Statistics 520 Computing Notes
Maximum Likelihood using Newton Raphson

On the course web page in the Rfunctions folder is a file named `newtraph.txt` that contains a generic Newton-Raphson algorithm that you can use as an alternative to the built-in R functions `optim` or `nlm`. This function also serves as a basic code that can be modified for particular models that require additional information be provided to the optimization algorithm.

Syntax

The syntax of `newtraph` is

```
object<-newtraph(ders,dat,x0)
```

The arguments are

1. `ders`: the name of a function (you must write) to compute the log likelihood and derivatives for whatever model you are trying to fit – there is more information on this function below.

2. `dat`: the data, in whatever form is being expected by `ders` (`newtraph`) does not do anything with the data directly, it simply passes them along to the function `ders`.

3. `x0`: starting values for parameters – note that the order of the parameters in this argument should match the order derivatives are taken in the function `ders`.

The output of `newtraph` is a list consisting of three unnamed components with the following content.

1. `[[1]]`: A vector of estimates

2. `[[2]]`: The maximized log likelihood.
3. `[[3]]`: The estimated observed inverse information matrix.

The ders Function

The `ders` function must have syntax

```
object<-ders(pars,dat)
```

Here, `pars` is a vector of parameter values and `dat` is data. The `pars` argument should be parameters in the same order as the `x0` vector of starting values in `newtraph`. The data can be in any format as `newtraph` takes its argument of `dat` and simply passes it to `ders`.

The output of `ders` is expected to be a list of three components.

1. `[[1]]`: The log likelihood evaluated at the value of `pars`.
2. `[[2]]`: The gradient (vector of first derivatives of the log likelihood) evaluated at the value of `pars`.
3. `[[3]]`: The Hessian or matrix of second derivatives of the log likelihood evaluated at the value of `pars`. This should be organized such that the diagonal is the second derivative with respect to the first element of `pars`, the second, and so forth.

Additional Settings

There are a couple of additional values used internally by `newtraph` that you can edit the function to change. These are values for convergence criteria that are set in the function (to reduce the number of arguments that need to be given). These are set in the first three executable lines of `newtraph` as `crit1`, `crit2`, and `crit3`. These are convergence criteria for change in log likelihood, change in estimates (as Euclidean distance), and sum of first derivatives.

In addition, there is an automatic step-halving that is performed if the log likelihood fails to increase at a given iteration. Currently this will repeat 10 times but then crash the function to avoid false convergence. The value of 10 can be increased or decreased as desired.

Writing a ders Function

As an example exercise, let's write a `ders` function for use with a one-sample model of independent and identically distributed random variables $Y_1, \ldots, Y_n$ having a common beta distribution with parameters $\alpha > 0$ and $\beta > 0$. The first step is to write out on paper the expressions we need to compute. The density function of the random variables is

$$f(y|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\,\Gamma(\beta)}\, y^{\alpha-1}\,(1-y)^{\beta-1}; \quad 0 < y < 1. \tag{1}$$

By independence the joint density is

$$f(y_1,\ldots,y_n|\alpha,\beta) = \prod_{i=1}^{n} f(y_i|\alpha,\beta),$$

and the log likelihood is

$$\ell(\alpha,\beta) = \sum_{i=1}^{n} \ell_i(\alpha,\beta) = \sum_{i=1}^{n} \log\{f(y_i|\alpha,\beta)\}. \tag{2}$$

Using (1) in (2) gives

$$\begin{aligned} \ell(\alpha,\,\beta) &= (\alpha-1)\sum_{i=1}^{n}\log(y_i) + (\beta-1)\sum_{i=1}^{n}\log(1-y_i) \\ &+ n\log\{\Gamma(\alpha+\beta)\} - n\log\{\Gamma(\alpha)\} - n\log\{\Gamma(\beta)\}\,. \end{aligned} \tag{3}$$

Derivatives of this log likelihood are

$$\frac{\partial}{\partial\alpha}\ell(\alpha,\beta) = \sum_{i=1}^{n}\log(y_i) + n\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)} - n\frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$$

$$\begin{aligned}
\frac{\partial}{\partial\beta}\ell(\alpha,\beta) &= \sum_{i=1}^{n}\log(1-y_i) + n\frac{\Gamma'(\alpha+\beta)}{\Gamma(\alpha+\beta)} - n\frac{\Gamma'(\beta)}{\Gamma(\beta)} \\
\frac{\partial^2}{\partial\alpha^2}\ell(\alpha,\beta) &= n\left[\frac{\Gamma''(\alpha+\beta)}{\Gamma(\alpha+\beta)} - \frac{\{\Gamma'(\alpha+\beta)\}^2}{\{\Gamma(\alpha+\beta)\}^2} - \frac{\Gamma''(\alpha)}{\Gamma(\alpha} + \frac{\{\Gamma'(\alpha)\}^2}{\{\Gamma(\alpha)\}^2}\right] \\
\frac{\partial^2}{\partial\beta^2}\ell(\alpha,\beta) &= n\left[\frac{\Gamma''(\alpha+\beta)}{\Gamma(\alpha+\beta)} - \frac{\{\Gamma'(\alpha+\beta)\}^2}{\{\Gamma(\alpha+\beta)\}^2} - \frac{\Gamma''(\beta)}{\Gamma(\beta)} + \frac{\{\Gamma'(\beta)\}^2}{\{\Gamma(\beta)\}^2}\right] \\
\frac{\partial^2}{\partial\alpha\partial\beta}\ell(\alpha,\beta) &= n\left[\frac{\Gamma''(\alpha+\beta)}{\Gamma(\alpha+\beta)} - \frac{\{\Gamma'(\alpha+\beta)\}^2}{\{\Gamma(\alpha+\beta)\}^2}\right]. \qquad (4)
\end{aligned}$$

In (4),

$$\begin{aligned}
\Gamma(x) &= \int_0^\infty t^{x-1}\,\exp(-t)\,dt \\
\Gamma'(x) &= \int_0^\infty \log(t)\,t^{x-1}\,\exp(-t)\,dt \\
\Gamma''(x) &= \int_0^\infty \{\log(t)\}^2\,t^{x-1}\,\exp(-t)\,dt
\end{aligned}$$

So we need to write an R function that computes the log likelihood in (3) and all of the derivatives in (4) then arranges them in a list with th log likelihood as the first component, the first derivatives as the second component, and a matrix of the second derivatives as the third component. To compute the derivatives we will need to evaluate a number of integrals as given in (5), but notice that these will be used repeatedly.

Let's start by writing a separate function for the integrals. An examination of (4) shows that what we will need to produce are values for

$$\Gamma(\alpha), \Gamma(\beta), \Gamma(\alpha+\beta), \Gamma'(\alpha), \Gamma'(\beta), \Gamma'(\alpha+\beta), \Gamma''(\alpha), \Gamma''(\beta) \text{ and } \Gamma''(\alpha+\beta).$$

We can arrange the output of this function in any way we want it to used in our primary `ders` function.

```
dergamfctn<-function(ps){
  a<-ps[1]
  b<-ps[2]
  ga<-gamma(a)
  gb<-gamma(b)
  gab<-gamma(a+b)
  gprime<-function(x,par){ (x^{par-1})*(log(x))*exp(-x) }
  g2prime<-function(x,par){ (x^{par-1})*(log(x)^2)*exp(-x) }
  gpa<-integrate(gprime,0,Inf,par=a)$value
  gpb<-integrate(gprime,0,Inf,par=b)$value
  gpab<-integrate(gprime,0,Inf,par=(a+b))$value
  g2pa<-integrate(g2prime,0,Inf,par=a)$value
  g2pb<-integrate(g2prime,0,Inf,par=b)$value
  g2pab<-integrate(g2prime,0,Inf,par=(a+b))$value
  res1<-c(ga,gb,gab,gpa,gpb,gpab,g2pa,g2pb,g2pab)
  res<-matrix(res1,3,3,byrow=F)
  return(res)
}
```

Notice that we chose to arrange the results of this function to be a matrix

$$\begin{pmatrix} \Gamma(\alpha) & \Gamma'(\alpha) & \Gamma''(\alpha) \\ \Gamma(\beta) & \Gamma'(\beta) & \Gamma''(\beta) \\ \Gamma(\alpha+\beta) & \Gamma'(\alpha+\beta) & \Gamma''(\alpha+\beta) \end{pmatrix}$$

Now we can write our derivative function. Most of the computational work will be done by `dergamfctn`.

```
betaders<-function(ps, y){
n<-length(y)
        y1<-sum(log(y)); y2<-sum(log(1-y))
        iall<-dergamfctn(ps)
        gan<-iall[1,1]; gbn<-iall[2,1]; gabn<-iall[3,1]
        fa<-y1+n*((iall[3,2]/gabn)-(iall[1,2]/gan))
        fb<-y2+n*((iall[3,2]/gabn)-(iall[2,2]/gbn))
        fab<-n*(((iall[3,3]*gabn)-(iall[3,2]^2))/(gabn^2))
        faa<-fab-n*((((iall[1,3]*gan)-(iall[1,2]^2))/gan^2))
        fbb<-fab-n*((((iall[2,3]*gbn)-(iall[2,2]^2))/gbn^2))
        fm<-c(faa,fab,fab,fbb)
        dim(fm)<-c(2,2)
        logl<-((ps[1]-1)*y1)+((ps[2]-1)*y2)+n*(log(gabn)-log(gan)-log(gbn))
        res<-list(logl,c(fa,fb),fm)
        return(res)
}
```

Note that this function produces output as will be expected by `newtraph`.

An Example

As an exercise, try the following steps.

1. Simulate data from a beta distribution. You can vary the sample size and parameter values as you wish. Recall that for a beta distribution with parameters α and β the expected value and variance are

$$\begin{aligned} E(Y) &= \frac{\alpha}{\alpha+\beta} \\ var(Y) &= \frac{\alpha\,\beta}{(\alpha+\beta)^2\,(\alpha+\beta+1)}. \end{aligned}$$

   I'll use $n = 50$ and parameter values $\alpha = 6$ and $\beta = 4$ which gives expected value 0.60 and variance 0.0218.

```
> jdat<-rbeta(50,6,4)
> mean(jdat)
[1] 0.5844055
> var(jdat)
[1] 0.02091382
```

2. Now use `newtraph` to locate maximum likelihood estimates of α and β. This is what you will see appear on the screen. Notice that we did nothing fancy to determine starting values, just using 1 and 1.

```
> jout<-newtraph(betaders,jdat,c(1,1))
While N-R may be used for either minimization or maximization
the checks for progress in this function are written for maximization.
If you want to minimize, chage your derivative calculations (multiply by -1).

Current estimates beginning iteration 1:
1 1
Log likelihood for these estimates:
0
Gradient for these estimates:
21.3364
2.787431

Current estimates beginning iteration 2:
1.792185
1.566656
Log likelihood for these estimates:
12.44971
Gradient for these estimates:
```

```
10.14616

0.7066747



Current estimates beginning iteration 3:

3.032504

2.394281

Log likelihood for these estimates:

21.10905

Gradient for these estimates:

4.378904

0.1094328

    .

    .

    .

[SKIPPING SOME ADDITIONAL ITERATIONS]

    .

    .

    .

Current estimates beginning iteration 8 :

6.28447

4.486214

Log likelihood for these estimates:

26.58379

Gradient for these estimates:

4.205543e-09

-8.897203e-10
```

```
Convergence criterion of
1e-10
met for change in log likelihood
Convergence criterion of
1e-06
met for change in estimates
Convergence criterion of
1e-06
met for sum of derivatives

Final Estimates Are:
6.28447
4.486214

Final Log Likelihood:
26.58379

Value of Gradient at Convergence:
3.679553e-09
-7.788543e-10

Inverse Observed Information:

         [,1]      [,2]
[1,] 1.537104 0.9819610
[2,] 0.981961 0.7586436
```

Notice, in particular, the value of the final gradient. While we like for all three types of convergence to occur (likelihood, estimates, and gradient) we should keep in mind that Newton-Raphson is *designed* to make the gradient 0 – see the course notes Chapter 3.7.3.

3. We can easily compute additional inferential quantities from these results. For example, endpoints of a 95% Wald theory interval for α would be (see course notes Chapter 3.5.3 if needed).

```
> jout[[1]][1]-1.96*sqrt(jout[[3]][1,1])
[1] 3.854462
> jout[[1]][1]+1.96*sqrt(jout[[3]][1,1])
[1] 8.714478
```

   So an approximate 95% interval estimate of α based on asymptotic normality of maximum likelihood estimators in this problem would be

$$(3.854,\ 8.174)$$

   If it has not been communicated to you before this point, one keeps as many digits as the computer will for computations and then rounds when reporting answers. Thus, the value of $\hat{\alpha}$ might be reported here as $\hat{\alpha} = 6.28$, but in computation of an interval for α the object `jout[[1]][1]` is used, not the typed number 6.28

4. To produce a histogram of the data overlain by the estimated beta density function we first set up a sequence of values for evaluation, then compute the density at those values, produce a histogram of the data on a probability scale (always use the option `prob=T` with the R function `hist`).

```
> us<-seq(0.0001,0.9999,0.0001)
> fs<-dbeta(us,jout[[1]][1],jout[[1]][2])
> hist(jdat,prob=T,xlim=c(0,1),xlab="Variate Value",ylab="Density",main="")
> lines(us,fs)
```

This should produce a graph similar to Figure 1.

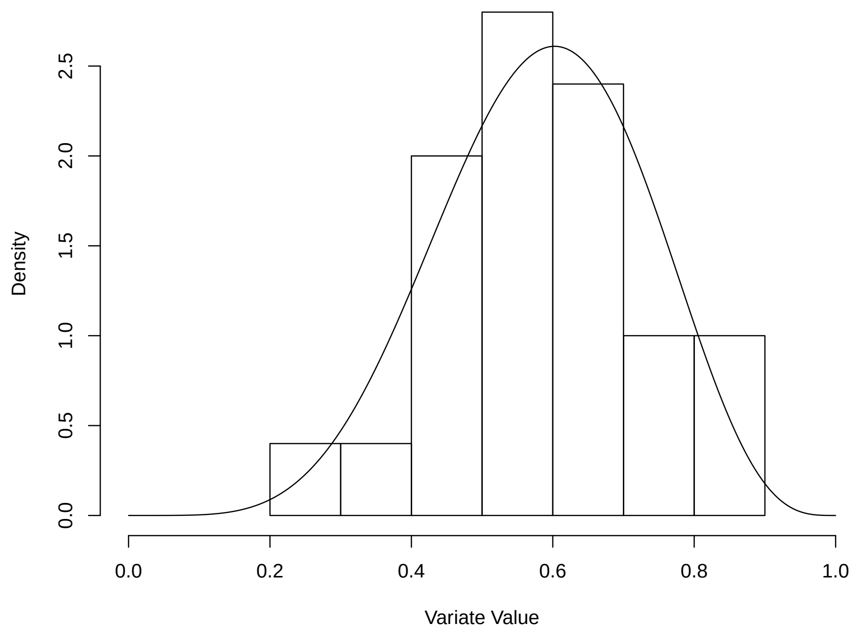

Figure 1: Graph of data and estimated density.