CNLR

CNLR is used to estimate the parameters of a function by minimizing a smooth nonlinear loss function (objective function) when the parameters are subject to a set of constraints.

Model

Consider the model

$$f = f(\underset{\sim}{x}, \underset{\sim}{\theta})$$

where $\underset{\sim}{\theta}$ is a $p \times 1$ parameter vector, $\underset{\sim}{x}$ is an independent variable vector, and f is a function of $\underset{\sim}{x}$ and $\underset{\sim}{\theta}$.

Goal

Find the estimate $\underset{\sim}{\theta}^*$ of $\underset{\sim}{\theta}$ such that $\underset{\sim}{\theta}^*$ minimizes

$$F = F(y, f)$$

subject to

$$\underset{\sim}{l} \leq \begin{Bmatrix} \underset{\sim}{\theta} \\ \mathbf{A}_L \underset{\sim}{q} \\ \mathbf{C}(\underset{\sim}{\theta}) \end{Bmatrix} \leq \underset{\sim}{u}$$

where F is the smooth loss function (objective function), which can be specified by the user. $\mathbf{A}_L$ is an $m_L \times p$ matrix of linear constraints, and $\mathbf{C}(\underset{\sim}{\theta})$ is an $m_N \times 1$ vector of nonlinear constraint functions. $\underset{\sim}{l}' = (\underset{\sim}{l}_B', \underset{\sim}{l}_L', \underset{\sim}{l}_N')$, where $\underset{\sim}{l}_B'$, $\underset{\sim}{l}_L'$,

and $\underset{\sim}{l}'_N$ represent the lower bounds, linear constraints and nonlinear constraints, respectively. The upper bound $\underset{\sim}{u}$ is defined similarly.

Algorithm

CNLR uses the algorithms proposed and implemented in NPSOL by Gill, Murray, Saunders, and Wright. A description of the algorithms can be found in the *User's Guide for NPSOL*, Version 4.0 (1986).

The method used in NPSOL is a sequential quadratic programming (SQP) method. For an overview of SQP methods, see Gill, Murray, and Wright (1981), pp. 237–242.

The basic structure of NPSOL involves major and minor iterations. Based on the given initial value $\underset{\sim}{\theta}^{(0)}$ of $\underset{\sim}{\theta}$, the algorithm first selects an initial working set that includes bounds or general inequality constraints that lie within a crash tolerance (CRSHTOL). At the *k*th iteration, the algorithm starts with

(I) Minor Iteration

This iteration searches for the direction P_k, which is the solution of a quadratic subproblem; that is, P_k, is found by minimizing

$$\mathbf{g}'_k\mathbf{P} + \tfrac{1}{2}\mathbf{P}'\mathbf{H}_k\mathbf{P} \qquad \textbf{(1)}$$

subject to

$$\underset{\sim}{\bar{l}}^{(k)} \le \begin{Bmatrix} \mathbf{P} \\ \mathbf{A}_L\mathbf{P} \\ \mathbf{A}_N\mathbf{P} \end{Bmatrix} \le \underset{\sim}{\bar{u}}^{(k)}$$

where $\mathbf{g}_k$ is the gradient of F at $\underset{\sim}{\theta}^{(k)}$, the matrixp $\mathbf{H}_k$ is a positive-definite quasi-Newton approximation to the Hessian of the Lagrangian function, $\mathbf{A}_N$ is the Jacobian matrix of the nonlinear-constraint vector C evaluated at $\underset{\sim}{\theta}^{(k)}$, and

$$\bar{\underset{\sim}{\mathbf{l}}}^{(k)} = \left(\bar{\mathbf{l}}_B', \bar{\mathbf{l}}_L', \bar{\mathbf{l}}_N'\right)$$

$$\bar{\underset{\sim}{\mathbf{u}}}^{(k)} = \left(\bar{\mathbf{u}}_B', \bar{\mathbf{u}}_L', \bar{\mathbf{u}}_N'\right)$$

$$\bar{\mathbf{l}}_B = \mathbf{l}_B - \underset{\sim}{\mathbf{q}}^{(k)}$$

$$\bar{\mathbf{l}}_L = \mathbf{l}_L - \mathbf{A}_L\, \underset{\sim}{\mathbf{q}}^{(k)}$$

$$\bar{\mathbf{l}}_N = \mathbf{l}_N - \mathbf{C}(\underset{\sim}{\mathbf{q}}^{(k)})$$

$$\bar{\mathbf{u}}_B = \mathbf{u}_B - \underset{\sim}{\mathbf{q}}^{(k)}$$

$$\bar{\mathbf{u}}_L = \mathbf{u}_L - \mathbf{A}_L\, \underset{\sim}{\mathbf{q}}^{(k)}$$

$$\bar{\mathbf{u}}_N = \mathbf{u}_N - \mathbf{C}(\underset{\sim}{\mathbf{q}}^{(k)}).$$

The linear feasibility tolerance, the nonlinear feasibility tolerance, and the feasibility tolerance are used to decide if a solution of **(1)** is feasible for linear and nonlinear constraints.

Once the search direction P_k is found, the algorithm goes to the major iteration.

(II) Major Iteration

The purpose of the major iteration is to find a non-negative scalar α_k such that

$$\underset{\sim}{\theta}^{(k+1)} = \underset{\sim}{\theta}^{(k)} + \alpha_k \mathbf{P}_k$$

satisfies the following conditions:

- $\underset{\sim}{\theta}^{(k+1)}$ produces a "sufficient decrease" in the augmented Lagrangian merit function

$$L(\underset{\sim}{\theta}, \underset{\sim}{\lambda}, \underset{\sim}{s}) = F(\underset{\sim}{\theta}) - \sum_i \lambda_i \left(c_i(\underset{\sim}{\theta}) - s_i \right) + \frac{1}{2} \sum_i \rho_i \left(c_i(\underset{\sim}{\theta}) - s_i \right)^2 \qquad \textbf{(2)}$$

  The summation terms in **(2)** involve only the nonlinear constraints. The vector λ is an estimate of the Lagrange multipliers for the nonlinear constraints. The non-negative slack variables $\{s_i\}$ allow nonlinear inequality constraints to be treated without introducing discontinuities. The solution of the *QP* subproblem defined in **(1)** provides a vector triple that serves as a direction search for $\underset{\sim}{\theta}$, $\underset{\sim}{\lambda}$ and $\underset{\sim}{s}$. The non-negative vector of penalty parameters (ρ) is initialized to zero at the beginning of the first major iteration. Function precision criteria are used as a measure of the accuracy with which the functions *F* and c_i can be evaluated.

- $\underset{\sim}{\theta}^{(k+1)}$ is close to a minimum of *F* along $\mathbf{P}_k$. The criterion is

$$\left| \mathbf{g}'(\underset{\sim}{\theta}^{(k+1)}) \mathbf{P}_k \right| < -\eta \mathbf{g}_k' \mathbf{P}_k$$

  where η is the Line Search Tolerance and $0 \le \eta < 1$. The value of η determines the accuracy with which α_k approximates a stationary point of *F* along P_k. A smaller value of η produces a more accurate line search.

- The step length is in a certain range; that is,

$$\left\| \underset{\sim}{\theta}^{(k+1)} - \underset{\sim}{\theta}^{(k)} \right\| = \left\| \alpha_k \mathbf{P}_k \right\| \le \text{Step Limit}.$$

(III) Convergence Tests

After α_k is determined from the major iteration, the following conditions are checked:

- $k+1 \leq$ Maximum number of major iterations
- The sequence $\{\underset{\sim}{\theta}^{(l)}\}$ converged at $\underset{\sim}{\theta}^{(k+1)}$; that is,

$$\left\|\alpha_k \mathbf{P}_k\right\| \leq \sqrt{r}\left(1+\|\underset{\sim}{\theta}^{(k+1)}\|\right)$$

- $\underset{\sim}{\theta}^{(k+1)}$ satisfies the Kuhn-Tucker conditions to the accuracy requested; that is,

$$\|\mathbf{g}_z(\underset{\sim}{\theta}^{(k+1)})\| \leq \sqrt{r}\left(1+\max\left(1+|F(\underset{\sim}{\theta}^{(k+1)})|, \|g(\underset{\sim}{\theta}^{(k+1)})\|\right)\right)$$

and

$$\left\|\text{res}_j\right\| \leq \text{FTOL}, \text{ for all } j,$$

where $\mathbf{g}_z$ is the projected gradient, $\mathbf{g}$ is the gradient of F with respect to the free parameters, res_j is the violation of the jth nonlinear constraint, FTOL is the Nonlinear Feasibility Tolerance, and r is the Optimality Tolerance.

If none of these three conditions are satisfied, the algorithm continues with the Minor Iteration to find a new search direction.

Termination

The following are termination conditions.

- Underflow. A single underflow will always occur if machine constants are computed automatically. Other floating-point underflows may occur occasionally, but can usually be ignored.

- Overflow. If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the *j*th constraint, it may be possible to avoid the difficulty by increasing the magnitude of FTOL, LFTOL, or NFTOL and rerunning the program. If the message recurs after this change, the offending linearly dependent constrains (with index "*j*") must be removed from the problem.

- Optimal solution found.

- Optimal solution found, but the requested accuracy could not be achieved, NPSOL terminates because no further improvement can be made in the merit function. This is probably caused by requesting a more accurate solution than is attainable with the given precision of the problem (as specified by FPRECISION).

- No point has been found that satisfies the linear constraints. NPSOL terminates without finding a feasible point for the given value of LFTOL. The user should check that there are no constraint redundancies and ensure that the value of LFTOL is greater than the precision of parameter estimates.

- No point has been found which satisfies the nonlinear constraints. There is no feasible point found in *QP* subproblems. The user should check the validity of constraints. If the user is convinced that a feasible point does exist, NPSOL should be restarted at a different starting point.

- Too many iterations. If the algorithm appears to be making progress, increase the value of ITER and rerun NPSOL. If the algorithm seems to be "bogged down", the user should check for incorrect gradients.

- Cannot improve on current point. A sufficient decrease in the merit function could not be attained during the final line search. This sometimes occurs because an overly stringent accuracy has been requested; for example, Optimality Tolerance is too small or a too-small step limit is given when the parameters are measured on different scales.

Please note the following:

- Unlike the other procedures in SPSS, the weight function is not treated as a case replicate in CNLR.

- When both weight and loss function are specified, the algorithm takes the product of these two functions are the loss function.

- If the loss function is not specified, the default loss function is a squared loss function and the default output in NLR will be printed. However, if the loss function is not a squared loss function, CNLR prints only the final parameter estimates, iteration history, and termination message. In order to obtain estimates of the standard errors of parameter estimates and correlations between parameter estimates, the bootstrapping method can be requested.

Bootstrapping Estimates

Bootstrapping is a nonparametric technique of estimating the standard error of a parameter estimate using repeated samples from the original data. This is done by sampling with replacement. CNLR computes and saves the parameter estimates for each sample generated. This results, for each parameter, in a sample of estimates from which the standard deviation is calculated. This is the estimated standard error.

Mathematically, the bootstrap covariance matrix **S** for the *p* parameter estimates is

$$\mathbf{S} = \left(s_{ij}\right)_{p\times p}$$

where

$$s_{ij} = \sum_{k=1}^{m}\left(\hat{\theta}_{ik} - \overline{\theta}_i\right)\left(\hat{\theta}_{jk} - \overline{\theta}_j\right)$$

$$\overline{\theta}_i = \sum_{k=1}^{m}\hat{\theta}_{ik}\big/m$$

and $\hat{\theta}_{ik}$ is the CNLR parameter estimate of θ_i for the *k*th bootstrap sample and *m* is the number of samples generated by the bootstrap. By default, $m = \dfrac{10p(p+1)}{2}$. The standard error for the *j*th parameter estimate is estimated by

$$\sqrt{\frac{s_{jj}}{m-1}}$$

and the correlation between the *i*th and *j*th parameter estimates is estimated by

$$\frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}}$$

The "95% Trimmed Range" values are the most extreme values that remain after trimming from the set of estimates for a parameter, the g largest, and the g smallest estimates, where g is the largest integer not exceeding $0.025m$.

References

Gill, P. E., Murray, W. M., and Wright, M. H. 1981. *Practical Optimization*. London: Academic Press.

Gill, P. E., Murray, W. M., Saunders, M. A., and Wright, M. H. 1984. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Transactions on Mathematical Software*, **10**(3): 282–296.

Gill, P. E., Murray, W. M., Saunders, M. A., and Wright, M. H. 1986. *User's guide for NPSOL* (*version* 4.0): *A fortran package for nonlinear programming*. Technical Report SOL 86–2, Department of Operations Research, Stanford University.