

Algorithms

This chapter describes how to approach the optimization problems posed by empirical likelihood. The emphasis is on computing the empirical likelihood, for statistics defined through estimating equations, with nuisance parameters and side constraints. Much of the discussion carries over to other nonparametric likelihoods.

Several optimization methods are presented. Choosing a method entails some trade-offs. The Newton-Lagrange algorithm appears to be fastest. A nested search algorithm is slower but more reliable. The most reliable algorithm tried so far is sequential quadratic programming, using the NPSOL code (see Chapter 12.7). It also handles censoring and truncation in a direct way. But NPSOL is slower than special purpose code could be, because for problems without censoring or truncation, the Hessian is very sparse and NPSOL does not exploit that sparsity. Sequential linearization methods appear to be both fast and reliable, but they only compute an approximation to the desired likelihood. An approach based on range space methods has the potential to be as reliable as sequential quadratic programming, but much faster.

There is also a choice of which likelihood to use. The empirical likelihood respects range restrictions on parameters and is transformation invariant, but only provides positive likelihood if 0 is in the convex hull of $m(X_i, \theta)$. This convex hull constraint may be unduly restrictive when n is small, or the number of parameters is large. Alternatives such as the Euclidean likelihood can escape the convex hull, but do not obey range constraints. The empirical likelihood-t approach from Chapter 10.4 has some attractive properties, but it has not as yet been thoroughly explored. Empirical entropy methods do not escape the convex hull, but they do allow distributions with weight 0 on one or more data points to contribute to the confidence region.

Numerical searches typically require starting values in order to succeed. Usually the MLE $\hat{\theta}$ corresponds to solving the estimating equations with equal weight on every observation, and for most statistics of interest we can find $\hat{\theta}$ by some pre-existing algorithm. The profiles and hypothesis tests of interest are best found through a sequence of optimization problems starting at $\hat{\theta}$. There can also be a need to rescale the problem. A cubic regression of Y on $X/1000$ may succeed where a cubic regression on X fails. Similarly, a B-spline basis is a better choice than the truncated power spline basis when one wants a spline.

For this text, it was desired to use exact likelihood computations instead of the

approximate ones, and NPSOL was used, except for some problems with quantiles that allow special methods. In the author's experience NPSOL has never failed to compute the empirical likelihood, if $\hat{\theta}$ has been available, though sometimes the problem must be rescaled. NPSOL often succeeds without $\hat{\theta}$. For small n it is best to use NPSOL directly, while for larger n , speed considerations lead to the use of NPSOL in the outer problem of the nested algorithm described in Chapter 12.3. While NPSOL was reliable without using explicit second derivatives, an algorithm using those derivatives might be faster or might be less sensitive to starting values and scaling.

12.1 Statistical tasks

The inferential problems to be addressed are testing hypotheses, computing maximum empirical likelihood estimates (MELE's), profiling likelihood ratio functions, and finding confidence sets.

The basic chore in all of these tasks is to maximize $R(F) = \sum_{i=1}^n \log(nw_i)$ subject to constraints

$$\begin{aligned} w_i &\geq 0, \\ \sum_{i=1}^n w_i &= 1, \\ \sum_{i=1}^n w_i m(X_i, \theta, \eta) &= 0, \\ C(\theta, \eta, \gamma, \delta) &= 0. \end{aligned}$$

The second last equation describes the estimating equations, defined in terms of observations X_i , the parameter vector θ , and another parameter vector η not previously discussed. For autoregressive models described in Chapter 8.2, m also depends on the index i , but we suppress that dependence here for simplicity of exposition. There are also side constraints on θ and η expressed through the vector-valued function C of θ , η , and two new parameter vectors γ and δ .

It is assumed that the estimating function m can be differentiated at least once with respect to the components of θ , but not necessarily with respect to η . Optimization with respect to components of θ may then be approached by standard smooth function optimization methods, but components of η are either not to be optimized over, or must be optimized over by other means, such as grid searches. For example, a tail probability could be a part of θ , while the corresponding quantile would be part of η . As a more complicated example, a conditional quantile may be expressed through

$$m(X, \theta, \eta) = I(X_{\eta_1} = \eta_2)(I(X_{\eta_3} \leq \eta_4) - \theta_1),$$

where to avoid deeply nested subscripts, 1_A has been written as $I(A)$. Conditionally on component η_1 of $X \in \mathbb{R}^d$ taking the value η_2 , component η_3 of X

has quantile η_4 with corresponding tail probability θ_1 . We might optimize over θ_1 with η fixed, or optimize over η_4 with θ_1 and the rest of η fixed. For a given data set there may not be a need to optimize over η_2 , or especially over the indices η_1 and η_3 .

The function C describes constraints on the parameters θ and η , introducing two new parameter vectors γ and δ . We assume that C is differentiable at least once with respect to γ (and θ), but not necessarily with respect to δ (or η). The reason to keep C separate from m is that it does not depend on X_i . The function C has to be computed only once where m has to be computed n times. If n is large or C is expensive, the time saving can be large. An example of a side constraint is that a linear combination of θ values takes a particular value:

$$C = \sum_{j=1}^p \theta_j \gamma_j - \gamma_{p+1} = 0.$$

If γ_{p+1} is supposed to be a response variable for observation i then we can use $\gamma_1, \dots, \gamma_p$ and $\delta_1 = i$, and encode the constraint as $\sum_{j=1}^p \theta_j \gamma_j - Y_{\delta_1}$.

A hypothesis test may be conducted by solving the problem above with some part of the list $(\theta, \eta, \gamma, \delta)$ fixed at defining values and the rest of it free to vary. The same problem arises in computing an MELE. To profile one or more members of the list $(\theta, \eta, \gamma, \delta)$, we conduct a sequence of hypothesis tests, in which some parameter components are frozen permanently at defining values, others change as we step through the sequence of hypothesis tests but are fixed in any one test, while the remainder are optimized over in each hypothesis test.

A confidence region is most easily found by computing a profile, and observing where the profile likelihood ratio function is above a threshold used for the confidence region. For a confidence interval the problem may also be approached by alternately maximizing and minimizing θ_j (or γ_j) subject to constraints including $\sum_{i=1}^n \log(nw_i) \geq \log(r_0)$.

To compute a univariate profile of θ_j , η_j , γ_j , or δ_j , we start at the MLE (or MELE) and solve a sequence of problems in which the parameter increases by steps. Then we return to the starting point and solve a sequence of problems in which the parameter decreases in a sequence of steps. The steps continue until either the desired range is covered, or the likelihood has fallen below a threshold.

Suppose that θ_1 has been profiled by maximizing empirical likelihood with θ_1 fixed at values $\theta_1(g)$ for a grid indexed by $g = 1, \dots, G$. The profile trace contains the record of concomitant values

$$(\theta(g), \eta(g), \gamma(g), \delta(g), \mathcal{R}(g)), \quad g = 1, \dots, G,$$

and possibly other quantities of interest, as computed during the optimizations.

For some parameters η_j and δ_j , such as observation component indices, a profile is not statistically meaningful. In other settings a profile can be used to generate the MLE of a curve. For example, if θ_1 is a tail probability corresponding to a quantile η_1 then profiling η_1 generates the empirical CDF in the profile trace

$(\theta_1(g), \eta_1(g))$. Of course, the CDF is easy to compute directly. But if we seek a CDF subject to some other estimating equation constraints, then computing it by profiling the quantile is attractive.

A bivariate profile may be computed by a sequence of problems in which the point (θ_j, θ_k) varies over a grid in the plane. The boundaries of this grid may be chosen after inspecting the two uni-parameter profiles to find the ranges of interest. It is advantageous to use a grid that contains the MLE $(\hat{\theta}_j, \hat{\theta}_k)$, which we take as the origin of the grid. Starting at the origin, we compute the empirical likelihood function outward in each of the four axis directions, with the solution at each grid point supplying starting values for the next one out. For every other cell in the grid define its predecessor as the cell that is one unit towards the MLE in the j direction and one unit towards the MLE in the k direction. These other cells can be visited in any order at all, so long as each cell is visited after its predecessor has been visited. This iteration allows starting values to be taken from the predecessors. If we anticipate a unimodal likelihood, then some speed can be gained by not computing at a cell whose predecessor had a very small likelihood.

The iteration above is how the bivariate profiles for this text were computed. Profiles of (γ_j, γ_k) or (θ_j, γ_k) are similar to the ones described above. Profiles involving statistically meaningful quantities η_j or δ_k might be approached this same way. Having computed the empirical log likelihood on a grid, the resulting values can be used in contour plots and perspective plots.

Some other strategies may be necessary to follow very twisty or narrow confidence regions, especially those from undetermined parameter vectors. It might also be faster to find and follow the contours of interest directly.

12.2 Smooth optimization

This section surveys numerical optimization of smooth functions. The texts cited in the bibliographic notes of Chapter 12.7 provide more detail. Parameters like η and δ are assumed to be fixed during any smooth optimization. Accordingly, we absorb η and δ into the definitions of m and C .

Basic smooth optimization methods are variations of Newton's method. They typically converge to local minima. If, however, the function is known to be convex, then a local minimum is a global one, and if the function is strictly convex, then a local minimum is the unique global minimum.

Let $f(x)$ be a real-valued function over $x \in \mathbb{R}^p$, and suppose that f is twice continuously differentiable. The gradient of f is

$$g(x) = \frac{\partial}{\partial x} f(x),$$

interpreted here as a column vector, and the Hessian of f is

$$H(x) = \frac{\partial^2}{\partial x \partial x'} f(x),$$

a p by p matrix of second order partial derivatives of f .

The basic method for optimizing f is Newton's method. Starting with a value x_0 , the Newton iteration takes

$$x_{k+1} = x_k - H(x_k)^{-1}g(x_k), \quad k \geq 0.$$

To simplify the notation, write $H_k = H(x_k)$ and $g_k = g(x_k)$. The typical behavior for Newton's method is quadratic convergence to a solution x_∞ of the equations $g(x) = 0$, if x_0 is close enough to x_∞ . Quadratic convergence means that $\limsup_{k \rightarrow \infty} \|x_{k+1} - x_\infty\| / \|x_k - x_\infty\|^2 \in (0, \infty)$. Roughly speaking, the number of correct bits or digits in x_{k+1} is twice the number in x_k . It is not uncommon for Newton's method to require only four or five iterations to converge in double precision.

The simplest version of Newton's method is not reliable. One problem is that Newton's method converges quadratically to a solution x_∞ that could be a local minimum, a local maximum, or a saddlepoint of f . A second problem is that if H_k is nearly singular, then a very large step can be taken and the algorithm might never recover.

The simplest usable versions of Newton's method apply some checks to the value $x_k - H_k^{-1}g_k$ before accepting it as x_{k+1} . A step-halving approach takes $x_{k+1} = x_{k+1,r} = x_k - 2^{-r}H_k^{-1}g_k$ where r is the smallest nonnegative integer for which $f(x_{k+1,r}) < f(x_k)$. A variant is to take $x_{k+1,r} = x_k - [H_k + \lambda_r D_k]^{-1}g_k$, where λ_r is an increasing sequence of nonnegative numbers, and D is a positive definite matrix such as the identity or perhaps the diagonal of H_k . This generalizes the Levenberg-Marquardt method, familiar in nonlinear least squares. As r increases, the step $[H_k + \lambda_r D_k]^{-1}g_k$ becomes shorter, and if $D = I$ it becomes more nearly parallel to the steepest descent direction $-g_k$.

More sophisticated step reduction approaches insist on a sufficient decrease in f . The decrease in f must be comparable to that predicted through a quadratic Taylor approximation $f \doteq f_k + (x - x_k)'g_k + (x - x_k)'H_k(x - x_k)/2$. If the decrease is not comparable, then the Taylor approximation is called into question for the vector $x_{k+1,r}$, and a shorter vector $x_{k+1,r+1}$ is tried.

When the Hessian is expensive to compute or difficult to program, then quasi-Newton methods are often used. Quasi-Newton methods do not compute the Hessian, but approximate it instead by using the computed values of g_k . If x_{k+1} is close to x_k then $g_{k+1} - g_k$ is approximately $H'_{k+1}(x_{k+1} - x_k)$. Starting with an approximation H_0 such as the identity matrix, each new gradient value g_{k+1} that is observed is used to make an update to H_{k+1} so that

$$(x_{k+1} - x_k)'H'_{k+1}(x_{k+1} - x_k) = (x_{k+1} - x_k)'(g_{k+1} - g_k),$$

making H_{k+1} consistent with the gradient information from step $k + 1$.

In some cases the Hessian is simply too large to store, whether exactly or approximately. If sparse methods are not available, then the method of conjugate gradients is a candidate.

Constrained optimization is a much deeper topic than unconstrained optimization. A brief sketch of the subject is given below. The reader may consult refer-

ences in Chapter 12.7 for more details. We only consider general smooth nonlinear equality constraints, as these are the kind that arise in most empirical likelihood problems. Special methods exist to exploit simpler cases such as bounds on parameters or linear constraints. Other specialized methods have been developed for inequality constraints.

Minimization of $f(x)$ subject to nonlinear equality constraints $c(x) = 0$, is usually organized around the Lagrangian

$$G(x, \lambda) = f(x) + \lambda'c(x).$$

The constrained optimum is described by values x and λ with $\partial G/\partial x = 0$ and $\partial G/\partial \lambda = 0$. If we knew the Lagrange multiplier λ , then the solution would be a stationary point x with $\partial G(x, \lambda)/\partial x = 0$. The stationary point is not necessarily a minimum of $G(\cdot, \lambda)$.

Augmented Lagrangian methods replace G by $G_\rho = G + (\rho/2)c(x)'c(x)$ for some $\rho > 0$. Some versions use a different ρ_i for each constraint in $c(x)$. For large enough ρ , the constrained optimum is a local minimum of G_ρ . The augmented Lagrangian works better than simplistic penalty methods that seek to minimize $P_\rho = f(x) + (\rho/2)c(x)'c(x)$, because these latter require $\rho \rightarrow \infty$ and the Hessian of P_ρ becomes badly conditioned for large ρ .

Sequential quadratic programming (SQP) methods may be applied to the augmented Lagrangian function. In SQP, given λ and ρ , the function $G_\rho(x, \lambda)$ is approximated by a quadratic in x , and the constraint $c(x)$ is approximated as a linear function in x . A solution is sought minimizing the quadratic function subject to linear constraints.

Practical implementations of SQP must have means for detecting defective subproblems, such as those that are unbounded below due to an indefinite Hessian. Augmented Lagrangian methods must also have methods for updating ρ and λ . The convergence rate for x is limited by that of λ .

The NPSOL software uses sequential quadratic programming with an augmented Lagrangian. The user of the code needs to provide starting values, routines to compute f and c , and ideally, routines to compute gradients of f and c with respect to x .

12.3 Estimating equation methods

Here we consider optimization of empirical likelihood subject to a constraint fixing some but not all of the parameters in the estimating equation. This formulation is the most commonly used one in this text. The parameters that are free to vary in the optimization may be nuisance parameters that had to be introduced in order to define the parameters of interest. Other times all the parameters are of interest, but each one becomes free to vary when one or more others are being profiled.

Suppose that the data are $X_i \in \mathbb{R}^d$, for $i = 1, \dots, n$. Suppose that the estimating equation of interest is $E(m(X, \theta, \nu)) = 0$. The nonsmooth parameter η from Chapter 12.1 has been absorbed into the definition of m . The parameter θ

from there has been split into two pieces. The parameters fixed for the inference have remained in $\theta \in \mathbb{R}^p$, while the freely varying parameters are now taken to be components of $\nu \in \mathbb{R}^q$. Assume that $m(X, \theta, \nu) \in \mathbb{R}^{p+q}$.

We wish to compute

$$\mathcal{R}(\theta) = \max_{\nu} \mathcal{R}(\theta, \nu).$$

where

$$\mathcal{R}(\theta, \nu) = \max \left\{ \prod_{i=1}^n n w_i \mid \sum_{i=1}^n w_i m(X_i, \theta, \nu) = 0, w_i \geq 0, \sum_{i=1}^n w_i = 1 \right\}.$$

The value of $\mathcal{R}(\theta, \nu)$ can be found by solving the numerical problem for simple hypotheses, by iterated least squares as described in Chapter 3.14. The convex dual representation from that chapter allows us to write

$$\log \mathcal{R}(\theta) = \max_{\nu} \min_{\lambda} \mathbb{L}_{\star}(\lambda, \theta, \nu), \quad (12.1)$$

where

$$\mathbb{L}_{\star}(\lambda, \theta, \nu) = - \sum_{i=1}^n \log_{\star}(1 + \lambda' m(X_i, \theta, \nu)), \quad (12.2)$$

and

$$\log_{\star}(z) = \begin{cases} \log(z), & \text{if } z \geq \varepsilon \\ \log(\varepsilon) - 1.5 + 2z/\varepsilon - z^2/(2\varepsilon^2), & \text{if } z \leq \varepsilon, \end{cases} \quad (12.3)$$

for some small $\varepsilon > 0$.

The point of replacing \log by \log_{\star} is that it frees us from imposing the constraints $w_i \geq 0$. The function \mathbb{L} is twice continuously differentiable in λ , and for each ν , it is convex in λ . If we know that $\max_i n w_i \leq A$ at the solution, then we can choose $\varepsilon = A^{-1}$ and get the same solution using \log_{\star} without constraints on λ as we would get using \log with constraints $w_i = (1/n)(1 + \lambda' m(X_i, \theta, \nu))^{-1} \geq 0$. In Chapter 3.14 we used $\varepsilon = 1/n$, because $w_i \leq 1$ at the solution. Larger values of ε might improve the condition of the Hessian with respect to λ of \mathbb{L}_{\star} , without affecting the asymptotic validity of the empirical likelihood inferences. See the discussion around equation (3.39).

A nested algorithm uses a literal interpretation of the maximin formulation (12.1). The inner stage is to minimize $\mathbb{L}_{\star}(\lambda, \theta, \nu)$ over λ for fixed values of θ and ν . Let $\lambda(\theta, \nu)$ be the minimizing value of λ . The outer stage is to maximize

$$\mathbb{M}_{\star}(\nu, \theta) \equiv \mathbb{L}_{\star}(\lambda(\theta, \nu), \theta, \nu)$$

over ν . The parameter θ remains fixed.

At the inner stage of the nested algorithm we may required the first and second partial derivatives:

$$\mathbb{L}_{\star}^{\lambda} \equiv \frac{\partial}{\partial \lambda} \mathbb{L}_{\star} \quad \text{and} \quad \mathbb{L}_{\star}^{\lambda\lambda} \equiv \frac{\partial^2}{\partial \lambda \partial \lambda'} \mathbb{L}_{\star}.$$

At the outer stage, we may need

$$\begin{aligned}\mathbb{M}_*^\nu(\nu, \theta) &= \frac{\partial}{\partial \nu} \mathbb{L}_*(\lambda(\theta, \nu), \theta, \nu), \quad \text{and} \\ \mathbb{M}_*^{\nu\nu}(\nu, \theta) &= \frac{\partial^2}{\partial \nu \partial \nu'} \mathbb{L}_*(\lambda(\theta, \nu), \theta, \nu).\end{aligned}$$

Expressions for all of these partial derivatives are given in Chapter 12.4.

Ideally $\mathbb{M}_*^{\nu\nu}$ should be negative definite, or at least negative semidefinite. This indeed holds for large enough n , well-posed estimating equations, and values of ν and θ that are not extreme. But \mathbb{M}_* can fail to have a negative definite Hessian in practice. The outer optimization in the nested algorithm is more difficult than the inner one because of this feature.

Sophisticated optimization code takes account of the possibility that the Hessian might not be definite. For minimization problems, such as minimizing $-\mathbb{M}_*$, the target function should have positive curvature along all linear paths. Directions of negative curvature tend to be promising because they may lead further downhill. For empirical likelihood problems we do not have to worry about the objective function $-\mathbb{M}_*$ being unbounded below, so the situation is simpler than for some other optimization problems.

Figure 12.1 illustrates this problem. The example is taken from Huber's robust location and scale estimator as applied to the passage time of light data in Chapter 3.12. Suppose that θ is the scale parameter and ν is the location parameter. As θ is reduced from the MLE it becomes harder to find the optimizing ν . Imagine a hiker starting at the MLE in Figure 12.1 on a hill with contours equal to the empirical likelihood function. In each step in the profiling of θ , the hiker is instructed to step one meter south, and then move to the highest point possible on the present line of latitude. At first there is a local maximum in ν at each θ . The hiker can follow a ridge. Eventually, however, the hiker encounters a θ for which there is a local minimum of ν between two local maxima corresponding to two ridges.

The profile trace in Figure 12.1 was found by NPSOL. For small values of the scale parameter θ there are two ridges in ν . NPSOL found the higher ridge, though perhaps it overstepped before finding it. The ideal path may be one where the profile trace $\nu(\theta)$ has a discontinuity in it. For very small values of θ there are a great many ridges, but these correspond to exceedingly small \mathcal{R} values that are not in any reasonable confidence set.

The expression (12.5) for $\mathbb{M}_*^{\nu\nu}$ is a sum of two terms, one negative definite, and one not. For problems of maximizing a Gaussian log likelihood for nonlinear least squares, a similar phenomenon occurs. The Gauss-Newton algorithm for nonnegative least squares simply uses the definite term and ignores the other one. A similar algorithm may improve the outer loop in the nested algorithm, but there will still be a difficulty if the gradient \mathbb{M}_*^ν is zero at a non-optimal value of ν . For this reason, a sophisticated algorithm like NPSOL is recommended for the outer optimization.

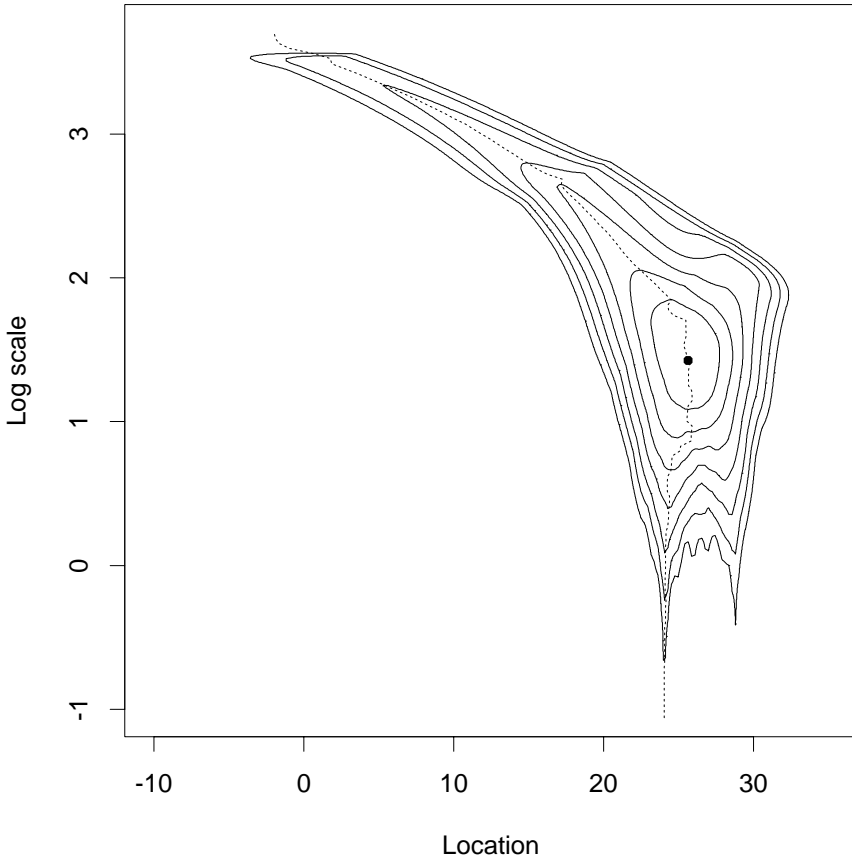


Figure 12.1 The contours are for the empirical likelihood of μ and $\log \sigma$ for Huber's robust location and scale estimates applied to the passage time of light data. This example was discussed in Chapter 3.12. Contour levels correspond to $\log \mathcal{R}(\mu, \sigma)$ from -1 to -7 by steps of -1 . The solid point is at $(\hat{\mu}, \log \hat{\sigma})$. The dotted path contains points of the form $(\mu(\sigma), \log \sigma)$ where $\mu(\sigma_0)$ maximizes empirical likelihood subject to $\sigma = \sigma_0$.

A non-nested approach is to solve the equations

$$\begin{pmatrix} \mathbb{L}_{\star}^{\lambda} \\ \mathbb{L}_{\star}^{\nu} \end{pmatrix} = 0 \in \mathbb{R}^{p+q}$$

jointly for λ and ν . Here \mathbb{L}_{\star}^{ν} is the partial derivative of \mathbb{L}_{\star} with respect to ν , given in Chapter 12.4.

The Newton-Lagrange algorithm is to apply Newton's method for solving these equations. This algorithm will converge quadratically, if started close enough to

the solution. The advantage in this approach is that it can be faster than a nested method. A nested method may expend a lot of effort optimizing λ for values of ν that are not nearly optimal. When quadratic convergence holds, it is typical to require 4 or 5 iterations of a non-nested algorithm. By comparison, a nested algorithm would require 4 or 5 outer iterations and 16 to 25 inner iterations. The speed advantage is smaller than this argument would suggest because the estimating equations $m(X_i, \theta, \nu)$ and their derivatives do not need to be recomputed at each inner iteration.

The Hessian matrix for this Newton-Lagrange approach is

$$\begin{pmatrix} \mathbb{L}_*^{\lambda\lambda} & \mathbb{L}_*^{\lambda\nu} \\ \mathbb{L}_*^{\nu\lambda} & \mathbb{L}_*^{\nu\nu} \end{pmatrix},$$

once again using partial derivatives from Chapter 12.4.

Step reduction for the Newton-Lagrange method is quite difficult. It is harder to measure progress towards a constrained optimum than towards an unconstrained one. A step that improves the objective function might also increase the violation of one or more constraints. A hybrid algorithm can give good results. The hybrid starts with Newton-like methods to solve the equations above, and then if the problem appears ill conditioned, it switches over to the nested algorithm.

12.4 Partial derivatives

Here we present the partial derivatives required by the optimization algorithms of Chapter 12.3 with $\mathbb{L}_*(\lambda, \theta, \nu)$ defined at (12.2). Some optimization software does not require second derivatives. The second derivatives have more complicated expressions than the first derivatives. They are presented here because analytic second derivatives can lead to more reliable optimization.

The partial derivatives of $\mathbb{L}_*(\lambda, \theta, \nu)$ require derivatives of the function $\log_*(z)$. The first two derivatives of $\log_*(z)$ are

$$\log_*^{(1)}(z) = \frac{d}{dz} \log_*(z) = \begin{cases} 1/z, & \text{if } z \geq \varepsilon \\ 2/\varepsilon - z/\varepsilon^2, & \text{if } z \leq \varepsilon, \end{cases}$$

and

$$\log_*^{(2)}(z) = \frac{d^2}{dz^2} \log_*(z) = \begin{cases} -1/z^2, & \text{if } z \geq \varepsilon \\ -1/\varepsilon^2, & \text{if } z \leq \varepsilon. \end{cases}$$

For $w_i \leq \varepsilon^{-1}/n$, we have $z_i = 1 + \lambda' m(X_i, \theta, \nu) \geq \varepsilon/n$, and

$$\begin{aligned} \log_*^{(1)}(z_i) &= nw_i \\ \log_*^{(2)}(z_i) &= -(nw_i)^2. \end{aligned}$$

To make some expressions shorter and more intuitive we define $\hat{w}_i = \hat{w}_i(\theta, \nu)$

and $\tilde{w}_i = \tilde{w}_i(\theta, \nu)$ through

$$\begin{aligned}\log_{\star}^{(1)}(1 + \lambda' m(X_i, \theta, \nu)) &= n\hat{w}_i \\ \log_{\star}^{(2)}(1 + \lambda' m(X_i, \theta, \nu)) &= -(n\tilde{w}_i)^2.\end{aligned}\tag{12.4}$$

Ordinarily $w_i \leq \varepsilon^{-1}/n$, and then $\hat{w}_i = \tilde{w}_i = w_i$.

The partial derivatives of \mathbb{L}_{\star} that we need are:

$$\begin{aligned}\mathbb{L}_{\star}^{\lambda} &= \frac{\partial}{\partial \lambda} \mathbb{L}_{\star}(\lambda, \theta, \nu), & \mathbb{L}_{\star}^{\lambda\lambda} &= \frac{\partial^2}{\partial \lambda \partial \lambda'} \mathbb{L}_{\star}(\lambda, \theta, \nu), \\ \mathbb{L}_{\star}^{\nu} &= \frac{\partial}{\partial \nu} \mathbb{L}_{\star}(\lambda, \theta, \nu), & \mathbb{L}_{\star}^{\nu\nu} &= \frac{\partial^2}{\partial \nu \partial \nu'} \mathbb{L}_{\star}(\lambda, \theta, \nu),\end{aligned}$$

and

$$\mathbb{L}_{\star}^{\nu\lambda} = \frac{\partial^2}{\partial \lambda \partial \nu'} \mathbb{L}_{\star}(\lambda, \theta, \nu).$$

These are of dimension $p, p \times p, q, q \times q$, and $q \times p$, respectively. Of course, $\mathbb{L}_{\star}^{\lambda\nu}$ is the transpose of $\mathbb{L}_{\star}^{\nu\lambda}$. By straightforward calculus,

$$\begin{aligned}\mathbb{L}_{\star}^{\lambda} &= - \sum_{i=1}^n \log_{\star}^{(1)}(1 + \lambda' m(X_i, \theta, \nu)) m(X_i, \theta, \nu) \\ &= - \sum_{i=1}^n n\hat{w}_i m(X_i, \theta, \nu)\end{aligned}$$

and

$$\begin{aligned}\mathbb{L}_{\star}^{\lambda\lambda} &= - \sum_{i=1}^n \log_{\star}^{(2)}(1 + \lambda' m(X_i, \theta, \nu)) m(X_i, \theta, \nu) m(X_i, \theta, \nu)' \\ &= \sum_{i=1}^n (n\tilde{w}_i)^2 m(X_i, \theta, \nu) m(X_i, \theta, \nu)'\end{aligned}$$

This Hessian is positive semidefinite and grows proportionally to n , assuming that \tilde{w}_i is roughly $1/n$. It is typical for the curvature in a parametric log likelihood ratio to grow proportionally to n . Here λ is a Lagrange multiplier and does not correspond to a tangible parameter of the data distribution. It is however the parameter in the dual likelihood. The fact that the curvature grows proportionally to n is a consequence of the scaling chosen when the problem was set up in equation (3.32).

For derivatives with respect to ν we introduce the $(p + q) \times q$ matrix

$$m_{\nu}(X_i, \theta, \nu) = \frac{\partial}{\partial \nu} m(X_i, \theta, \nu)$$

of partial derivatives of the $p + q$ estimating equations with respect to the q free

parameters. Now

$$\begin{aligned}\mathbb{L}_*^\nu &= -\sum_{i=1}^n \log_*^{(1)}(1 + \lambda' m(X_i, \theta, \nu)) m_\nu(X_i, \theta, \nu)' \lambda \\ &= -\sum_{i=1}^n n \widehat{w}_i m_\nu(X_i, \theta, \nu)' \lambda,\end{aligned}$$

and

$$\begin{aligned}\mathbb{L}_*^{\nu\lambda} &= -\sum_{i=1}^n \log_*^{(1)}(1 + \lambda' m(X_i, \theta, \nu)) m_\nu(X_i, \theta, \nu) \\ &\quad - \sum_{i=1}^n \log_*^{(2)}(1 + \lambda' m(X_i, \theta, \nu)) m_\nu(X_i, \theta, \nu)' \lambda m(X_i, \theta, \nu)' \\ &= -\sum_{i=1}^n n \widehat{w}_i m_\nu(X_i, \theta, \nu) \\ &\quad + \sum_{i=1}^n (n \widetilde{w}_i)^2 m_\nu(X_i, \theta, \nu)' \lambda m(X_i, \theta, \nu)'.\end{aligned}$$

Differentiating twice with respect to ν requires second derivatives of m with respect to ν . There is a $(p+q) \times q \times q$ array of such derivatives. To keep within the usual matrix notation, we introduce an index $j = 1, \dots, p+q$ for the estimating equations. That is,

$$\lambda' m(X_i, \theta, \nu) = \sum_{j=1}^{p+q} \lambda_j m^j(X_i, \theta, \nu).$$

Introduce

$$m_\nu^j \equiv \frac{\partial}{\partial \nu} m^j(X_i, \theta, \nu), \quad m_{\nu\nu}^j \equiv \frac{\partial^2}{\partial \nu \partial \nu'} m^j(X_i, \theta, \nu),$$

of dimensions q and $q \times q$, respectively. Now

$$\begin{aligned}\mathbb{L}_*^{\nu\nu} &= -\sum_{i=1}^n \log_*^{(1)}(1 + \lambda' m(X_i, \theta, \nu)) \sum_{j=1}^{p+q} m_{\nu\nu}^j(X_i, \theta, \nu) \\ &\quad - \sum_{i=1}^n \log_*^{(2)}(1 + \lambda' m(X_i, \theta, \nu)) \left(\sum_{j=1}^{p+q} \lambda_j m_\nu^j \right) \left(\sum_{j=1}^{p+q} \lambda_j m_\nu^j \right)' \\ &= -\sum_{i=1}^n n \widehat{w}_i \sum_{j=1}^{p+q} \lambda_j m_{\nu\nu}^j(X_i, \theta, \nu) \\ &\quad + \sum_{i=1}^n (n \widetilde{w}_i)^2 m_\nu' \lambda \lambda' m_\nu.\end{aligned}$$

Next we develop expressions for the derivatives of \mathbb{M}_* . We need

$$\frac{\partial \lambda}{\partial \nu} = \frac{\partial}{\partial \nu} \lambda(\theta, \nu),$$

a $(p + q) \times q$ matrix of partial derivatives. The chain rule gives

$$\mathbb{M}_*^\nu(\nu, \theta) = \left(\frac{\partial \lambda}{\partial \nu} \right)' \mathbb{L}_*^\lambda + \mathbb{L}_*^\nu = \mathbb{L}_*^\nu,$$

because $\mathbb{L}_*^\lambda = 0$ at the minimizer $\lambda(\theta, \nu)$. Similarly,

$$\mathbb{M}_*^{\nu\nu}(\nu, \theta) = \left(\frac{\partial \lambda}{\partial \nu} \right)' \mathbb{L}_*^{\lambda\lambda} \left(\frac{\partial \lambda}{\partial \nu} \right) + 2 \left(\frac{\partial \lambda}{\partial \nu} \right)' \mathbb{L}_*^{\lambda\nu} + \mathbb{L}_*^{\nu\nu}.$$

From a local quadratic approximation to \mathbb{L}_* we may find that

$$\frac{\partial \lambda}{\partial \nu} = - (\mathbb{L}_*^{\lambda\lambda})^{-1} \mathbb{L}_*^{\lambda\nu},$$

so that

$$\mathbb{M}_*^{\nu\nu}(\nu, \theta) = \mathbb{L}_*^{\nu\nu} - \mathbb{L}_*^{\nu\lambda} (\mathbb{L}_*^{\lambda\lambda})^{-1} \mathbb{L}_*^{\lambda\nu}. \quad (12.5)$$

Optimization over ν is easier if $\mathbb{M}_*^{\nu\nu}$ is negative definite. The second term in (12.5) is negative semidefinite, but the first term $\mathbb{L}_*^{\nu\nu}$ might not be. For $\lambda = O_p(n^{-1/2})$ and $n\tilde{w}_i$ and $n\tilde{w}_i$ both near 1, we expect $\mathbb{L}_*^{\nu\nu} = O_p(n^{-1/2})$, and $\mathbb{L}_*^{\nu\lambda} = \mathbb{L}_*^{\lambda\lambda} = O(n)$. Then $\mathbb{M}_*^{\nu\nu}$ is $O(n)$ and dominated by a negative semidefinite term.

12.5 Primal problem

The primal problem is to maximize $\sum_{i=1}^n \log(nw_i)$, subject to the constraints $w_i \geq 0$, $\sum_{i=1}^n w_i = 1$, and $\sum_{i=1}^n w_i m(X_i, \theta, \nu) = 0$, without first encoding the n dimensional vector of weights through a Lagrange multiplier. The free variables in the primal problem are $u_i = nw_i$, $i = 1, \dots, n$, and $\nu \in \mathbb{R}^q$. We work with $u_i = nw_i$ because they are of order 1 as n increases. As with the dual problem, we can replace $\log(u_i)$ by $\log_*(u_i)$ instead of or in addition to imposing the constraints $u_i \geq 0$. The Lagrangian is

$$\mathbb{P}_*(u, \theta, \nu, \lambda, \gamma) = \sum_{i=1}^n \log_*(u_i) + \lambda' \sum_{i=1}^n u_i m(X_i, \theta, \nu) + \gamma \left(\sum_{i=1}^n u_i - n \right),$$

for $u \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^r$, with $r = p + q$, $\theta \in \mathbb{R}^p$ fixed, $\nu \in \mathbb{R}^q$, and $\gamma \in \mathbb{R}$. Now instead of expressing u_i in terms of λ , we work directly with $n + r + q + 1$ free variables in u , λ , ν , and γ .

The gradient and Hessian may be written in obvious notation as

$$\begin{pmatrix} \mathbb{P}_*^u \\ \mathbb{P}_*^\lambda \\ \mathbb{P}_*^\nu \\ \mathbb{P}_*^\gamma \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbb{P}_*^{uu} & \mathbb{P}_*^{u\lambda} & \mathbb{P}_*^{u\nu} & \mathbb{P}_*^{u\gamma} \\ \mathbb{P}_*^{\lambda u} & \mathbb{P}_*^{\lambda\lambda} & \mathbb{P}_*^{\lambda\nu} & \mathbb{P}_*^{\lambda\gamma} \\ \mathbb{P}_*^{\nu u} & \mathbb{P}_*^{\nu\lambda} & \mathbb{P}_*^{\nu\nu} & \mathbb{P}_*^{\nu\gamma} \\ \mathbb{P}_*^{\gamma u} & \mathbb{P}_*^{\gamma\lambda} & \mathbb{P}_*^{\gamma\nu} & \mathbb{P}_*^{\gamma\gamma} \end{pmatrix}$$

of dimensions

$$\begin{pmatrix} n \times 1 \\ r \times 1 \\ q \times 1 \\ 1 \times 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} n \times n & n \times r & n \times q & n \times 1 \\ r \times n & r \times r & r \times q & r \times 1 \\ q \times n & q \times r & q \times q & q \times 1 \\ 1 \times n & 1 \times r & 1 \times q & 1 \times 1 \end{pmatrix}.$$

In most problems n is much larger than $r + q + 1$, and so the components \mathbb{P}_*^{uu} comprise most of the Hessian.

The derivatives with respect to u_i are

$$\begin{aligned} \frac{\partial}{\partial u_i} \mathbb{P}_* &= \log_*^{(1)}(u_i) + \lambda' m(X_i, \theta, \nu) + \gamma \\ &= -\hat{u}_i + \lambda' m(X_i, \theta, \nu) + \gamma, \end{aligned}$$

where $\hat{u}_i = n\hat{w}_i$, defined at (12.4), and

$$\frac{\partial^2}{\partial u_i \partial u_j} \mathbb{P}_* = \log_*^{(2)}(u_i) 1_{i=j} = -\tilde{u}_i^2 1_{i=j},$$

where $\tilde{u}_i = n\tilde{w}_i$, also defined at (12.4). Notice particularly that the n by n Hessian matrix \mathbb{P}_*^{uu} is diagonal and negative definite. Therefore, the Hessian of \mathbb{P}_* is very sparse when $n \gg q + r + 1$.

The derivatives with respect to λ are

$$\mathbb{P}_*^\lambda = \sum_{i=1}^n u_i m(X_i, \theta, \nu) \quad \text{and} \quad \mathbb{P}_*^{\lambda\lambda} = 0.$$

The derivatives with respect to ν are

$$\mathbb{P}_*^\nu = \left(\sum_{i=1}^n u_i m_\nu(X_i, \theta, \nu) \right) \lambda$$

and

$$\mathbb{P}_*^{\nu\nu} = \sum_{i=1}^n u_i \sum_{j=1}^{p+q} \lambda_j m_{\nu\nu}^j(X_i, \theta, \nu).$$

The derivatives with respect to γ are

$$\mathbb{P}_*^\gamma = \sum_{i=1}^n u_i - n \quad \text{and} \quad \mathbb{P}_*^{\gamma\gamma} = 0.$$

The cross partial derivatives $\mathbb{P}_*^{\lambda\gamma}$, and $\mathbb{P}_*^{\nu\gamma}$ are zero. Also

$$\mathbb{P}_*^{u\gamma} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \quad \mathbb{P}_*^{u\lambda} = \begin{pmatrix} m(X_1, \theta, \nu)' \\ m(X_2, \theta, \nu)' \\ \vdots \\ m(X_n, \theta, \nu)' \end{pmatrix}, \quad \mathbb{P}_*^{u\nu} = \begin{pmatrix} \lambda' m_\nu(X_1, \theta, \nu) \\ \lambda' m_\nu(X_2, \theta, \nu) \\ \vdots \\ \lambda' m_\nu(X_n, \theta, \nu) \end{pmatrix},$$

and finally,

$$\mathbb{P}_*^{\lambda\nu} = \sum_{i=1}^n u_i m_\nu(X_i, \theta, \nu).$$

The disadvantage of the primal formulation is that it requires a much larger Hessian matrix. Direct approaches based on computing and solving this matrix require $O(n^3)$ time per iteration and $O(n^2)$ storage. Methods that exploit sparsity can remove both issues at the expense of some additional algorithmic complexity.

The primal formulation has the advantage of simpler expressions for required partial derivatives. It may also be possible for a primal problem to find a path through the space of u_i variables that goes around some difficult point in the space of ν and λ values.

The Hessian in the primal formulation is sparse because the upper left n by n submatrix in it is diagonal. Partition the Hessian of \mathbb{P}_* as

$$\begin{pmatrix} \mathbb{P}_*^{uu} & \mathbb{P}_*^{uv} \\ \mathbb{P}_*^{vu} & \mathbb{P}_*^{vv} \end{pmatrix},$$

where v denotes the variables other than u , namely λ , ν , and γ . Similarly, let \mathbb{P}_*^v contain the gradient of \mathbb{P}_* with respect to all variables other than u . Then the Newton step solves

$$\begin{pmatrix} \mathbb{P}_*^{uu} & \mathbb{P}_*^{uv} \\ \mathbb{P}_*^{vu} & \mathbb{P}_*^{vv} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} \mathbb{P}_*^u \\ \mathbb{P}_*^v \end{pmatrix}$$

for vectors u and v . Because \mathbb{P}_*^{uu} is diagonal, it is advantageous to rewrite the first row as

$$\mathbb{P}_*^{uu} u = - (\mathbb{P}_*^u + \mathbb{P}_*^{uv} v).$$

Thus, once v is found, it is easy to find u . To find v , write out the second row of equations, and substitute for u , obtaining

$$\left(\mathbb{P}_*^{vv} - \mathbb{P}_*^{vu} (\mathbb{P}_*^{uu})^{-1} \mathbb{P}_*^{uv} \right) v = - \left(\mathbb{P}_*^v - \mathbb{P}_*^{vu} (\mathbb{P}_*^{uu})^{-1} \mathbb{P}_*^u \right).$$

The equations to solve for v are not usually sparse, but there are only $q + r + 1 = p + 2q + 1$ of them. This approach to factoring sparse Hessians is known as the range space technique. Of course, step reduction strategies are required, as is a merit function trading off likelihood optimization and constraint satisfaction.

12.6 Sequential linearization

Since the optimization problem for a mean is so simple, a strategy based on approximating the statistic of interest by a mean can be effective. This approach is available for statistics not conveniently expressed in terms of estimating equations, such as U -statistics. Let us suppose that a statistic θ is defined through a function $\theta = T(F)$, for the distribution F putting weight w_i on observation X_i . We will suppose at first that any process for maximizing over nuisance parameters is embedded into $T(F)$. We will use the same symbol T to denote the function of w_1, \dots, w_n defined by

$$T(w_1, \dots, w_n) \equiv T\left(\sum_{i=1}^n w_i \delta_{X_i}\right).$$

The NPMLLE is $\hat{\theta} = T(\hat{F})$. Suppose that T is smooth, so that we may write

$$T(w_1, \dots, w_n) \doteq \hat{\theta} + \sum_{i=1}^n w_i T_i(\hat{F}),$$

where

$$T_i(F) = \lim_{\varepsilon \rightarrow 0} \frac{T((1-\varepsilon)F + \varepsilon \delta_{X_i}) - T(F)}{\varepsilon} \quad (12.6)$$

measures the effect of small increases in the weight w_i .

With linearization, we write

$$\mathcal{R}_L(\theta) = \max \left\{ \prod_{i=1}^n n w_i \mid \sum_{i=1}^n w_i (\hat{\theta} + T_i(\hat{F}) - \theta) = 0, w_i \geq 0, \sum_{i=1}^n w_i = 1 \right\}.$$

This linearized profile empirical likelihood ratio function can be computed using the algorithm for the vector mean described in Chapter 3.14. Empirical likelihood inferences based on profiling $\mathcal{R}_L(\theta)$ use the same $T_i(\hat{F})$ quantities as are used in the infinitesimal jackknife (IJ) described in Chapter 9.9. The difference is that the IJ simply uses them to construct a variance estimate. Turning the IJ variance estimate into a confidence ellipsoid amounts to using the Euclidean likelihood for the mean of $T_i(\hat{F})$.

Having computed the optimizing w_i we can evaluate $T(w_1, \dots, w_n)$. In general $T(w_1, \dots, w_n) \neq \theta$, because the linearization was only an approximation. We can however re-linearize around these values w_i and repeat the process. Suppose that $w_i^{(k)}$ are the weights after k linearizations have been done, starting with $w_i^{(0)} = 1/n$. The weights $w_i^{(k+1)}$ are found by maximizing $\prod_{i=1}^n n w_i^{(k+1)}$ subject

to

$$\begin{aligned}
 0 &= \sum_{i=1}^n w_i^{(k+1)} \left(\hat{\theta}_k + T_i(\hat{F}^{(k)}) - \theta \right), \\
 0 &\leq w_i^{(k+1)}, \quad \text{and} \\
 1 &= \sum_{i=1}^n w_i^{(k+1)},
 \end{aligned}$$

where $\hat{F}^{(k)}$ uses weights $w_i^{(k)}$, and $\hat{\theta}_k = T(\hat{F}^{(k)})$. The resulting maximum is denoted by $\mathcal{R}_{L,k+1}(\theta)$.

It is tempting to use $\mathcal{R}_{L,\infty}(\theta) = \lim_{k \rightarrow \infty} \mathcal{R}_{L,k}(\theta)$, but very often the sequence does not converge, especially for values of θ far from $\hat{\theta}$. Fortunately, good results can be obtained from a fixed number k of iterations. Let $\varphi \in \mathbb{R}^p$ be a fixed vector and define $\varphi_n = n^{-1/2}\varphi$. Then, under smoothness conditions, including existence of $k + 2$ continuous derivatives for $T(w_1, \dots, w_n)$,

$$-2 \log \mathcal{R}_{L,k}(\theta_0 + \varphi_n) = -2 \log \mathcal{R}(\theta_0 + \varphi_n)(1 + O_p(n^{-k/2})), \quad (12.7)$$

where θ_0 is the true value of θ . The two log likelihood ratios agree within a relative error of $O_p(n^{-k/2})$, over the region where the likelihood is not negligible. For $k \geq 1$, there is an asymptotic χ^2 calibration for $-2 \log \mathcal{R}_{L,k}(\theta_0)$. When $k \geq 2$, the result is qualitatively different from the simple linearization used in the infinitesimal jackknife. When the empirical likelihood is Bartlett correctable, then so is $\mathcal{R}_{L,k}$ for $k \geq 4$. See Chapter 12.7.

Now consider the problem of finding the derivatives $T_i(F)$. When F has an explicit representation in terms of w_i , it may be possible to compute $T_i(F)$ by analytic differentiation. Otherwise, divided differences of the form (12.6) for small $\varepsilon > 0$ may be used. This requires $n + 1$ function evaluations per iteration, or fewer if there are ties among the X_i . A central divided difference approximation

$$T_i(F) \doteq \frac{T((1 + \varepsilon_i)F - \varepsilon_i \delta_{X_i}) - T((1 - \varepsilon_i)F + \varepsilon_i \delta_{X_i})}{2\varepsilon_i}, \quad (12.8)$$

should provide more accurate T_i at the expense of nearly doubling the cost of differentiation. If T is not defined for negative observation weights, then we require $\varepsilon_i < (1 - F(\{X_i\}))^{-1}F(\{X_i\})$ in order to use (12.8).

For some complicated statistics, $T(F)$ is only available for weights $w_i = n_i/N$, where n_i are integers and $N = \sum_{i=1}^n n_i$. This corresponds to making an artificial data set in which observation X_i appears n_i times. At least for $k = 0$ it is straightforward to find $T_i(F^{(k)})$. One can define T_i using equation (12.6) with $\varepsilon = -1/n$, or with $\varepsilon = 1/(n + 1)$, by deleting observation X_i , or by including it twice, respectively. For $k \geq 1$, the distribution $\hat{F}^{(k)}$ may not correspond to one with an integer n_i copies of X_i , and a large value of N may be required for a good approximation.

Now consider applying linearization to a statistic $\theta \in \mathbb{R}^p$, defined through

estimating equations involving a nuisance parameter $\nu \in \mathbb{R}^q$. Let $\theta(w_1, \dots, w_n)$ and $\nu(w_1, \dots, w_n)$ be jointly defined by

$$\sum_{i=1}^n w_i m(X_i, \theta, \nu) = 0,$$

where $m(X, \theta, \nu)$ is continuously differentiable in (θ, ν) . The linearization of θ about $w^k = (w_1^k, \dots, w_n^k)$ is written

$$\theta(w) \doteq \theta(w^k) + \sum_{i=1}^n w_i T_i(w^k),$$

with (12.6) expressed in weight notation as

$$T_j(w^k) = \lim_{\varepsilon \rightarrow 0} \frac{T((1 - \varepsilon)w^k + \varepsilon \delta_{X_j}) - T(w^k)}{\varepsilon}.$$

Let $w_i^{k,\varepsilon} = (1 - \varepsilon)w_i^k + \varepsilon 1_{i=j}$, and let θ^ε and ν^ε be defined through these weights. Let m_θ and m_ν be the $(p + q) \times p$ and $(p + q) \times q$ matrices of partial derivatives of m with respect to θ and ν . Then, ignoring terms of order ε^2 and higher,

$$\begin{aligned} 0 &= \sum_{i=1}^n w_i^{k,\varepsilon} m(X_i, \theta^\varepsilon, \nu^\varepsilon) \\ &\doteq \sum_{i=1}^n w_i^{k,\varepsilon} \left[m(X_i, \theta, \nu) + \begin{pmatrix} m_\theta(X_i, \theta, \nu) & m_\nu(X_i, \theta, \nu) \end{pmatrix} \begin{pmatrix} \theta^\varepsilon - \theta \\ \nu^\varepsilon - \nu \end{pmatrix} \right] \\ &\doteq \varepsilon m(X_j, \theta, \nu) + \left[\sum_{i=1}^n w_i^k \begin{pmatrix} m_\theta(X_i, \theta, \nu) & m_\nu(X_i, \theta, \nu) \end{pmatrix} \right] \begin{pmatrix} \theta^\varepsilon - \theta \\ \nu^\varepsilon - \nu \end{pmatrix}, \end{aligned}$$

from which

$$\begin{pmatrix} T_j(w^k) \\ N_j(w^k) \end{pmatrix} = -J(w^k)^{-1} m(X_j, \theta, \nu), \quad (12.9)$$

where

$$J(w^k) = \sum_{i=1}^n w_i^k \begin{pmatrix} m_\theta(X_i, \theta, \nu) & m_\nu(X_i, \theta, \nu) \end{pmatrix} \quad (12.10)$$

and $N_j(w^k)$ is the linearization term for ν , which we need not compute. Notice that the matrix J in (12.10) does not depend on j . Thus it only needs to be factored once for each set of w_i^k . The matrix is invertible under the assumption that the estimating equations define θ and ν .

For parameters defined through estimating equations with nuisance parameters, sequential linearization can be carried out for some number k of iterations as described above. The linearization (12.9) is recomputed at each iteration.

12.7 Bibliographic notes

Standard references on numerical optimization include Gill et al. (1981) and Fletcher (1987). Gill et al. (1981) include a very comprehensive discussion of constrained optimization. NPSOL is described in Gill, Murray, Saunders & Wright (1999). Coleman (1984) describes large sparse optimization problems, including range space methods.

Owen (1990*b*) describes how to compute empirical likelihood for a vector mean, corresponding to the inner loop of the nested algorithm. See Chapter 3.14. The nested algorithm was presented in Owen (1990*a*). and Owen (1990*b*) Newton's method was proposed by Hall & La Scala (1990). The primal problem was solved using NPSOL by Owen (1991), for regression problems, and by Kolaczyk (1994), for generalized linear models. Computation of profile empirical likelihood ratios for constraints was discussed in Owen (1992). Owen (1990*b*) reports that iterated linearizations can fail to converge. Wood et al. (1996) consider linearization inferences and were the first to point out the value of stopping after a fixed number of iterations. Equation (12.7) is based on a theorem in Wood et al. (1996).

Semi-infinite programming is a technique for handling optimizations with an infinite number of constraints, only a finite number of which are binding. Lesperance & Kalbfleisch (1992), use semi-infinite programming on a nonparametric likelihood ratio problem arising from mixture sampling.

Owen (1988*a*) describes an approach to profiling a pair of parameters by following a spiral path along a grid of points, starting at the MLE. Bates & Watts (1988) describe an approach to profiling a sum of squares function in a parameter $(\theta_1, \theta_2) \in \mathbb{R}^2$ using profile traces. The two profile traces intersect a desired contour at four points. The contour is also normal to the profile trace at the intersections. Thus the traces provide eight pieces of information on the desired contour, four locations and four slopes. They use a spline that agrees with these eight pieces of information.