CHAPTER 4

Iterative Methods for Parameter Estimation

A wide variety of parameter estimation techniques require the ability to minimize or maximize a complicated function of the parameters. In this chapter we look at several general methods for optimization.

Newton-Raphson for Maximum Likelihood Estimation

Let $X_1, \ldots, X_n$ be n independent random variables such that the probability density or mass function of X_i is $f_i(x;\theta)$ which depends on a vector θ of r parameters. Then the maximum likelihood estimator $\hat{\theta}$ of θ is that value that maximizes the log likelihood function

$$l(\theta) = \sum_{i=1}^{n} \log f_i(X_i;\theta),$$

or, equivalently, minimizes $L(\theta) = -l(\theta)$. Given an initial guess θ_0 for $\hat{\theta}$, Newton-Raphson can be used to find the MLE. The jth element of the gradient and the (j,k)th element of the Hessian are given by

$$\begin{aligned} g_j &= \frac{\partial L}{\partial \theta_j} = -\sum_{i=1}^{n} \frac{\partial \log f_i(X_i;\theta)}{\partial \theta_j}, \qquad j = 1, \ldots, r, \\ H_{jk} &= \frac{\partial^2 L}{\partial \theta_j \partial \theta_k} = -\sum_{i=1}^{n} \frac{\partial^2 \log f_i(X_i;\theta)}{\partial \theta_j \partial \theta_k}, \qquad j,k = 1, \ldots, r. \end{aligned} \tag{1}$$

To perform inferences about θ based on $\hat{\theta}$, we can use the fact that

$$\hat{\theta} \sim \mathrm{AN}(\theta, V(\theta)), \quad V(\theta) = I^{-1}(\theta),$$

where the (j,k)th element of the information matrix $I(\theta)$ is

$$I_{jk}(\theta) = -\mathrm{E}\left(\frac{\partial^2 \log l(\theta)}{\partial \theta_j \partial \theta_k}\right) = \mathrm{E}\left(\frac{\partial^2 \log L(\theta)}{\partial \theta_j \partial \theta_k}\right) = -\sum_{i=1}^{n} \mathrm{E}\left(\frac{\partial^2 \log f_i(X_i;\theta)}{\partial \theta_j \partial \theta_k}\right)$$

If in addition to being independent, the X's are identically distributed with common density or mass function f, then all of the expectations in this last expression will be the same and we can write

$$I_{jk}(\theta) = n i_{jk}(\theta) = -n\mathrm{E}\left(\frac{\partial^2 \log f(X;\theta)}{\partial \theta_j \partial \theta_k}\right),$$

where the matrix $i(\theta)$ is called the information in a single observation.

It is often difficult to evaluate the expectations in $i(\theta)$ so a natural procedure is to approximate them by their sample averages, which from (1) are

$$\hat{i}_{jk}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\frac{\partial^2 \log f(X_j;\theta)}{\partial\theta_j\partial\theta_k} = \frac{1}{n}H_{jk}(\theta),$$

and so a popular method of finding standard errors of $\hat{\theta}$ is to use covariance matrix $H^{-1}(\hat{\theta})$, that is, the inverse of the Hessian matrix at the last Newton-Raphson iteration.

The Method of Scoring

The method of scoring (see Rao, 1973, p. 366, for example) is a procedure that is very similar to Newton-Raphson and consists of iterations of the form

$$\theta_{i+1} = \theta_i + I^{-1}(\theta_i)g_i.$$

MLE's for the Weibull Distribution

To illustrate the process, we consider the two dimensional Weibull distribution which is an often-used distribution in survival analysis and has density, distribution, and quantile functions

$$f(x) = \begin{cases} \alpha\beta x^{\beta-1}e^{-\alpha x^\beta}, & x > 0, \alpha > 0, \beta > 0 \\ 0, & \text{otherwise,} \end{cases}$$

$$F(x) = 1 - e^{-\alpha x^\beta}, \quad x > 0,$$

$$Q(u) = \left[\frac{-\log(1-u)}{\alpha}\right]^{1/\beta}, \quad 0 \le u \le 1.$$

Note that the Weibull distribution becomes the exponential distribution when $\beta = 1$. We consider estimating the parameters α and β given a sample $X_1, \ldots, X_n$.

The MLE's $\hat{\alpha}$ and $\hat{\beta}$ are the values of α and β maximizing the log likelihood

$$l(\alpha, \beta; X) = n\log\alpha + n\log\beta + (\beta - 1)\sum_{i=1}^{n}\log X_i - \alpha\sum_{i=1}^{n}X_i^\beta.$$

It is easy to verify that the gradient and Hessian corresponding to minimizing $L(\theta) = -l(\theta)$ are given by

$$g = \begin{bmatrix} \sum_{i=1}^n X_i^\beta - \dfrac{n}{\alpha}, \\ \alpha\sum_{i=1}^n X_i^\beta \log X_i - \sum_{i=1}^n \log X_i - \dfrac{n}{\beta}, \end{bmatrix},$$

$$H = \begin{bmatrix} \dfrac{n}{\alpha^2} & \sum_{i=1}^n X_i^\beta \log X_i, \\ \text{Sym} & \alpha\sum_{i=1}^n X_i^\beta(\log X_i)^2 + \dfrac{n}{\beta^2}. \end{bmatrix}.$$

From $Q(u) = (-\log(1-u)/\alpha)^{1/\beta}$, we have $y = \beta_0 + \beta_1 x$, where $\beta_0 = \log\alpha$, $\beta_1 = \beta$, $y = \log(-\log(1-u))$, and $x = \log Q(u)$. Thus if we let

$$u_i = \frac{i - 1/2}{n}, \quad y_i = \log(-\log(1-u_i)), \quad x_i = \log\hat{Q}(u_i) = \log X_{(i)}, \quad i = 1, \ldots, n,$$

where $X_{(i)}$ is the ith order statistic of the random sample, we can regress the y_i's on the x_i's to get estimators $\hat{\beta}_0$ and $\hat{\beta}_1$, from which we get starting values for the Newton-Raphson procedure as $(e^{\hat{\beta}_0}, \hat{\beta}_1)$.

To find standard errors for $\hat{\alpha}$ and $\hat{\beta}$ we need to find

$$I = -\begin{bmatrix} \frac{1}{\alpha^2} & \mathrm{E}(X^\beta \log X) \\ \mathrm{Sym} & \frac{1}{\beta^2} + \mathrm{E}(\alpha X^\beta (\log X)^2) \end{bmatrix},$$

which is very difficult analytically, so the final Hessian would be used.

Estimating Survival Probabilities

An important quantity in survival analysis is the probability of surviving at least x time units, that is, p_x. The MLE of p_x is just $\hat{p}_x = e^{-\hat{\alpha}x^{\hat{\beta}}}$, and another standard result for MLE's gives that

$$\hat{p}_x \sim \mathrm{AN}(p_x, r^T(\theta)V(\theta)r(\theta)/n),$$

where $r(\theta) = (\partial p_x/\partial\alpha, \partial p_x/\partial\beta)^T$. Thus the asymptotic standard error of $\hat{p}_x$ is given by

$$\mathrm{ase}(p_x) = \sqrt{r^T(\theta)I^{-1}(\theta)r(\theta)/n},$$

which is estimated by substituting $\hat{\theta}$ in $r(\theta)$ and using $\hat{I}^{-1}(\hat{\theta})$ instead of $I^{-1}(\theta)$.

Newton-Raphson and Nonlinear Regression

If we have a set of n observations $y_1, \ldots, y_n$ on some dependent variable y and associated with the ith y we have a vector X_i of measurements on k independent variables, and the y's and X's satisfy the nonlinear regression model

$$y_i = f(X_i; \theta) + \epsilon_i, \qquad i = 1, \ldots, n$$

where f is a function that is nonlinear in the r elements of θ and the ϵ's are uncorrelated with mean zero and constant variance σ^2, then the least squares estimators $\hat{\theta}$ of θ minimize the sum of squares of residuals, that is,

$$S(\theta) = \sum_{i=1}^{n} (y_i - f(X_i; \theta))^2,$$

and we can again use Newton-Raphson to find $\hat{\theta}$. In this case, it is easy to derive the fact that the elements of the gradient and Hessian can be written as

$$g_j = \frac{\partial S}{\partial \theta_j} = -2\sum_{i=1}^{n} (y_i - f_i)\frac{\partial f_i}{\partial \theta_j}$$
$$H_{jk} = \frac{\partial^2 S}{\partial \theta_j \partial \theta_k} = -2\sum_{i=1}^{n} \left[(y_i - f_i)\frac{\partial^2 f_i}{\partial \theta_j \partial \theta_k} - \frac{\partial f_i}{\partial \theta_j}\frac{\partial f_i}{\partial \theta_k} \right],$$

where $f_i = f(X_i; \theta)$. If we define the $(n \times r)$ Jacobian matrix J to have (i, j)th element $\partial f_i/\partial\theta_j$, and the $(r \times r)$ matrices $G_1, \ldots, G_n$ so that the (j, k)th element of G_i is $\partial f_i^2/\partial\theta_j\partial\theta_k$, and finally the vector r of length n to have ith element $r_i = y_i - f_i$, then the gradient and Hessian are given by

$$g = -2J^T r, \qquad H = 2J^T J - 2\sum_{i=1}^{n} r_i G_i. \tag{2}$$

Often, the Hessian is approximated by the first term in this sum, which gives what is called the Gauss-Newton algorithm. In this case, we need to solve at each iteration the 'normal equations' $J_i^T J_i d_i = -J_i^T r$. In either case, inferences about θ based on $\hat{\theta}$ follow from the fact that

$$\hat{\theta} \sim \text{AN}(\theta, \sigma^2 (J^T J)^{-1}),$$

and estimating σ^2 by $s^2 = S(\hat{\theta})/(n-r)$ and J by $J(\hat{\theta})$.

If the errors in the model are normally distributed, then so are the y's and it is easy to show that the negative of the log likelihood function of θ and σ^2 is

$$L(\theta, \sigma^2) = -\frac{1}{2\sigma^2} S(\theta) - \frac{n}{2} \log 2\pi\sigma^2,$$

which means that

$$\frac{\partial L}{\sigma^2} = \frac{1}{2\sigma^4} S(\theta) - \frac{n}{2\sigma^2}, \qquad \frac{\partial L}{\partial \theta} = \frac{1}{2\sigma^2} \frac{\partial S}{\partial \theta}, \qquad \frac{\partial^2 L}{\partial \theta^2} = \frac{1}{2\sigma^2} \frac{\partial^2 S}{\partial \theta^2},$$

which in turn means that $\hat{\sigma}^2 = S(\hat{\theta})/n$ and that the Newton-Raphson iterations to minimize $S(\theta)$ are exactly the same as the Newton-Raphson iterations to maximize the log likelihood function with respect to θ, and thus the least squares estimates and MLE's are the same.

One other interesting thing to notice is that in the Gaussian error case, the method of scoring is the same as the Gauss-Newton method since

$$\left[-\text{E}\left(\frac{\partial^2 l}{\partial \theta^2}\right)\right]^{-1} \frac{\partial l}{\partial \theta} = -\left[\text{E}\left(\frac{\partial^2 S}{\partial \theta^2}\right)\right]^{-1} \frac{\partial S}{\partial \theta} = -\left[\text{E}(H)\right]^{-1} g = -\left[J^T J\right]^{-1} J^T r,$$

since from (2), we have

$$\text{E}(H) = 2J^T J - 2\sum_{i=1}^{n} \text{E}(y_i - f_i) G,$$

and this last expectation is zero.

A Nonlinear Regression Example

To illustrate the procedure above, we consider an example discussed in Draper and Smith's regression book (p. 475), where y represents the percent of chlorine in a unit of some product and the single explanatory variable X is the length of time since the product was produced. The 44 data points in the example are listed at the beginning of the Fortran program at the end of this section. The nonlinear model used to explain y in terms of X has two parameters α and β and is given by

$$y_i = \alpha + (0.49 - \alpha) e^{-\beta(X_i - 8)} + \epsilon_i, \qquad i = 1, \ldots, 44,$$

that is, $f_i = \alpha + (0.49 - \alpha) e_i$, where $e_i = e^{-\beta x_i}$, with $x_i = X_i - 8$. From this we have

$$\frac{\partial f_i}{\partial \alpha} = 1 - e_i, \quad \frac{\partial f_i}{\partial \beta} = -\alpha^* x_i e_i, \quad \frac{\partial^2 f_i}{\partial \alpha^2} = 0, \quad \frac{\partial^2 f_i}{\partial \beta^2} = \alpha^* x_i^2 e_i, \quad \frac{\partial^2 f_i}{\partial \alpha \partial \beta} = x_i e_i,$$

where $\alpha^* = 0.49 - \alpha$, which gives

$$g = \begin{bmatrix} -2\sum_{i=1}^{n} r_i (1 - e_i) \\ 2\alpha^* \sum_{i=1}^{n} r_i x_i e_i \end{bmatrix}, \qquad H = 2J^T J - 2 \begin{bmatrix} 0 & \text{symmetric} \\ \sum_{i=1}^{n} r_i x_i e_i & \alpha^* \sum_{i=1}^{n} r_i x_i^2 e_i \end{bmatrix}.$$

In the Fortran program discussed at the end of this chapter, we have written the subroutine that evaluates the Hessian so that it is easy to include the second term or not. The program calls the Newton-Raphson subroutne twice, the first time does Gauss-Newton and the second time it uses the full Hessian. The output is as follows:

```
Draper and Smith example: Gauss-Newton:

         alpha          beta        SS Res
-----------------------------------------
       .300000       .020000       .026315
       .841649       .100682      4.488134
       .390135       .100419       .005006
       .390135       .101605       .005002
       .390140       .101632       .005002
ier =     0

Draper and Smith example: Full Hessian:

         alpha          beta        SS Res
-----------------------------------------
       .300000       .020000       .026315
       .259564       .024729       .012877
ier =     2
```

Note that Gauss-Newton converges quickly, while using the full Hessian immediately results in a Hessian that is not positive definite (this is what `ier=2` indicates; see the discussion at the end of the chapter for more details). This is a common phenomenon in nonlinear regression and using Newton-Raphson in general.

Iteratively Reweighted Least Squares

We have seen that the Gauss-Newton procedure for nonlinear regression (equivalent to scoring for the Gaussian case) led to solving a set of ordinary least squares normal equations at each iteration. It turns out that there are a number of estimation procedures that lead at each iteration to solving weighted least squares normal equations. Such an iterative procedure is called iteratively reweighted least squares and it naturally arises when we have m independent random variables $y_1, \ldots, y_m$ whose likelihood depends on a set of parameters η which in turn depends functionally on a smaller set of parameters θ. In Gaussian nonlinear regression for example, the dependent variables $y_1, \ldots, y_n$ are normally distributed with means $\mu_1, \ldots, \mu_n$ which in turn depend on the vector θ by $\mu_i = f(X_i; \theta)$. Another example is logistic regression wherein we observe $y_1, \ldots, y_m$ which are assumed to be independent random variables with y_i having the binomial distribution with parameters n_i and

$$p_i = \frac{1}{1 + e^{-X_i^T \beta}}, \qquad i = 1, \ldots, m,$$

where for each i, $X_i = (X_{i1}, \ldots, X_{ip})^T$ is a vector of covariates, and the vector $\beta = (\beta_1, \ldots, \beta_p)^T$ contains the parameters of interest. For example, a negative value of β_j indicates that the jth covariate has a negative partial effect on the success probability. Thus we have a vector $p = (p_1, \ldots, p_m)^T$ of parameters which are a nonlinear function of a usually much smaller set β of parameters.

Thus in general we consider a negative log likelihood L which depends on a vector η of n parameters which are functionally related to a vector θ of p parameters. If we use the chain rule when differentiating L and if we use the expectation of the Hessian instead of the Hessian itself (that is, use the method of scoring), then the iterations one obtains are of the form

$$\theta_{i+1} = \theta_i - d_i, \qquad W_i^T A_i W_i d_i = W_i^T z_i,$$

where W is $(n \times p)$, A is $(n \times n)$, z is $(n \times 1)$, and

$$W_{jk} = \frac{\partial \eta_j}{\partial \theta_k}, \qquad A_{jk} = -\mathrm{E}\left(\frac{\partial^2 L}{\partial \eta_j \partial \eta_k}\right) = \mathrm{E}\left(\frac{\partial L}{\partial \eta_j}\frac{\partial L}{\partial \eta_k}\right), \qquad z_j = \frac{\partial L}{\partial \eta_j}.$$

Thus at each iteration, one needs to solve weighted least squares normal equations for the regression of z on AW with error covariance matrix A. This can be done via the modified Cholesky decomposition method described in Section ????.

For Gaussian nonlinear regression, the matrix W is the same as the Jacobian J, while since $\partial L/\partial f_i = -(y_i - f_i)/\sigma^2$, we have that $z = -r/\sigma^2$ and the matrix A is σ^2 times the identity matrix, and thus iteratively reweighted least squares is also identical to Gauss-Newton and the method of scoring.

For logistic regression, if we let L be the negative of the log likelihood of β, then

$$L = -\sum_{i=1}^{m} \left[\log \binom{n_i}{y_i} + y_i \log p_i + (n_i - y_i) \log(1 - p_i) \right],$$

from which we get

$$z_j = \frac{\partial L}{\partial p_j} = \frac{n_j - y_j}{1 - p_j} - \frac{y_j}{p_j}, \qquad W_{jk} = \frac{\partial p_j}{\partial \beta_k} = \frac{X_{jk} e^{-X_j^T \beta}}{1 + e^{-X_j^T \beta}} = X_{jk}(1 - p_j),$$

and

$$\frac{\partial^2 L}{\partial p_j \partial p_k} = \delta_{j-k} \left[\frac{n_j - y_j}{(1 - p_j)^2} + \frac{y_j}{p_j^2} \right],$$

where δ_v is the Kronecker delta function, that is, δ_v is one if $v = 0$ and zero otherwise. Thus we have

$$W = \text{Diag}\{1 - p_j\} X, \qquad A = -\text{E}\left(\frac{\partial^2 L}{\partial p \partial p} \right) = \text{Diag}\left\{ \frac{n_j}{p_j(1 - p_j)} \right\},$$

and thus

$$W^T A W = X^T \text{Diag}\{1 - p_j\} \text{Diag}\left\{ \frac{n_j}{p_j(1 - p_j)} \right\} \text{Diag}\{1 - p_j\} X$$

$$= X^T \text{Diag}\left\{ \frac{n_j(1 - p_j)}{p_j} \right\} X = X^T \text{Diag}\{n_j e^{-X_j^T \beta}\} X.$$

To find starting values, we note that

$$r_i = -\log\left(\frac{1 - p_i}{p_i} \right) = \sum_{j=1}^{p} \beta_j X_{ij}, \qquad i = 1, \ldots, m,$$

and thus if we let $\hat{p}_i = y_i/n_i$, we can do ordinary least squares regression of the resulting $\hat{r}$'s on the $(m \times p)$ matrix X of covariates.

A Fortran Implementation

We have listed below a Fortran implementation of the two examples we have described above. There are a number of important features of Fortran used in this program, in particular, the use of the `external` statement and named `common`. The program calls a series of subprograms that are not in the listing, including `runif` for generating U(0,1) variables, `qsort` to sort a vector, `slr` to do simple linear regression, and `solvch` to solve a system of equations via the modified Cholesky decomposition.

Passing a Subroutine Name as an Argument: The External Statement

At the end of the listing is a subroutine called `nr` that is designed to do the Newton-Raphson procedure for any function L. It calls a subroutine called `grdhss` which has input arguments `theta`, `n`, and `ndim` and output arguments `g` and `h` which are the gradient and Hessian of the function being minimized. The subroutine `grdhss` is actually a 'dummy name' that is passed to `nr` through its argument list. In the example, `nr` gets called twice, once for the Draper and Smith nonlinear regression example in which case the gradient

and Hessian subroutine is called `dsgh`, and once for the Weibull example in which case the subroutine is called `weibgh`. When one passes the name of a subroutine or function to a subroutine or function, it must be declared in an `external` statement (see the beginning of the main program in the listing).

The Common and Named Common Statements

The `nr` subroutine has no way of knowing what information other than a vector `theta` the gradient and Hessian subroutine it is calling needs to know in order to do its work. Thus this information can't be passed to `grdhss` in its argument list (you can't have varying numbers of arguments in a call to a subroutine either). Thus the information the real gradient and Hessian routine needs (as opposed to the dummy `grdhss`) gets passed in a `common` statement (see the beginning of the main program and the beginning of each of `dsgh` and `weibgh` to see how this works).

If we were only using one gradient and Hessian routine, the `common` statement would not have to be 'named', that is, the first statement would just be `common dsx(44),dsy(44),nds,iopt,ssr`. In this case we have to have two such statements with each one named.

There is one other thing to be sure to notice. We wanted to define the Draper and Smith data in a `data` statement rather than reading it in. Unfortunately, one is not allowed to have `common` variables in a data statement so we defined another pair of vectors and defined them in a `data` statement and then copied them to the vectors in the `common`.

Checking for Convergence

In the `nr` subroutine we have checked for convergence of successive iterations by seeing if all of the elements of successive iterations are within `eps` of each other in terms of 'relative absolute error,' which for numbers x and y is defined to be $|x-y|/|x|$. Since this denominator can be close to zero, we actually check whether $|x-y| < \epsilon|x|$.

```
1         subroutine nr(theta,n,ndim,maxit,eps,grdhss,g,h,al,d,wk,
2      1                del,ier)
3  c-----------------------------------------------------------------
4  c
5  c   Subroutine to perform Newton-Raphson minimization of a
6  c   function of n variables.
7  c
8  c   Input:
9  c          theta:  vector of length n with starting values
10 c          n
11 c          ndim:   row dimension of h and al in calling routine
12 c          maxit:  maximum number of iterations to perform
13 c          eps:    convergence criterion, how close successive
14 c                  iterations must be to conclude that
15 c                  convergence has been reached.
16 c          grdhss: name of the subroutine that will calculate the
17 c                  gradient and Hessian (must be declared
18 c                  external in calling routine).
19 c
20 c   Work:
21 c         al:      (n by n) matrix having ndim rows
22 c         d, wk, del: vectors of length n
23 c
24 c   Output:
25 c         theta:   value of variables at last iteration
26 c         g, h:    gradient and Hessian at final iteration
27 c         ier:     error indicator (0 means none, -1 means no
28 c                  convergence, + means H not positive definite).
29 c
30 c   Subprograms Called: grdhss, solvch (mchol)
31 c
32 c-----------------------------------------------------------------
33
34        double precision theta(n),g(n),h(ndim,1),al(ndim,1),d(n),
35     1            wk(n),del(n)
36
37        double precision eps
38
39        do 100 it=1,maxit
40
41                call grdhss(theta,n,ndim,g,h)
```

```
42                call solvch(h,g,n,ndim,al,d,wk,del,ier)
43
44                if(ier.ne.0) return
45
46                do 10 i=1,n
47   10           if(abs(del(i)).ge.(eps*abs(theta(i)))) go to 20
48
49                ier=0
50                return
51
52   20           if(it.ge.maxit) then
53                        ier=-1
54                        return
55                endif
56
57                do 30 i=1,n
58   30           theta(i)=theta(i)-del(i)
59
60  100    continue
61
62        return
63        end
64
65
```

The Output

The computer program produces the following output:

```
Draper and Smith example: Gauss-Newton:

         alpha          beta        SS Res
-----------------------------------------
      0.300000      0.020000      0.026315
      0.841649      0.100682      4.488132
      0.390135      0.100419      0.005006
      0.390135      0.101605      0.005002
      0.390140      0.101632      0.005002
ier =     0

Draper and Smith example: Full Hessian:

         alpha          beta        SS Res
-----------------------------------------
      0.300000      0.020000      0.026315
      0.259564      0.024729      0.012877
ier =     2

For Weibull MLE example:

         alpha          beta        loglik
-----------------------------------------
      3.667686      1.321373      9.309643
      5.299813      1.648922     11.631004
      6.708105      1.811216     12.248871
      7.504770      1.883775     12.345123
      7.697009      1.899736     12.348808
      7.706138      1.900465     12.348829
ier =     0
```

Another Example: MLE's for the Logistic Distribution

Let $X_1, \ldots, X_n$ be a random sample from the logistic distribution having pdf

$$f(x) = \frac{e^{-y}}{(1+e^{-y})^2},$$

where $y = (x - \alpha)/\beta$ for $-\infty < x < \infty$, $-\infty < \alpha < \infty$, and $\beta > 0$. Then the negative of the log likelihood of $\theta = (\alpha, \beta)^T$ given the X's is given by

$$L(\theta) = n \log \beta + \sum_{i=1}^{n} y_i + 2 \sum_{i=1}^{n} \log(1 + e^{-y_i}),$$

where $y_i = (X_i - \alpha)/\beta$. From this we have that the gradient and Hessian are given by

$$g = \begin{bmatrix} -\dfrac{1}{\beta} \displaystyle\sum_{i=1}^{n} \dfrac{1 - z_i}{1 + z_i} \\ \dfrac{n}{\beta} - \dfrac{1}{\beta} \displaystyle\sum_{i=1}^{n} \dfrac{y_i(1 - z_i)}{1 + z_i} \end{bmatrix},$$

and

$$H = \begin{bmatrix} \dfrac{2}{\beta^2} \displaystyle\sum_{i=1}^{n} \dfrac{z_i}{(1 + z_i)^2} & \text{Sym} \\ \dfrac{1}{\beta^2} \displaystyle\sum_{i=1}^{n} \left[\dfrac{2 z_i y_i}{(1 + z_i)^2} + \dfrac{1 - z_i}{1 + z_i} \right] & \dfrac{2}{\beta^2} \displaystyle\sum_{i=1}^{n} \left[\dfrac{z_i y_i^2}{(1 + z_i)^2} + \dfrac{y_i(1 - z_i)}{1 + z_i} \right] - \dfrac{n}{\beta^2} \end{bmatrix},$$

where $z_i = e^{-y_i}$.

The subroutine `lgstgh` calculates g and H for a given θ. One thing to notice is that the line `zi=exp(-yi)` can lead to an error of `yi` is too large. Subroutine `lgstgh` handles this by skipping any term in the sum for which the absolute value of `yi` is too large.

```
        subroutine lgstgh(theta,nr,ndim,g,h)
c-----------------------------------------------------------------
c
c   Evaluate the gradient and Hessian for the negative
c   log likelihood of the prameters of a Logistic
c   Distribution based on a sample x of size n.
c
c-----------------------------------------------------------------

        dimension theta(nr),g(nr),h(ndim,nr)
        double precision yi,zi,opzi,omzi,ri,si
        common /logist/ x(100),n

        alpha=theta(1)
        beta=theta(2)

        g1=0.0
        g2=0.0
        h11=0.0
        h12=0.0
        h22=0.0

        do 10 i=1,n

                yi=(x(i)-alpha)/beta
                if(abs(yi).gt.60) go to 10
                zi=exp(-yi)
                opzi=1.+zi
                omzi=1.-zi
                ri=omzi/opzi
                si=zi/(opzi*opzi)

                g1=g1+ri
                g2=g2+yi*ri

                h11=h11+si
                h12=h12 + 2.*si*yi+ri
                h22=h22 + si*yi*yi+ri*yi

  10    continue

```

```
42          g(1)=-g1/beta
43          g(2)=(n-g2)/beta
44
45          beta2=beta*beta
46          h(1,1)=2*h11/beta2
47          h(1,2)=h12/beta2
48          h(2,1)=h(1,2)
49          h(2,2)=(2*h22-n)/beta2
50
51          return
52          end
```

There are two natural methods for finding starting values for the Newton-Raphson procedure. Since $\mathrm{E}(X) = \alpha$ and $\mathrm{Var}(X) = \beta^2\pi^2/3$, it is very easy to find the method of moments estimators

$$\hat{\alpha}_1 = \bar{X}, \qquad \hat{\beta}_1 = \sqrt{\frac{3\hat{\sigma}^2}{\pi^2}},$$

where $\hat{\sigma}^2 = \sum_{i=1}^{n}(X_i - \bar{X})^2/n$.

Since the quantile function of X is given by

$$Q(u) = \alpha - \beta \log\left(\frac{1-u}{u}\right),$$

we can estimate $Q(u)$ at $u_i = (i - 1/2)/n$ for $i = 1, \ldots, n$ by the order statistics $X_{(1)} < \cdots < X_{(n)}$ and then get another set of starting values $\hat{\alpha}_2$ and $\hat{\beta}_2$ by the simple linear regression (with intercept) of $X_{(i)}$ on $-\log((1-u_i)/u_i)$.

A Program Comparing Logistic Parameter Estimators

In the Fortran program listed below we do a simulation of three estimators; the method of moments, the 'regression of quantiles,' and the Newton-Raphson MLE's using the method of moments estimates as starting values. The program asks the user for values of α and β, for a random number generator seed, for a number of sample sizes, and the values of sample sizes, and the number of samples to generate for each sample size. It also asks for a value of an integer called `iopt` which should be 1 if the user wants the program to display the which sample it is doing every 100th sample and a 0 otherwise. This is useful since the program may execute for a long time if the number of replciations is large and printing every 100th sample number allows the user to monitor the program's progress.

For each sample size, the program calculates the average and mean square error (average squared distance from the true parameter value) for each of the three estimators of the two parameters. It also calculates the average of the standard errors of the MLE's of α and β where the standard error is calculated in two ways; first by evaluating the inverse of the information matrix

$$I(\theta) = \frac{n}{3\beta^2}\begin{bmatrix} 1 & 0 \\ 0 & 1+\pi^2/3 \end{bmatrix}.$$

at the MLE of β, and secondly from the inverse of the final Hessian matrix in the Newton-Raphson algorithm. For each sample these two methods of finding standard errors are used to see if the confidence interval for the parameters include the true values and the number of times it does is reported.

Finally, since it is possible for the Newton-Raphson method to not converge or for a Hessian to be not positive definite, the program keeps generating samples until the desired number of samples that have no Newton-Raphson errors are actually generated. Th program keeps track of the number of times te two types of errors occur (the variable `ier1` is the number of times the iterations didn't converge while `ier2` is the number of times a Hessian wasn't positive definite.

```
1           dimension g(10),h(10,10),theta(2),al(10,10),d(10),
```

```
     1           wk(10),del(10)
        dimension u(100),nn(10)
        character*40 infile
        real*4 smean,svar
        double precision dseed
        external lgstgh
        common /logist/ x(100),n
        data ndim/10/

        pi=4.0*atan(1.0)

c--------------------------------------
c   Get user input and open output file:
c--------------------------------------

        write(*,1)
   1    format(' Enter alpha, beta, dseed, nsamps, nreps, iopt: ',$)
        read(*,*) alpha,beta,dseed,nsamps,nreps,iopt
        write(*,3)
   3    format(' Enter sample sizes: ',$)
        read(*,*) (nn(i),i=1,nsamps)
        write(*,4)
   4    format(' Enter output file name: ',$)
        read(*,5) infile
   5    format(a40)

        open(1,file=infile,status='new')

        write(1,6)
   6    format(1x,'LOGISTIC SIMULATION SUMMARY'/,1x,27(1h-))
        write(1,7) alpha,beta,dseed
   7    format(1x,'True alpha = ',f10.6/,1x,'True beta =',f10.6/,
     1         1x,'seed = ',f12.0)
        write(1,8)
   8    format(/,27x,5hALPHA,17x,4hBETA/1x,55(1h-))
        write(1,9)
   9    format(1x,21x,4hMEAN,7x,3hMSE,6x,4hMEAN,7x,3hMSE/1x,55(1h-))

c-------------------------
c   Loop over sample size:
c-------------------------

        do 300 in=1,nsamps

        n=nn(in)
        write(1,10) n
  10    format(1x,'n = ',i3)

        do 11 i=1,n
        u(i)=(i-0.5)/n
  11    u(i)=-log((1.-u(i))/u(i))

        ier0=0
        ier1=0
        ier2=0

        a1bar=0.0
        a2bar=0.0
        a3bar=0.0
        a1mse=0.0
        a2mse=0.0
        a3mse=0.0
        b1bar=0.0
        b2bar=0.0
        b3bar=0.0
        b1mse=0.0
        b2mse=0.0
        b3mse=0.0
        vhabar=0.0
        vhbbar=0.0
        vvabar=0.0
        vvbbar=0.0
        ncova1=0
        ncova2=0
        ncovb1=0
        ncovb2=0

c------------------------
c   Loop over reps:
c------------------------
```

```
82
83          do 100 ir=1,nreps
84
85                  if(iopt.eq.1.and.(100*(ir/100).eq.ir)) write(*,*) n,ir
86
87    12            continue
88
89                  call rlogist(dseed,n,alpha,beta,x)
90
91  c   Method of Moments:
92
93                  a1=smean(x,n)
94                  b1=sqrt(3.*svar(x,n))/pi
95
96  c   Regression of quantiles:
97
98                  call qsort(x,n)
99                  call slr(u,x,n,a2,b2)
100
101 c   Newton-Raphson MLE:
102
103                 theta(1)=a1
104                 theta(2)=b1
105                 call nr(theta,2,ndim,20,.0001,lgstgh,g,h,al,d,wk,
106      1                  del,ier)
107                 a3=theta(1)
108                 b3=theta(2)
109
110                 if(ier.eq.0) ier0=ier0+1
111                 if(ier.lt.0) ier1=ier1+1
112                 if(ier.gt.0) ier2=ier2+1
113
114 c   Go back to get another sample if ier.ne.0:
115
116                 if(ier.ne.0) go to 12
117
118 c   Update means and mse's for the three estimators:
119
120                 a1bar=a1bar+a1
121                 b1bar=b1bar+b1
122                 a1mse=a1mse+(a1-alpha)**2
123                 b1mse=b1mse+(b1-beta)**2
124
125                 a2bar=a2bar+a2
126                 b2bar=b2bar+b2
127                 a2mse=a2mse+(a2-alpha)**2
128                 b2mse=b2mse+(b2-beta)**2
129
130                 a3bar=a3bar+a3
131                 b3bar=b3bar+b3
132                 a3mse=a3mse+(a3-alpha)**2
133                 b3mse=b3mse+(b3-beta)**2
134
135 c   Update mean of two estimates of standard errors:
136
137                 hd=h(1,1)*h(2,2)-h(1,2)*h(2,1)
138                 vha=h(2,2)/hd
139                 vhb=h(1,1)/hd
140                 vhabar=vhabar+vha
141                 vhbbar=vhbbar+vhb
142
143                 vva=3.*b3*b3/n
144                 vvb=3.*b3*b3/(n*(1.+pi*pi/3.))
145                 vvabar=vvabar+vva
146                 vvbbar=vvbbar+vvb
147
148 c   See if confidence intervals for two standard errors cover:
149
150                 a3i=abs(a3-alpha)
151                 b3i=abs(b3-beta)
152
153                 if(vha.le.0.0.or.vhb.le.0.0) go to 20
154
155                 if(a3i.lt.1.96*sqrt(vva))   ncova1=ncova1+1
156                 if(a3i.lt.1.96*sqrt(vha))   ncova2=ncova2+1
157                 if(b3i.lt.1.96*sqrt(vvb))   ncovb1=ncovb1+1
158                 if(b3i.lt.1.96*sqrt(vhb))   ncovb2=ncovb2+1
159
160   20            continue
161
```

```
c------------------
c   End reps loop:
c------------------

 100    continue

c--------------------------------------
c   Get true asymptotic standard error:
c--------------------------------------

        v1=3.*beta*beta/n
        v2=3.*beta*beta/(n*(1.+pi*pi/3.))

c-------------------------
c   Write out information:
c-------------------------

        write(1,110) a1bar/nreps,a1mse/nreps,b1bar/nreps,b1mse/nreps
 110    format(1x,12x,3hMOM,4f10.6)
        write(1,111) a2bar/nreps,a2mse/nreps,b2bar/nreps,b2mse/nreps
 111    format(1x,11x,4hQREG,4f10.6)
        write(1,112) a3bar/nreps,a3mse/nreps,b3bar/nreps,b3mse/nreps
 112    format(1x,12x,3hMLE,4f10.6)
        write(1,113) v1,v2
 113    format(1x,8x,7HTRUE SE,10x,f10.6,10x,f10.6)
        write(1,114) vvabar/nreps,vvbbar/nreps
 114    format(1x,4x,11hAVE INFO SE,10x,f10.6,10x,f10.6)
        write(1,115) vhabar/nreps,vhbbar/nreps
 115    format(1x,4x,11hAVE HESS SE,10x,f10.6,10x,f10.6)
        write(1,116) ier0,ier1,ier2
 116    format(1x,1x,14hIER0,IER1,IER2,3i10)
        write(1,117) ncova1,ncovb1
 117    format(1x,2x,13hINFO CI COVER,10x,i10,10x,i10)
        write(1,118) ncova2,ncovb2
 118    format(1x,2x,13hHESS CI COVER,10x,i10,10x,i10/1x,55(1h-))

 300    continue

        stop
        end
```

An Example Output

Here is the output for a sample run having 100,000 samples for samples of size 10, 20, 40, and 100. Note that the MLE seems to be slightly better than the other two estimators and that the observed MSE agrees quite well with the theoretical asymptotic variance even for the small sample sizes. Further, the confidence interval coverages seem close to the 95% nominal confidence coefficient.

```
LOGISTIC SIMULATION SUMMARY
---------------------------
True alpha =    .000000
True beta =  1.000000
seed =       12345.

                            ALPHA                 BETA
------------------------------------------------------------
                       MEAN       MSE       MEAN       MSE
------------------------------------------------------------
n =  10
             MOM   .002560   .327587   .961811   .076593
            QREG   .002560   .327587   .971834   .075911
             MLE   .002341   .306029   .940377   .071973
         TRUE SE             .300000             .069932
     AVE INFO SE             .285823             .066628
     AVE HESS SE             .291413             .066353
 IER0,IER1,IER2    100000        19        49
  INFO CI COVER                91016               86581
  HESS CI COVER                91222               86521
------------------------------------------------------------
n =  20
             MOM  -.001721   .164009   .980222   .038741
            QREG  -.001721   .164009   .982977   .037710
             MLE  -.001445   .151161   .969890   .035712
         TRUE SE             .150000             .034966
     AVE INFO SE             .146324             .034110
     AVE HESS SE             .147668             .034058
 IER0,IER1,IER2    100000        31         6
  INFO CI COVER                93052               90705
  HESS CI COVER                93146               90703
------------------------------------------------------------
n =  40
             MOM   .000132   .081978   .990527   .019696
            QREG   .000132   .081979   .990653   .019131
             MLE  -.000376   .075247   .985780   .017673
         TRUE SE             .075000             .017483
     AVE INFO SE             .074193             .017295
     AVE HESS SE             .074533             .017280
 IER0,IER1,IER2    100000        59         0
  INFO CI COVER                94065               92892
  HESS CI COVER                94119               92891
------------------------------------------------------------
n = 100
             MOM   .000353   .033357   .996293   .007935
            QREG   .000353   .033357   .995616   .007743
             MLE   .000466   .030556   .994402   .007008
         TRUE SE             .030000             .006993
     AVE INFO SE             .029874             .006964
     AVE HESS SE             .029927             .006962
 IER0,IER1,IER2    100000        97         0
  INFO CI COVER                94429               94148
  HESS CI COVER                94462               94146
------------------------------------------------------------
```