_{T_k}(v) < E_{T_k}(w_k)$, 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 $K3$=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 $a_{I:r}^m = \frac{1}{M}\sum_{m=1}^{M} y_r^{(m)}$ 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 $R = \min(M,2N,1000)$, 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.
$\phi_j\left(X^{(m)}\right)$	jth 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.
$a_{i:j}^m$	Unit j of layer i, pattern m, $j = 0, ..., J_i$; $i = 0, ..., I$.
w_{rj}	weight connecting rth output unit and jth 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 $a_{1:j} = \phi_j(X)$ and

$$\phi_j(X) = \exp\left(-\sum_{p=1}^{P}\frac{1}{2\sigma_{jp}^2}(x_p - \mu_{jp})^2\right)\bigg/\sum_{j=1}^{J_1}\exp\left(-\sum_{p=1}^{P}\frac{1}{2\sigma_{jp}^2}(x_p - \mu_{jp})^2\right).$$

Output layer: J_2=R units, $a_{I:1}, \cdots, a_{I:J_2}$; with $a_{I:r} = w_{r0} + \sum_{j=1}^{J_1} w_{rj}\phi_j(X)$.

Error Function

Sum-of-squares error is used:

$$E_T(w) = \sum_{m=1}^{M} E_m(w)$$

where

$$E_m(w) = \frac{1}{2}\sum_{r=1}^{R}\left(y_r^{(m)} - a_{I:r}^m\right)^2$$

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, $a_{I:r}^m$ approximates the conditional expectation of the target value $E\left(y_r|X^{(m)}\right)$. For categorical targets, $a_{I:r}^m$ approximates the posterior probability of class k: $P\left(y_r = 1|X^{(m)}\right)$.

Note: though $\Sigma a_{I:r}^m = 1$ (the sum is over all classes of the same categorical target field), $a_{I:r}^m$ 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

$$\sigma_{jp} = h^{1/2} \begin{cases} s_{jp} & \text{if pth field is continuous} \\ \sqrt{p_{jp}(1-p_{jp})} & \text{if pth field is a dummy field of a categorical field} \end{cases}$$

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 $\min\{\sigma_j : \sigma_j \neq 0,\}_{j=1}^{J_1}$. If all σ_j are zero, set all of them to be $\sqrt{h}$.

When there are a large number of predictors, $\sum_{p=1}^{P}(x_p - \mu_{jp})^2$ could be easily very large and hence $\exp\left(-\sum_{p=1}^{P}\frac{1}{2\sigma_j^2}(x_p - \mu_{jp})^2\right)$ 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 $K_2 = \max(10, 2K, R)$.

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 each continuous target r, this is

$$accuracy = \frac{1}{n} \sum_{m=1}^{M} \left(1 - \frac{\left|y_r^{(m)} - \hat{y}_r^{(m)}\right|}{\max_m\left(y_r^{(m)}\right) - \min_m\left(y_r^{(m)}\right)}\right)$$

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" in Chapter 29 on p. 325.

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 ith 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.
Ent(S)	The class entropy of S.
E(A, T, S)	The class entropy of partition induced by T on A.
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).

319

Class Entropy

Let there be k classes C_1, ..., C_k and let $P(C_i, S)$ 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 $A \le T$, and $S_2 = S{-}S_1$. The class information entropy of the partition induced by T, E(A, T; S), is defined as

$$E\left(A, T; S\right) = \frac{|S_1|}{|S|} Ent\left(S_1\right) + \frac{|S_2|}{|S|} Ent\left(S_2\right)$$

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:

$$Gain\,(A, T; S) = 0.5916728$$

$$\frac{\log_2\,(N-1)}{N} + \frac{\Delta\,(A, T; S)}{N} = 0.6530774$$

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 $\leq T$, 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*$\neq\varnothing$) 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 $i \leq n$, 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. $D = \emptyset$;

2. EqualFrequency(A, n, D; S).

3. $T_A = \emptyset$.

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 $n_{\text{final}}+1$ 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. 320.

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.

325

$$VI_i = \frac{S_i}{\sum_{j=1}^{k} S_j}$$

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 $N \times k$:

$$\mathbf{M}_1 = \begin{matrix} x_1^{(1)} & x_2^{(1)} & \ldots & x_k^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ldots & x_k^{(2)} \\ \vdots & \vdots & \ldots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \ldots & x_k^{(N)} \end{matrix}$$

and

$$\mathbf{M}_2 = \begin{matrix} x_1^{(1')} & x_2^{(1')} & \ldots & x_k^{(1')} \\ x_1^{(2')} & x_2^{(2')} & \ldots & x_k^{(2')} \\ \vdots & \vdots & \ldots & \vdots \\ x_1^{(N')} & x_2^{(N')} & \ldots & x_k^{(N')} \end{matrix}$$

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 $\mathbf{N}_j$.

$$\mathbf{N}_j = \begin{matrix} x_1^{(1')} & x_2^{(1')} & \ldots & x_j^{(1)} & \ldots & x_k^{(1')} \\ x_1^{(2')} & x_2^{(2')} & \ldots & x_j^{(2)} & \ldots & x_k^{(2')} \\ \vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\ x_1^{(N')} & x_2^{(N')} & \ldots & x_j^{(N)} & \ldots & x_k^{(N')} \end{matrix}$$

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.

$$\hat{V}(Y) = \frac{1}{N-1}\sum_{r=1}^{N} f^2\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) - \hat{E}^2(Y)$$

where

$$\hat{E}^2(Y) = \left[\frac{1}{N}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right)\right]^2$$

$$\hat{V}(E(Y|X_j)) = \hat{U}_j - \hat{E}^2(Y)$$

where

$$\hat{U}_j = \frac{1}{N-1}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) f\left(x_1^{(r')}, x_2^{(r')}, \ldots, x_{(j-1)}^{(r')}, x_j^{(r)}, x_{(j+1)}^{(r')}, \ldots, x_k^{(r')}\right)$$

and

$$\hat{E}^2(Y) = \frac{1}{N}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) f\left(x_1^{(r')}, x_2^{(r')}, \ldots, x_k^{(r')}\right)$$

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:

$$\bigcap_{i \in I} \frac{1}{D}\sum_{j=t-D+1}^{t} \frac{\left|S_i(j) - \overline{S}_i\right|}{\overline{S}_i} < \epsilon$$

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. 337) 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. 333.

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 $\alpha_{nominal} = 0.05$.

Field Selection

1. For each predictor field X, if X is a symbolic (categorical) field, either nominal or ordinal, compute the p value of a Pearson chi-square test of independence between X and the dependent field. If X is scale-level (continuous), 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 than α_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" in Chapter 21 on p. 233.)

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 $p = \mathrm{Pr}\left(\chi_d^2 > X^2\right)$, where χ_d^2 follows a chi-square distribution with $d = (J-1)(I-1)$ degrees of freedom.

Expected Frequencies for Chi-Square Test

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

$$p_X = \mathrm{Pr}\left(F\left(J_t - 1, N_f(t) - J_t\right) > F_X\right)$$

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 v:

$$\overline{\mathbf{g}} = \frac{\sum_n f_n \mathbf{g}_n}{N_f}, \overline{\mathbf{g}}^{(j)} = \frac{\sum_n f_n \mathbf{g}_n I(y_n = j)}{N_{f,j}}$$

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, $\mathbf{D}$ = diag($d_1,...,d_I$) such that $d_1 \geq ... \geq d_I \geq 0$. Let $\mathbf{D}^{-\frac{1}{2}} = diag\,(d_1^*, ..., d_I^*)$ 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 $\mathbf{a}$ which is associated with its largest eigenvalue.

The largest discriminant coordinate of $\mathbf{g}$ 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 $\min\,(s_A^2, s_B^2) = 0$, 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 $\varepsilon = 10^{-12}$. 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.

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 $N_{f,j}(t)$ is the sum of frequency weights for records in category j for node t, and $N_f(t)$ 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 $p_f(t)$ 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 $N_{f,j}(t)$ 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 $N_{f,j}$ 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

$$p(s^* \approx s_X | t) = \begin{cases} \sum_j \dfrac{\pi(j) N_{w,j}(s^* \approx s_X, t)}{N_{w,j}(X^* \cap X)} & \text{if } Y \text{ is categorical} \\[2em] \dfrac{N_w(s^* \approx s_X, t)}{N_w(X^* \cap X)} & \text{if } Y \text{ is continuous} \end{cases}$$

with

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

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

$$N_{w,j}\left(X^*\cap X\right)=\sum_{n\in\hbar_{X^*\cap X}}w_nf_nI\left(y_n=j\right),\,N_{w,j}\left(X^*\cap X\right)=\sum_{n\in\hbar_{X^*\cap X}(t)}w_nf_nI\left(y_n=j\right)$$

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

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 $\pi'(j)$ is the adjusted prior for category *j*, and $\pi(j)$ 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. 338 and on p. 339, 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:

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

$R(T)$ 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 ($\alpha = 0$), 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 $T(\alpha)$, where $T(\alpha)$ 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 ($\alpha = 0$). 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 $\alpha > \alpha'$. 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 $\alpha_1 = 0$ 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, $T_1, T_2, \dots T_k$.

▶ 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

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

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 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, $N_{w,j}(t)$ 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),

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

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:

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

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. 333.

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 kth input field for record i
$\overline{X}_k$	Sample mean for the kth 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 kth 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

$$F - to - remove_k = \frac{(C - p^*)\,|V_k|}{r_{yy}}$$

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, $P\,(F - to - remove_k) > P_{out}$ (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, $P\,(F - to - enter_k) < P_{in}$ (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 $\mathbf{R}$ 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 non-positive definite $\mathbf{R}$ 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

$$R_{adj}^2 = R^2 - \frac{\left(1 - R^2\right)p}{C - p^*}$$

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 $null$.

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

Sequence	Support	Sequence	Support
{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
{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)
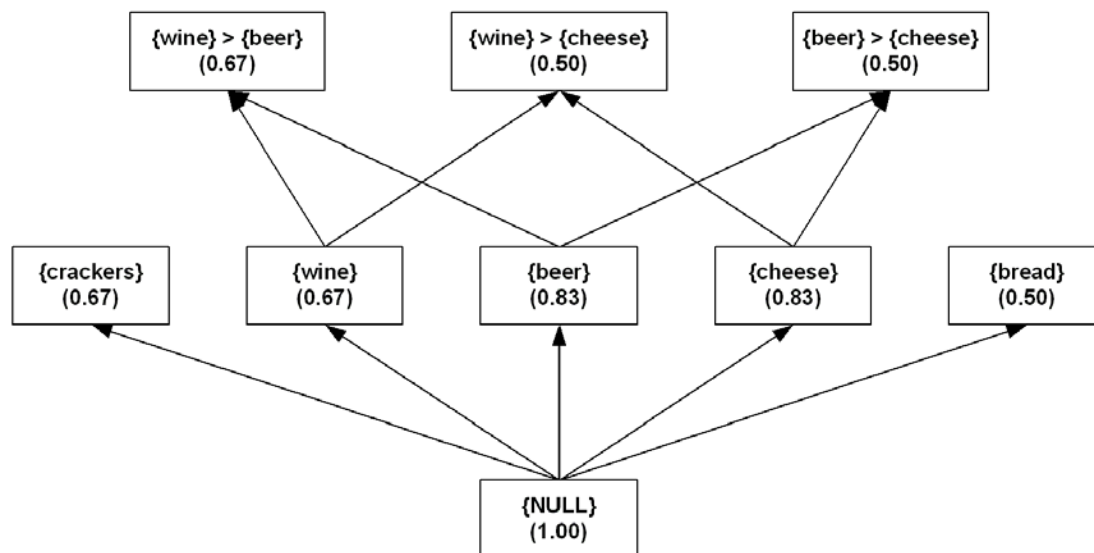
Mining for Frequent Sequences

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
{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)
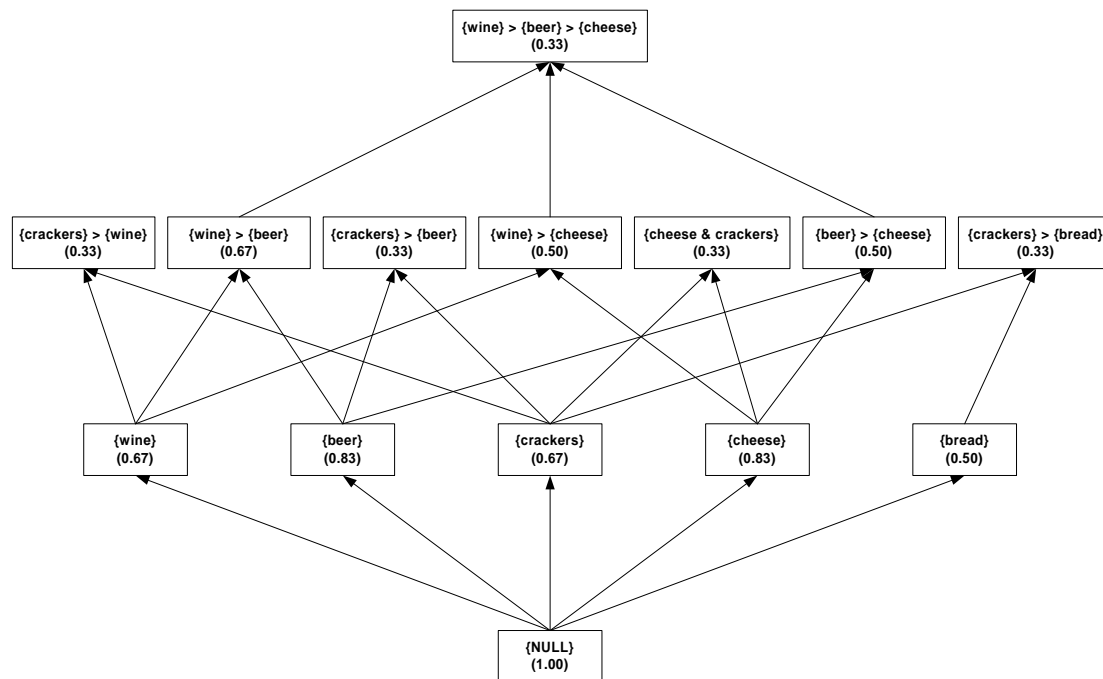
Notice that the lattice does not contain [{*crackers*} > {*wine*} > {*beer*}] although the support for this sequence exceeds the threshold. Although [{*crackers*} > {*wine*} > {*beer*}] occurs in one-third of the transactions, Carma cannot add this sequence to the lattice until all of its subsequences 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. 354.

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.

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 33-1
Notation

Notation	Description
J_0	Total number of predictors.
$\mathbf{X}$	Categorical predictor vector $\mathbf{X'} = (X_1, ..., X_J)$, 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:

$$P\left(\mathbf{X}=\mathbf{x}\,|Y=k\right)=\prod_{j=1}^{J}P\left(X_j=x_j|\,Y=k\right)$$

These probabilities are estimated from training data by the following equations:

$$\pi_k=P\left(Y=k\right)=\frac{N_k+\lambda}{N+K\lambda}$$

$$p_{mk}^j=P\left(X_j=m|Y=k\right)=\frac{N_{mk}^j+f}{\Sigma_{l=1}^{M_j}N_{lk}^j+M_jf}$$

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 $\hat{P}(X_j|Y = y)$

▶ 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. 358. 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_{mk}^j.

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 $\hat{P}(X_j | Y = y)$

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:

$$\begin{aligned} M\left(Y, X\right) &\equiv \sum_i f_x\left(x_j\right) \Delta H\left(x_j\right) \\ &= H_Y - H_{Y|X} \\ &= H_Y + H_X - H_{Y,X} \end{aligned}$$

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.

1. *SNA - Group Analysis Algorithm*

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.
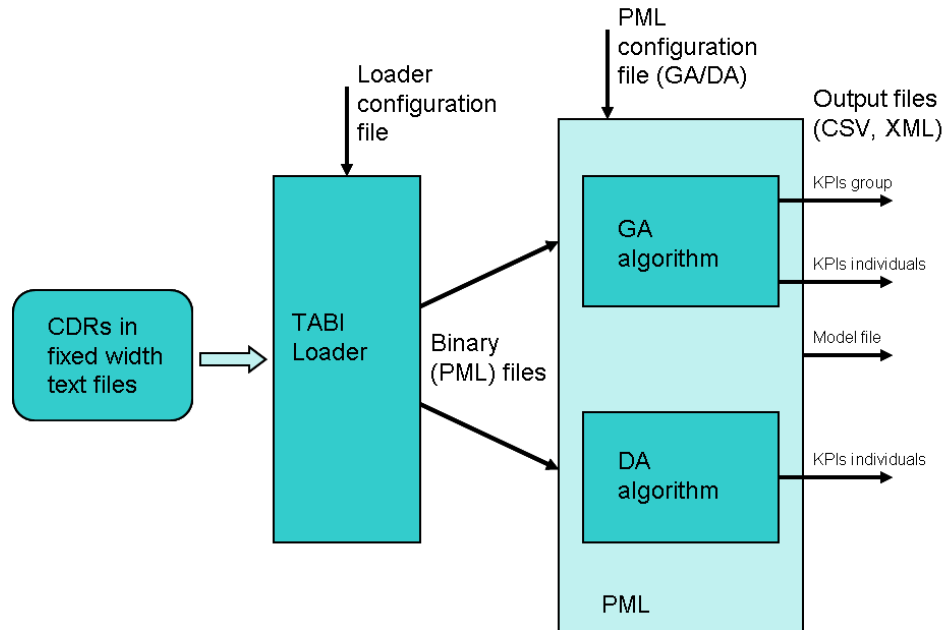
2.3 Sample Parameter File

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:

$$P_{00} \cdot \log\left(P_{00}/(Q1 \cdot Q2)\right) +$$
$$P_{00} \cdot \log\left(P_{01}/(Q_1 \cdot (1 - Q_2))\right) +$$
$$P_{10} \cdot \log\left(P_{10}/((1 - Q1) \cdot Q2)\right) +$$
$$P_{11} \cdot \log\left(P_{11}/((1 - Q1) \cdot (1 - Q2))\right)$$

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 $S_1, S_2, \ldots S_l$ be the core groups to whom i called

If (l > 0)

Let $k_1, \ldots ,k_l$ 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.
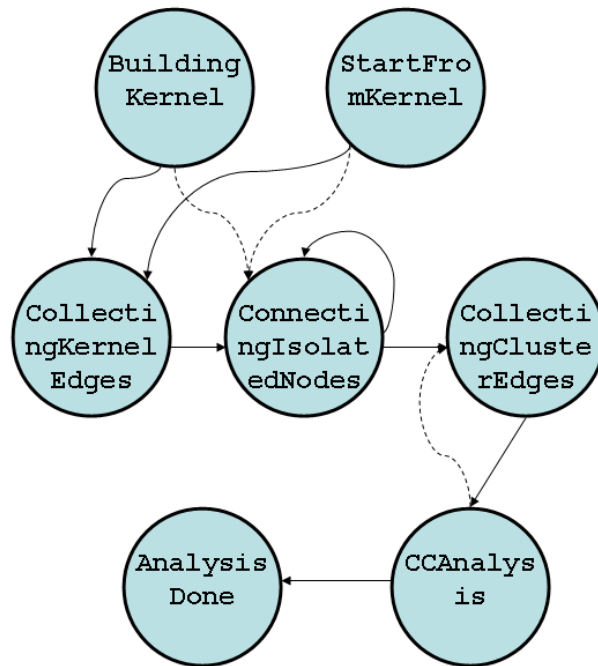
Circles represent states and edges represent the transitions between states. Dashed edges represent transitions which are possible under some parameter combinations, but are not common. The actual logic of the algorithm is complicated and should be learned from the code. In the current implementation 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

1. *SNA - Diffusion Analysis Algorithm*

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,

$w_{i,j} = \#calls(i \rightarrow j)$.

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,

$w_{i,j} = \sum_{calls(i \rightarrow j)} duration\_call(i \rightarrow j)$. 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:

$$C_T(n+1) = C_T(n) + C_F(n+1) - C_G(n+1)$$
$$= C_T(n) + d \cdot C_F(n) \cdot T_{adj} - d \cdot C_F(n) \cdot \Delta$$
$$= C_T(n) + d \cdot C_F(n) \cdot (T_{adj} - \Delta)$$

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.
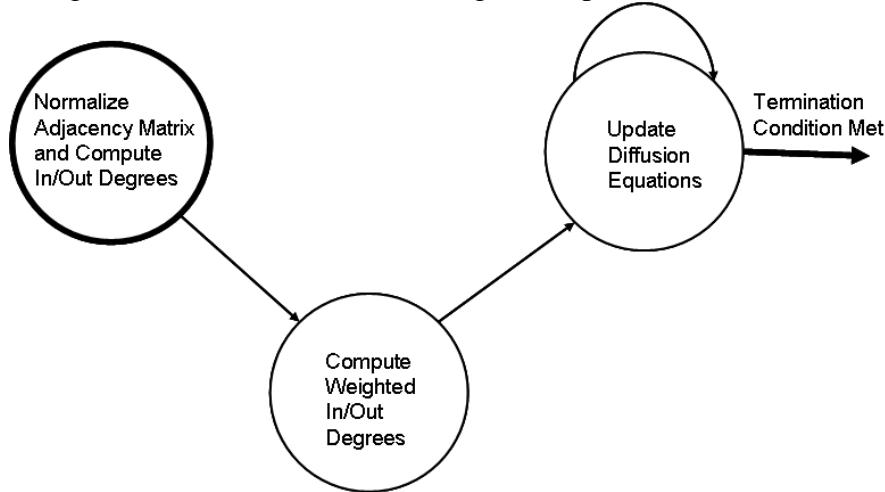
Circles represent states and edges represent the transitions between states. The process begins with the normalizing state (bold circle on the left hand side). During normalization in and out degrees are also calculated. The process continues with the computation of the weighted in and out degrees. The 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 $> \theta_{ITER}$

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

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 ith training sample
y_i	The class label for the ith 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. 367.

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:

$$\min f\left(\alpha\right) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

such that $0 \le \alpha_i \le C, i = 1, ..., l$ and $y^T\alpha = 0$, where

$$\alpha = \left(\alpha_1, \alpha_2, ..., \alpha_l\right)^T$$

and Q is an $l \times l$ matrix, $Q\left(x_i \cdot x_j\right) = y_i y_j K\left(x_i \cdot x_j\right)$

The decision function is

$$sgn\left(\sum_{i=1}^{l} y_i \alpha_i K\left(x_i, x\right) + b\right)$$

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

$$\min f\left(\alpha, \alpha^*\right) = \frac{1}{2}(\alpha - \alpha^*)^T Q\left(\alpha - \alpha^*\right) + \varepsilon\sum_{i=1}^{l}\left(\alpha_i + \alpha_i^*\right) + z_i\sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right)$$

such that $0 \le \alpha_i$ and $\alpha_i^* \le C$ for $i = 1, ..., l$, and

$$\sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right) = 0$$

where $\alpha = \left(\alpha_1, \alpha_2, ..., \alpha_l\right)^T$, $\alpha^* = \left(\alpha_1^*, \alpha_2^*, ..., \alpha_l^*\right)^T$, and Q is an $l \times l$ matrix, $Q_{ij} = K\left(x_i \cdot x_j\right)$

The approximate function is

$$\sum_{i=1}^{l} \left(-\alpha_i + \alpha_i^*\right) K\left(x_i, x\right) + b$$

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

$$\min \quad f\left(\alpha, \alpha^*\right) = \frac{1}{2}\left(\alpha^T (\alpha^*)^T\right) \begin{pmatrix} Q & -Q \\ -Q & Q \end{pmatrix} \begin{pmatrix} \alpha \\ \alpha^* \end{pmatrix} + \begin{pmatrix} \varepsilon e^T + z^T \\ \varepsilon e^T - z^T \end{pmatrix}^T \begin{pmatrix} \alpha \\ \alpha^* \end{pmatrix}$$

such that $y^T \begin{pmatrix} \alpha \\ \alpha^* \end{pmatrix} = 0$ and $0 \leq \alpha_i, \alpha_i^* \leq C$ for $i = 1, \dots , l$, where y is a $2l \times 1$ vector with $y_i = 1$ for $i = 1, \dots, l$ and $y_i = -1$ for $i = l + 1, \dots , 2l$.

Given this, the general form is

$$\min f\left(\alpha\right) = \frac{1}{2}\alpha^T Q \alpha + p^T \alpha$$

such that $0 \leq \alpha_i \leq C$ for $i = 1, \ldots, l$, and $y^T \alpha = constant$

	α in *W*(α)	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 $0 < \alpha_i < C$,

$$r_1 = \frac{\Sigma_{0 < \alpha_i < C, y_i = 1} \nabla f(\alpha_i)}{\Sigma_{0 < \alpha_i < C, y_i = 1} 1}$$

Otherwise,

$$r_1 = \frac{\max_{\alpha_i = C, y_i = 1} \nabla f(\alpha_i) + \min_{\alpha_i = 0, y_i = 1} \nabla f(\alpha_i)}{2}$$

▶ For $y_i = -1$:

If $0 < \alpha_i < C$,

$$r_2 = \frac{\Sigma_{0 < \alpha_i < C, y_i = -1} \nabla f(\alpha_i)}{\Sigma_{0 < \alpha_i < C, y_i = -1} 1}$$

Otherwise,

$$r_2 = \frac{\max_{\alpha_i = C, y_i = -1} \nabla f(\alpha_i) + \min_{\alpha_i = 0, y_i = -1} \nabla f(\alpha_i)}{2}$$

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	$K\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ $= \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma = \frac{1}{2\sigma^2}$
Sigmoid function	$K\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) = \tanh\left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)$ $\tanh\left(x\right) = \frac{e^x}{e^x + 1}$

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 $K(\mathbf{x}_i, \mathbf{x}_j)$.

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

$$i \in \arg\max_t \left\{ -y_t \nabla f\left(\alpha^k\right)_t | t \in I_{up}\left(\alpha^k\right) \right\},$$
$$j \in \arg\min_t \left\{ -\frac{b_{it}^2}{\bar{a}_{it}} | t \in I_{low}\left(\alpha^k\right), -y_t \nabla f\left(\alpha^k\right)_t < -y_i \nabla f\left(\alpha^k\right)_i \right\}$$

where

$$I_{up}\left(\alpha\right) = \left\{ t | \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1 \right\}$$

$$I_{low}\left(\alpha\right) = \left\{ t | \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1 \right\}$$

Return $B = \{i, j\}$, where $\nabla f\left(\alpha\right) = Q\alpha + \mathbf{p}$.

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:

$$\min_{\alpha_A} \frac{1}{2} \alpha_A^T Q_{AA} \alpha_A - \left(\mathbf{p}_A - Q_{AN}\alpha_N\right)^T \alpha_A$$

s. t. $0 \leq \left(\alpha_A\right)_t \leq C, t = 1, \cdots, q$

$$\mathbf{y}_A^T \alpha_A = const - \mathbf{y}_N^T \alpha_N$$

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 $m\left(\alpha^k\right) > M\left(\alpha^k\right)$. Until $m\left(\alpha^k\right) - M\left(\alpha^k\right) \leq \epsilon$ is satisfied, we can shrink variables in the following set:

$$\left\{ t | -y_t \nabla f\left(\alpha_t\right) > m\left(\alpha^k\right), \alpha_t = C, y_t = 1 \text{ or } \alpha_t = 0, y_t = -1 \right\} \cup$$
$$\left\{ t | -y_t \nabla f\left(\alpha_t\right) < M\left(\alpha^k\right), \alpha_t = 0, y_t = 1 \text{ or } \alpha_t = C, y_t = -1 \right\}$$

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

$$m\left(\alpha^k\right) \leq M\left(\alpha^k\right) + 10\epsilon$$

After reconstructing the gradient, we restore some of the previously shrunk variables based on the relationship

$$\left\{ t | - y_t \nabla f\left(\alpha_t\right) \le m\left(\alpha^k\right), \alpha_t = C, y_t = 1 \text{ or } \alpha_t = 0, y_t = -1 \right\} \cup$$
$$\left\{ t | - y_t \nabla f\left(\alpha_t\right) \ge M\left(\alpha^k\right), \alpha_t = 0, y_t = 1 \text{ or } \alpha_t = C, y_t = -1 \right\}$$

Gradient Reconstruction

To decrease the cost of reconstruction of the gradient $\nabla f\left(\alpha\right)$, 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
$y_i \ne y_j$	$\alpha_i - \alpha_j > C_i - C_j$ and $\alpha_i > C_i$	$\alpha_i^{new} = C_i$ $\alpha_j^{new} = C_i - (\alpha_i - \alpha_j)$
	$\alpha_i - \alpha_j \le C_i - C_j$ 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 $\alpha_j < 0$	$\alpha_j^{new} = 0$ $\alpha_i^{new} = (\alpha_i - \alpha_j)$
	$\alpha_i - \alpha_j \le 0$ 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 $\alpha_i > C_i$	$\alpha_i^{new} = C_i$ $\alpha_j^{new} = (\alpha_i + \alpha_j) - C_i$
	$\alpha_i + \alpha_j \le C_i$ 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. 367.)

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. 368.)

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. 369.)

► 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

$$\alpha_i^{new} = \alpha_i^{old} + \frac{y_i b_{ij}}{a_{ij}}$$
$$\alpha_j^{new} = \alpha_j^{old} - \frac{y_j b_{ij}}{a_{ij}}$$

▶ 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 $\mathbf{Q}$ 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. 367. 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 ($d \le l$).

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. 367.

▶ 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. 372.

▶ Return to step 1 unless any of the following stopping conditions is true:

  ■ $|\Delta SV| < 20$ and (Number of learned samples) $> l$

  ■ $SV \ge (d - 1)$

  ■ (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. 372..

The method is initialized by setting the subset to contain the first d records in the training data and the queue $\mathbf{Q}_S$ 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:

$$L(t) = \alpha Y(t) + (1 - \alpha) L(t - 1)$$

$$\hat{Y}_t(k) = L(t)$$

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:

$$L(t) = \alpha Y(t) + (1 - \alpha) L(t - 1)$$

$$T(t) = \alpha(L(t) - L(t - 1)) + (1 - \alpha) T(t - 1)$$

$$\hat{Y}_t(k) = L(t) + \left((k - 1) + \alpha^{-1}\right) T(t)$$

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:

$$L(t) = \alpha Y(t) + (1 - \alpha)(L(t - 1) + T(t - 1))$$

$$T(t) = \gamma(L(t) - L(t - 1)) + (1 - \gamma) T(t - 1)$$

$$\hat{Y}_t(k) = L(t) + kT(t)$$

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:

$$L(t) = \alpha Y(t) + (1 - \alpha)(L(t - 1) + \phi T(t - 1))$$

$$T(t) = \gamma(L(t) - L(t - 1)) + (1 - \gamma)\phi T(t - 1)$$

$$\hat{Y}_t(k) = L(t) + \sum_{i=1}^{k} \phi^i T(t)$$

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:

$$L(t) = \alpha(Y(t) - S(t - s)) + (1 - \alpha) L(t - 1)$$

$$S(t) = \delta(Y(t) - L(t)) + (1 - \delta) S(t - s)$$

$$\hat{Y}_t(k) = L(t) + S(t + k - s)$$

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:

$$L(t) = \alpha(Y(t) - S(t - s)) + (1 - \alpha)(L(t - 1) + T(t - 1))$$

$$T(t) = \gamma(L(t) - L(t - 1)) + (1 - \gamma)T(t - 1)$$

$$S(t) = \delta(Y(t) - L(t)) + (1 - \delta)S(t - s)$$

$$\hat{Y}_t(k) = L(t) + kT(t) + S(t + k - s)$$

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:

$$L(t) = \alpha(Y(t)/S(t - s)) + (1 - \alpha)(L(t - 1) + T(t - 1))$$

$$T(t) = \gamma(L(t) - L(t - 1)) + (1 - \gamma)T(t - 1)$$

$$S(t) = \delta(Y(t)/L(t)) + (1 - \delta)S(t - s)$$

$$\hat{Y}_t(k) = (L(t) + kT(t))S(t + k - s)$$

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 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 $S = (\mu + \beta) / (\Sigma \mu_i + \beta_i)$.

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
$\phi_p(B)$	AR polynomial of B of order p, $\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \ldots - \varphi_p B^p$
$\theta_q(B)$	MA polynomial of B of order q, $\theta_q(B) = 1 - \vartheta_1 B - \vartheta_2 B^2 - \ldots - \vartheta_q B^q$
$\Phi_P(B^s)$	Seasonal AR polynomial of BS of order P, $\Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{s2} - \ldots - \Phi_P B^{sP}$
$\Theta_Q(B^s)$	Seasonal MA polynomial of BS of order Q, $\Theta_Q(B^s) = 1 - \Theta_1 B^s - \Theta_2 B^{s2} - \ldots - \Theta_Q B^{sQ}$
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:

$$Z_t = f(Y_t),$$

$$\Delta Z_t = \mu + \sum_{i=1}^{k} \frac{Num_i}{Den_i} \Delta_i B^{b_i} f_i(X_{it}) + \frac{MA}{AR} a_t.$$

The univariate ARIMA model simply drops the predictors from the TF model; thus, it has the following form:

$$\Delta Z_t = \mu + \frac{MA}{AR} a_t$$

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 $MA = \theta_q(B)\Theta_Q(B^s)$ and the auto-regressive lag polynomial $AR = \phi_p(B)\Phi_P(B^s)$.

- 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 $\hat{N}_t(k)$ and $\sigma_t^2(k)$ be the k-step forecast and forecast variance, respectively.

Conditional Least Squares (CLS) Method

$\hat{N}_t(k) = E(N_{t+k}|N_t, N_{t-1}, \cdots)$ assuming $N_t = 0$ for t<0.

$$\sigma_t^2(k) = \sigma^2 \sum_{j=0}^{k-1} \psi_j^2$$

where ψ_j are coefficients of the power series expansion of $MA/(\Delta \times AR)$.

Minimize $S = \Sigma \left(N_t - \hat{N}_t(1) \right)^2$.

Missing values are imputed with forecast values of N_t.

Maximum Likelihood (ML) Method (Brockwell and Davis, 1991)

$\hat{N}_t(k) = E(N_{t+k}|N_t, N_{t-1}, \cdots, N_1)$

Maximize likelihood of $\left\{ N_t - \hat{N}_t(1) \right\}_{t=1}^{n}$; that is,

$$L = -\ln(S/n) - (1/n) \sum_{j=1}^{n} \ln(\eta_j)$$

where $S = \Sigma \left(N_t - \hat{N}_t(1) \right)^2 / \eta_t$, 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 $\hat{N}_t(k)$.

Error Variance

$\hat{\sigma}^2 = S/(n-k)$

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 lth 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

$$\lambda_l \approx E\left((Y_{t+l} - \varphi_1 Y_{t+l-1} - \cdots - \varphi_p Y_{t+l-p})\left(Y_t - \varphi'_1 Y_{t-1} - \cdots - \varphi'_{p+q} Y_{t-p-q}\right)\right)$$

$$= \left(\rho_l - \sum_{j=1}^{p+q} \varphi_j \rho_{l+j} - \sum_{i=1}^{p} \varphi_i \rho_{l-i} + \sum_{i=1}^{p} \sum_{j=1}^{p+q} \varphi_i \varphi_j \rho_{l+j-i}\right) c_0$$

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

$$Q\left(K\right) = n\left(n+2\right)\sum_{k=1}^{K} r_k^2/\left(n-k\right)$$

where r_k is the kth lag ACF of residual.

Q(K) is approximately distributed as $\chi^2\left(K-m\right)$, 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

$$Y(t) = U(t) + wS_T(t)$$

where $S_T(t) = \frac{1}{1-B}I_T(t) = \begin{cases} 0 & t < T \\ 1 & t \geq T \end{cases}$ is a step function.

TC (Temporary/Transient Change)

Assuming that a TC outlier occurs at time $t=T$, then

$$Y(t) = U(t) + wD_T(t)$$

where $D_T(t) = \frac{1}{1-\delta B}I_T(t), 0 < \delta < 1$ 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 $e(t)$ be the residual and $x(t) = \pi(B) L(B) \Delta I_T(t)$, so:

$$e(t) = wx(t) + a(t)$$

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:

$$\lambda_j(T) = \frac{w_j(T)}{\sqrt{\mathrm{Var}(w_j(T))}}$$

Under the null hypothesis of no outlier, $\lambda_j(T)$ 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:

$$\chi^2\left(T\right) = \frac{\mathbf{w}'\left(T\right)\left(X_T X_T\right)\mathbf{w}\left(T\right)}{\sigma^2}$$

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
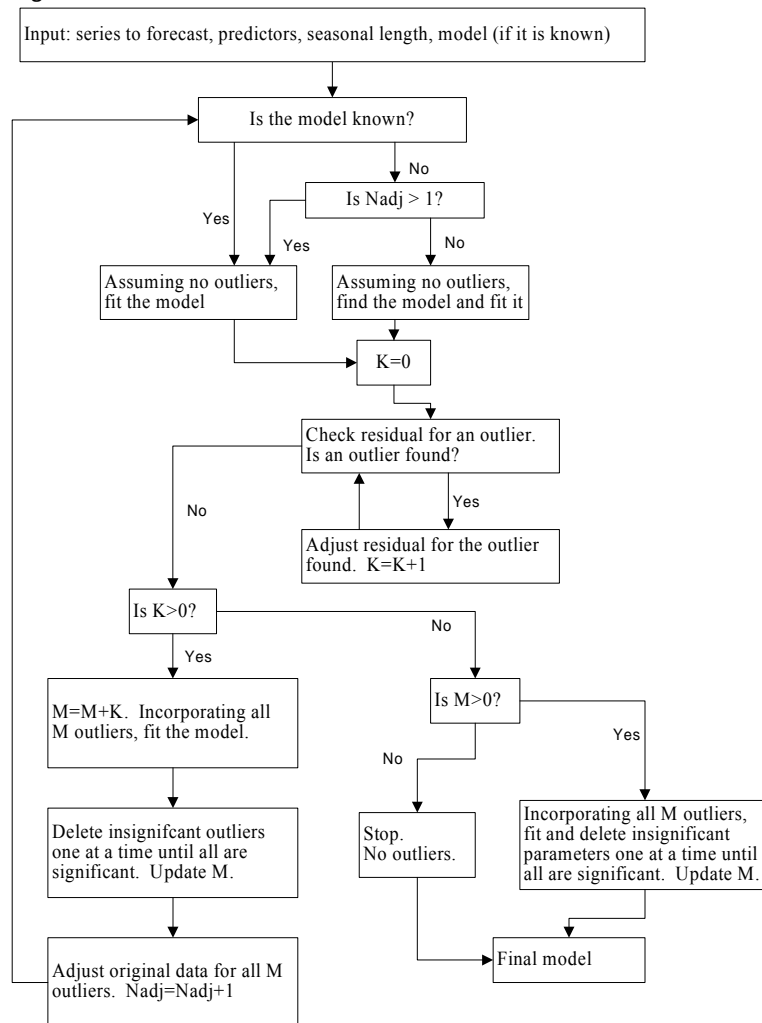
Goodness-of-Fit Statistics

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

$$MAPE = \frac{100}{n} \Sigma \left| \left(Y\left(t \right) - \hat{Y}\left(t \right) \right) / Y\left(t \right) \right|$$

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

$$MAE = \frac{1}{n}\Sigma \left| Y\left(t\right) - \hat{Y}\left(t\right) \right|$$

Maximum Absolute Error

$$MaxAE = \max \left(\left| Y\left(t\right) - \hat{Y}\left(t\right) \right| \right)$$

Normalized Bayesian Information Criterion

$$\text{Normalized } BIC = \ln\left(MSE\right) + k\frac{\ln(n)}{n}$$

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).

$$R_S^2 = 1 - \frac{\displaystyle\sum_t \left(Z\left(t\right) - \hat{Z}\left(t\right)\right)^2}{\displaystyle\sum_t \left(\Delta Z\left(t\right) - \overline{\Delta Z}\right)^2}$$

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 $\left(-\infty, 1\right]$. 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

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

Fit the model by ML → Delete insignificant parameters

Modify model (only once)

Diagnostic checking Ljung-Box, ACF/PACF
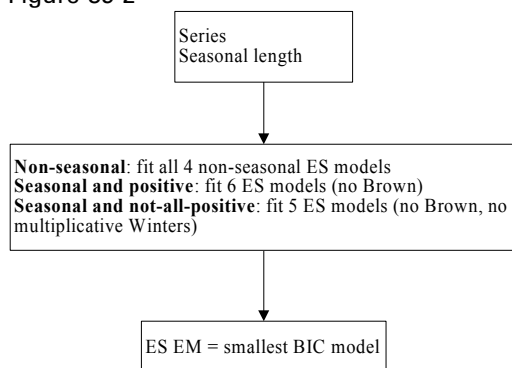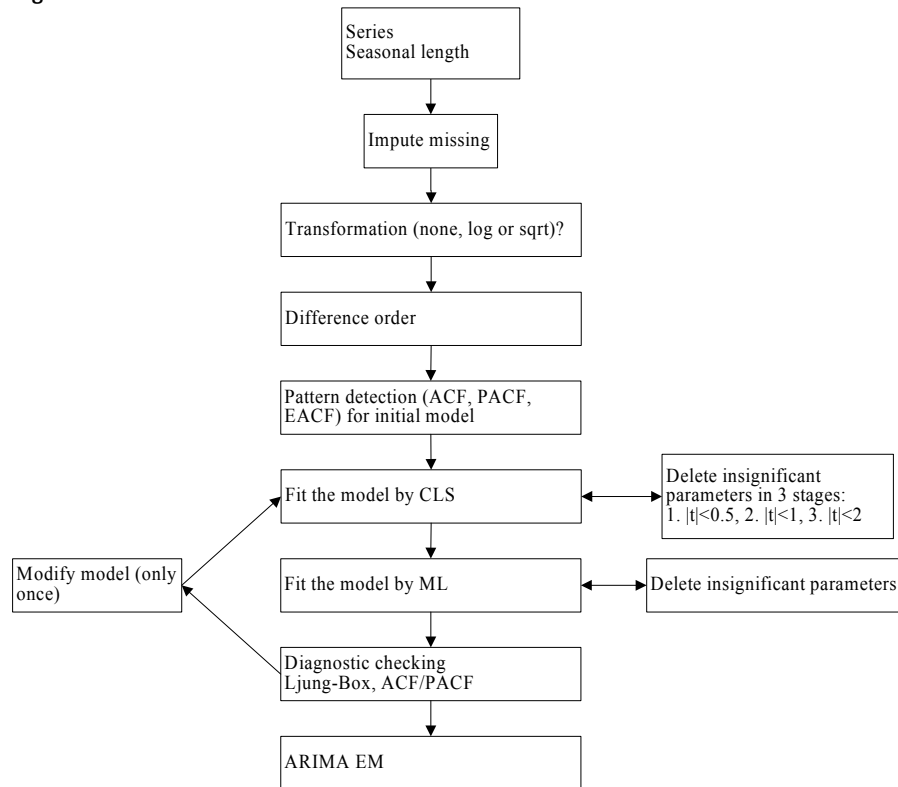
ARIMA EM

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. 389.

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
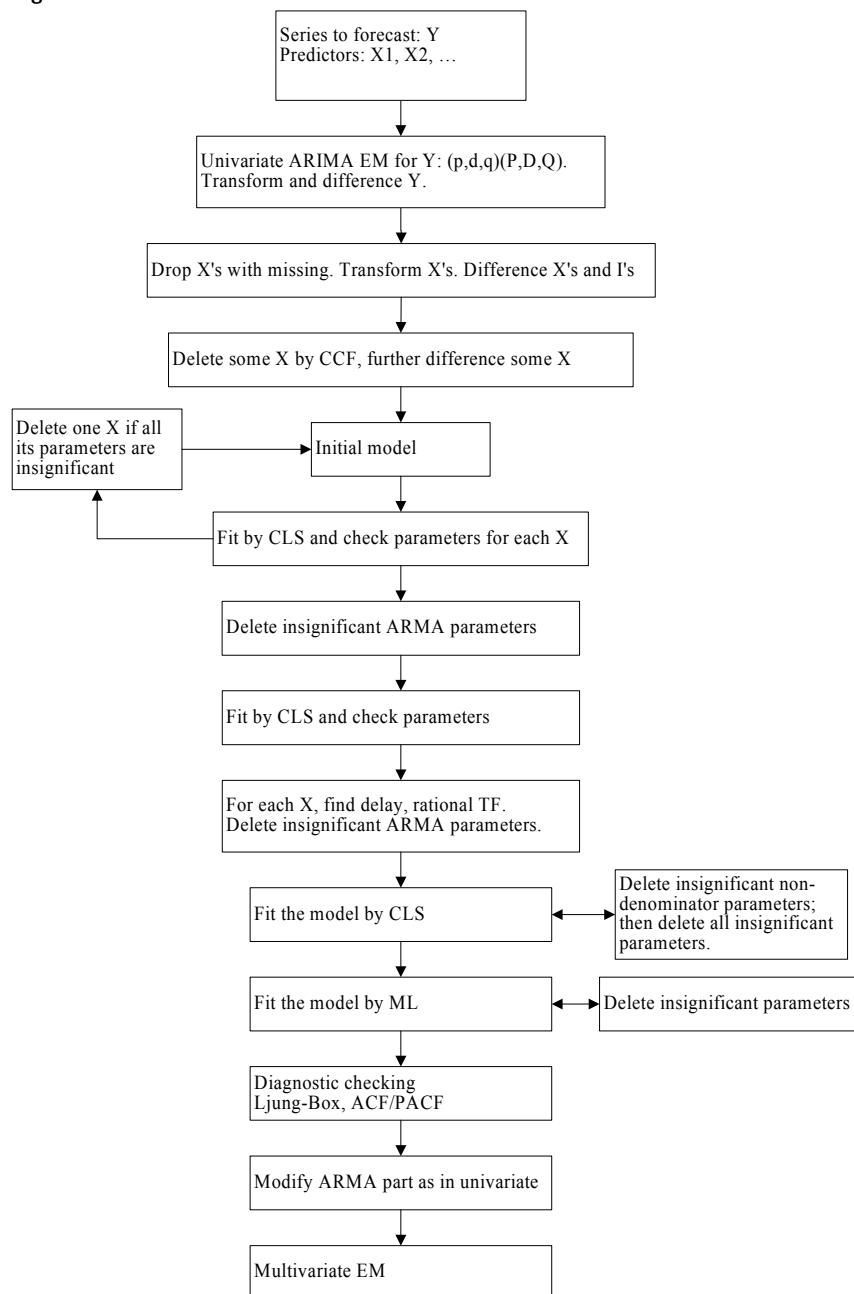
Note: For short series, n<max(20,3s), fit a univariate expert model.

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. 379.

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. 375. For information on forecasting in ARIMA models, see "Estimation and Forecasting of ARIMA/TF" on p. 379.

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. 393.) 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 kth 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 lth category of the kth symbolic field,

$\hat{\sigma}_k^2$ is the estimated variance of the kth continuous variable for all records,

$\hat{\sigma}_{vk}^2$ is the estimated variance of the kth continuous variable for records in the vth 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_{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 $d_{\min}(C)$ is the minimum inter-cluster distance for cluster model C.

▶ Now from model C_{k-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. 394.

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 $null$.

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.

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.

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: .

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.

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.

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.

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.

Rumelhart, D. E., J. L. McClelland, and . The PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*. Cambridge, MA: MIT 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.

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

score statistic
  Cox regression algorithms, 104
score test
  multinomial logistic regression algorithms, 255
score values (CHAID), 88
scoring
  Decision List algorithm, 117
  in anomaly detection, 7
self-learning response model algorithms, 357
  information measure, 361
  model assessment, 358
  predictor importance, 360–361
  scoring, 359
  updating the model, 359
sensitivity analysis
  in neural networks, 301
separation
  checking for in logistic regression, 249
sequence rules, 355
  adjacency lattice, 350
  antecedents, 350
  blank handling, 354, 356
  Carma algorithm, 351
  confidence, 354, 356
  consequents, 350
  frequent sequences, 349
  gap, 349
  itemsets, 347
  length of sequences, 348
  maximal sequences, 348
  overview, 347
  predictions, 355
  sequential patterns, 353
  size of sequences, 348
  subsequences, 348
  support, 348
  timestamp tolerance, 349
  transactions, 347
sequences
  in sequence rules, 348
sequential Bonferroni
  generalized linear mixed models algorithms, 209
sequential minimal optimization (SMO) algorithm
  support vector machines (SVM), 367
sequential sidak
  generalized linear mixed models algorithms, 209
set encoding value
  in k-means, 234
sigmoid kernel function (SVM), 367
sigmoid transfer function
  in neural networks, 293
Silhouette coefficient
  cluster evaluation algorithms, 95
simplemax
  neural network confidence, 316
size
  of sequences, 348

softmax
  neural network confidence, 303
softmax activation function
  multilayer perceptron algorithms, 306
splitting
  of merged categories (CHAID), 82
splitting nodes
  CHAID, 83
stagewise additive modeling
  boosting algorithms, 135
standardized residuals
  logistic regression algorithms, 266
stepping
  in logistic regression, 249
stepwise field selection
  in regression, 344
stepwise selection
  Cox regression algorithms, 104
stopping rules
  in C&RT, 71
  in CHAID, 87
  in QUEST, 335
  multilayer perceptron algorithms, 311
studentized residuals
  linear modeling algorithms, 288
  logistic regression algorithms, 266
subpopulations, 246
subsequences, 348
sum of squares between
  cluster evaluation algorithms, 95
sum of squares error
  cluster evaluation algorithms, 95
  multilayer perceptron algorithms, 306
support
  sequence rules, 348
support vector machines (SVM), 363
  ε-Support Vector Regression (ε-SVR), 364
  algorithm notation, 363
  blank handling, 373–374
  C-support vector classification, 364
  decision function constant, 366
  fast training algorithm, 371
  gradient reconstruction, 369
  kernel functions, 367
  model building algorithm, 367
  parallel optimization, 371
  predicted probabilities, 373
  predictions, 373
  queue method, 372
  scoring, 373
  sequential minimal optimization (SMO) algorithm, 367
  sequential optimization, 372
  shrinking, 368
  SMO decomposition, 370
  solving quadratic problems, 365
  subset selection, 372
  types of SVM models, 364