Chapter 1

Multivariate Maximization

1.1 Variations of Newton-Raphson

Let $\underset{\sim}{x} = [x_1, x_2, \ldots, x_p]^\top$ be a vector input to scalar-valued function $f(\underset{\sim}{x})$ that we wish to maximize. That is, $f : \mathbb{R}^p \to \mathbb{R}^1$. Define

$$g(\underset{\sim}{x}) \;=\; \mathrm{d}f(\underset{\sim}{x}) \;=\; \begin{bmatrix} g_1(\underset{\sim}{x}) \\ g_2(\underset{\sim}{x}) \\ \vdots \\ g_p(\underset{\sim}{x}) \end{bmatrix} \;=\; \begin{bmatrix} \partial f_1(\underset{\sim}{x})/\partial x_1 \\ \partial f_2(\underset{\sim}{x})/\partial x_2 \\ \vdots \\ \partial f_p(\underset{\sim}{x})/\partial x_p \end{bmatrix},$$

that is, $g(\underset{\sim}{x})$ is a column vector of partial derivatives of $f(\underset{\sim}{x})$.

A standard approach to maximizing $f(\underset{\sim}{x})$ is to solve the system of p possibly non-linear equations

$$g(\underset{\sim}{x}) \;=\; \begin{bmatrix} g_1(\underset{\sim}{x}) \\ g_2(\underset{\sim}{x}) \\ \vdots \\ g_p(\underset{\sim}{x}) \end{bmatrix} \;=\; \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \;\equiv\; \underset{\sim}{0}_p,$$

That is, locate roots of $g(\underset{\sim}{x})$ and check for maxima.

I will focus on variations of NR to solve $g(\underset{\sim}{x}) = \underset{\sim}{0}_p$. To generalize the 1-D NR, let

$$\begin{aligned}
\mathrm{d}g(\underset{\sim}{x}) &= \begin{bmatrix} \mathrm{d}g_1(\underset{\sim}{x}) \\ \mathrm{d}g_2(\underset{\sim}{x}) \\ \vdots \\ \mathrm{d}g_p(\underset{\sim}{x}) \end{bmatrix} \\
&= \begin{bmatrix} \partial^2 f(\underset{\sim}{x})/\partial x_1^2 & \partial^2 f(\underset{\sim}{x})/\partial x_1 \partial x_2 & \cdots & \partial^2 f(\underset{\sim}{x})/\partial x_1 \partial x_p \\ \partial^2 f(\underset{\sim}{x})/\partial x_2 \partial x_1 & \partial^2 f(\underset{\sim}{x})/\partial x_2^2 & \cdots & \partial^2 f(\underset{\sim}{x})/\partial x_2 \partial x_p \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f(\underset{\sim}{x})/\partial x_p \partial x_1 & \partial^2 f(\underset{\sim}{x})/\partial x_p \partial x_2 & \cdots & \partial^2 f(\underset{\sim}{x})/\partial x_p^2 \end{bmatrix}_{p\text{-by-}p}
\end{aligned}$$

be the p-by-p matrix of second partial derivatives. If $\underset{\sim}{x}^* = [x_1^*, x_2^*, \ldots, x_p^*]^\top$ is a root of $g(\underset{\sim}{x})$ (that is, $g(\underset{\sim}{x}^*) = \underset{\sim}{0}_p$) then for $\underset{\sim}{x}$ near $\underset{\sim}{x}^*$ a linear Taylor series expansion gives

$$\begin{aligned}
\underset{\sim}{0}_p &= g(\underset{\sim}{x}^*) \doteq g(\underset{\sim}{x}) + [\,\mathrm{d}g(\underset{\sim}{x})](\underset{\sim}{x}^* - \underset{\sim}{x}) \qquad \text{or} \\
\underset{\sim}{x}^* &\doteq \underset{\sim}{x} - [\,\mathrm{d}g(\underset{\sim}{x})]^{-1} g(\underset{\sim}{x}),
\end{aligned}$$

assuming the inverse exists.

This suggests an iterative scheme for approximating $\underset{\sim}{x}^*$ From an initial guess $\underset{\sim}{x}_0$:

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{x}_i - [\,\mathrm{d}g(\underset{\sim}{x}_i)]^{-1} g(\underset{\sim}{x}_i), \quad i = 0, 1, \ldots.$$

Keep in mind that $\underset{\sim}{x}_i$ is a vector and $[\,\mathrm{d}g(\underset{\sim}{x}_i)]$ is a matrix.

An absolute convergence criterion would lead to iterating until

$$|\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i| = \sqrt{\sum_{\ell=1}^{p} (x_{\ell,i+1} - x_{\ell,i})^2} < \varepsilon.$$

If the root $\underset{\sim}{x}^*$ is far from $\underset{\sim}{0}_p$, then it makes sense to use a relative convergence criterion

$$\frac{|\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i|}{|\underset{\sim}{x}_i|} \; < \; \varepsilon.$$

Using either criterion, we iterate until change in the estimated root between steps is small.

Remarks

1. If for some i we have $\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i = \underset{\sim}{0}$, then $[\,\mathrm{d}g(\underset{\sim}{x}_i)]^{-1} g(\underset{\sim}{x}_i) = \underset{\sim}{0}$, which implies $g(\underset{\sim}{x}_i) = \underset{\sim}{0}$. That is, we found a root.

2. The relative convergence criterion is usually preferred to the absolute convergence criterion.

3. Convergence to a root is quadratic, assuming the initial guess is close to $\underset{\sim}{x}^*$.

4. If you are trying to maximize $f(\underset{\sim}{x})$, you should check that $\underset{\sim}{x}^*$ corresponds to a (local or global) maximum and not a minimum!

5. The initial guess is very important. In many statistical problems there is a natural starting point.

6. NR Can be interpreted geometrically in terms of iterative tangent plane approximations.

7. Instead of computing $\underset{\sim}{x}_{i+1}$ from $\underset{\sim}{x}_i$ via

$$\underset{\sim}{x}_{i+1} \; = \; \underset{\sim}{x}_i - [\,\mathrm{d}g(\underset{\sim}{x}_i)]^{-1} g(\underset{\sim}{x}_i),$$

it is preferable to avoid matrix inversion and directly solve

$$[\,\mathrm{d}g(\underset{\sim}{x}_i)](\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i) \;=\; -g(\underset{\sim}{x}_i)$$

for $(\underset{\sim}{x}_{i+1} - \underset{\sim}{x}_i)$, which leads to $\underset{\sim}{x}_{i+1}$.

8. NR is one a large collection of iterative schemes of the form

$$\underset{\sim}{x}_{i+1} \;=\; \underset{\sim}{x}_i - \mathbf{J}_i^{-1} g(\underset{\sim}{x}_i), \quad i = 0, 1, \ldots,$$

for a suitably defined p-by-p matrix $\mathbf{J}_i$.

- The multivariate secant method sets $\mathbf{J}_i \doteq \mathrm{d}g(\underset{\sim}{x})$ evaluated numerically.
- Rescaled simple iteration takes

$$\begin{aligned} \mathbf{J}_i \;&=\; \mathbf{J} \qquad \text{(same for each iteration)} \\ &= \begin{bmatrix} \alpha_1 & & 0 \\ & \alpha_2 & \\ & \ddots & \\ 0 & & \alpha_p \end{bmatrix} \end{aligned}$$

where

$$\alpha_\ell \;=\; \left.\frac{\partial g_\ell(\underset{\sim}{x})}{\partial x_\ell}\right|_{\underset{\sim}{x}_0}$$

is the partial of g_ℓ with respect to element x_ℓ evaluated at the initial guess $\underset{\sim}{x}_0$.

The point to recognize is that if such an iteration converges, then from remark (1) above, we know it converges to a root $\underset{\sim}{x}^*$.

9. If we relate NR to our original objective of maximizing $f(\underset{\sim}{x})$ then the NR iteration has to form

$$\underset{\sim}{x}_{i+1} \;=\; \underset{\sim}{x}_i - [\,\mathrm{d}^2 f(\underset{\sim}{x}_i)]^{-1}\,\mathrm{d}f(\underset{\sim}{x}_i), \quad i = 0, 1, \ldots,$$

where

$$\begin{aligned} \mathrm{d}f(\underset{\sim}{x}_i) &= p\text{-by-1 vector of partial derivatives evaluated at } \underset{\sim}{x}_i \\ \mathrm{d}^2 f(\underset{\sim}{x}_i) &= p\text{-by-}p \text{ matrix of second partial derivatives.} \end{aligned}$$

NR tells us that starting from $\underset{\sim}{x}_i$ to move in the direction of

$$[\,\mathrm{d}^2 f(\underset{\sim}{x}_i)]^{-1}\,\mathrm{d}f(\underset{\sim}{x}_i)$$

to get the $(i{+}1)$th step estimate $\underset{\sim}{x}_{i+1}$. However, there is no guarantee that

$$f(\underset{\sim}{x}_{i+1}) \;=\; f(\underset{\sim}{x}_i - [\,\mathrm{d}^2 f(\underset{\sim}{x}_i)]^{-1}\,\mathrm{d}f(\underset{\sim}{x}_i)) \;>\; f(\underset{\sim}{x}_i).$$

That is, there is no guarantee that we are increasing the function value as the iteration proceeds.

A popular modification of NR is to consider

$$\underset{\sim}{x}_{i+1}(\alpha) \;=\; \underset{\sim}{x}_i - \alpha[\,\mathrm{d}^2 f(\underset{\sim}{x}_i)]^{-1}\,\mathrm{d}f(\underset{\sim}{x}_i),$$

where the "step-size" scalar α is chosen to maximize $f(\underset{\sim}{x}_{i+1}(\alpha))$. Finding α that maximizes $f(\underset{\sim}{x}_{i+1}(\alpha))$ for a given $\underset{\sim}{x}_i$ is a single variable maximization problem. In practice, it usually suffices to descretize α, that is, set

$$\alpha \;=\; -1, -0.9, \ldots, -0.1, [\text{not zero}], 0.1, 0.2, \ldots, 1[\text{NR}], 1.1, \ldots, 2$$

and maximize $f(\underset{\sim}{x}_{i+1}(\alpha))$ over this grid. Once you find the maximizing value $\alpha_{\max}$, you set

$$\underset{\sim}{x}_{i+1} \;=\; \underset{\sim}{x}_i - \alpha_{\max}[\,\mathrm{d}^2 f(\underset{\sim}{x}_i)]^{-1}\,\mathrm{d}f(\underset{\sim}{x}_i)$$

and continue iterating.

This modification slows down NR, but usually leads to a much more stable algorithm that is less likely to wander off far from a maximum.

1.2 Maximum likelihood estimate (MLE)

Suppose we have a random variable $\underset{\sim}{Y} = [Y_1, Y_2, \ldots, Y_n]^\top$ with the probability density or mass function that depends on $\underset{\sim}{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^\top$, say $\Pr[\underset{\sim}{y}|\underset{\sim}{\theta}] = \Pr[Y_1, Y_2, \ldots, Y_n|\underset{\sim}{\theta}]$. The Ys may be a random sample with common distribution $h(y_i|\underset{\sim}{\theta})$ and if so

$$\Pr[Y_1, Y_2, \ldots, Y_n|\underset{\sim}{\theta}] \;=\; \prod_{i=1}^{n} h(y_i|\underset{\sim}{\theta}).$$

However, the setup is more general, allowing for arbitrary joint distributions. The *likelihood function* for $\underset{\sim}{\theta}$ given data $\underset{\sim}{Y} = [Y_1, Y_2, \ldots, Y_n]^\top$ is

$$L(\underset{\sim}{\theta}) \;=\; \Pr[\underset{\sim}{y}|\underset{\sim}{\theta}]$$

and the *log-likelihood function* is

$$\ell(\underset{\sim}{\theta}) \;=\; \log(L(\underset{\sim}{\theta})).$$

The MLE of $\underset{\sim}{\theta}$, say $\hat{vtheta}$, Is the value that minimizes $L(\underset{\sim}{\theta})$, or equivalently $\ell(\underset{\sim}{\theta})$. Typically, MLEs are obtained by solving the likelihood equations

$$\text{Score function:} \quad \dot{\ell}(\underset{\sim}{\theta}) \;=\; \begin{bmatrix} \partial\ell_1(\underset{\sim}{x})/\partial\theta_1 \\ \partial\ell_2(\underset{\sim}{x})/\partial\theta_2 \\ \vdots \\ \partial\ell_p(\underset{\sim}{x})/\partial\theta_p \end{bmatrix} \;=\; \underset{\sim}{0}_p.$$

Let

$$\ddot{\ell}(\underset{\sim}{\theta}) \;=\; \left[\frac{\partial^2 \ell}{\partial\theta_i \partial\theta_j}\right]_{p\text{-by-}p} \qquad i\text{th row, } j\text{th column element}$$

Be the matrix of second partial derivatives of $\ell(\underset{\sim}{\theta})$ with respect to elements of $\underset{\sim}{\theta}$. Following Remark 9 on page 5, one might consider the following NR procedure to compute $\hat{\underset{\sim}{\theta}}$:

$$\hat{\underset{\sim}{\theta}}_{i+1} \;=\; \hat{\underset{\sim}{\theta}}_i - [\ddot{\ell}(\hat{\underset{\sim}{\theta}}_i)]^{-1}\dot{\ell}(\hat{\underset{\sim}{\theta}}_i), \quad i = 0, 1, \ldots.$$

Note that convergence to a root $\hat{\underset{\sim}{\theta}}^*$ implies $\dot{\ell}(\hat{\underset{\sim}{\theta}}^*) = 0$, as desired. Of course, we need to check whether the root is the MLE!

An alternative iterative procedure is known as **Fisher's Method of Scoring**. Thinking of $\ddot{\ell}(\underset{\sim}{\theta})$ as a random variable (that is, it depends on a random $\underset{\sim}{Y}$) define the *expected Fisher information matrix* as

$$\mathbf{I}(\underset{\sim}{\theta}) \;=\; \mathrm{E}[-\ddot{\ell}(\underset{\sim}{\theta})] \;=\; \mathrm{E}[\dot{\ell}(\underset{\sim}{\theta})\dot{\ell}^\top(\underset{\sim}{\theta})],$$

where the last equality follows under "standard conditions".

The Method of Scoring replaces $-\ddot{\ell}(\hat{\underset{\sim}{\theta}}_i)$ by $\mathbf{I}(\underset{\sim}{\theta})$ in the iteration

$$\hat{\underset{\sim}{\theta}}_{i+1} \;=\; \hat{\underset{\sim}{\theta}}_i + [\mathbf{I}(\hat{\underset{\sim}{\theta}})]^{-1}\dot{\ell}(\hat{\underset{\sim}{\theta}}_i), \quad i = 0, 1, \ldots.$$

This is an example of Remark 8 on page 4.

The NR adjustment $-[\ddot{\ell}(\hat{\underset{\sim}{\theta}}_i)]^{-1}\dot{\ell}(\hat{\underset{\sim}{\theta}}_i)$ Is a function of the derivative of $\ell(\underset{\sim}{\theta})$ relative to the second derivative. We saw in the one-dimensional example of maximizing $f(x) = \log(x)/(1+x)$ that the adjustment can overshoot the root or lead to moving slowly to the root. These tend to occur when $\ell(\underset{\sim}{\theta})$ is either very peaked or very flat near the maximum. One possible remedy is to use the average value of the second derivative, $\mathbf{I}(\underset{\sim}{\theta})$, instead in the iteration.

Both NR and Scoring may be improved by adding a step-size parameter α, for example

$$\hat{\underset{\sim}{\theta}}_{i+1} \;=\; \hat{\underset{\sim}{\theta}}_i - \alpha[\ddot{\ell}(\hat{\underset{\sim}{\theta}}_i)]^{-1}\dot{\ell}(\hat{\underset{\sim}{\theta}}_i),$$

where α is chosen to maximize $\ell(\hat{\underset{\sim}{\theta}}_{i+1}(\alpha))$ for fixed $\hat{\underset{\sim}{\theta}}_i$. Standard distribution theory for MLEs shows that

$$\hat{\underset{\sim}{\theta}} \;\sim\; \text{Normal}_p(\underset{\sim}{\theta}, \text{Var}[\hat{\underset{\sim}{\theta}}])$$

Under suitable conditions, where

$$\text{Var}[\hat{\underset{\sim}{\theta}}] \;=\; \mathbf{I}^{-1}(\underset{\sim}{\theta})$$

can be estimated by either

$$\begin{aligned} \hat{\text{Var}}[\hat{\underset{\sim}{\theta}}] \;&=\; \mathbf{I}^{-1}(\hat{\underset{\sim}{\theta}}) \qquad \text{inverse of expected Fisher information at MLE} \\ &=\; -[\ddot{\ell}(\hat{\underset{\sim}{\theta}}_i)]^{-1} \qquad \text{observed information matrix at MLE.} \end{aligned}$$

In some cases the two estimates agree. There is no general consensus on which estimator is to be preferred. Most knowledgeable statisticians tend to use the observed information matrix.

I will consider two examples of computing MLEs, a single parameter case and a multiparameter case.

Example: Multinomial with one parameter Suppose $\underset{\sim}{Y} = [Y_1, Y_2, \ldots, Y_n]^{\top}$ has a multinomial distribution with sample size m and probabilities $p_i(\theta)$ that depend on a single parameter $\theta > 0$, with pmf

$$\Pr[\underset{\sim}{Y}|\theta] \;=\; \frac{m!}{\prod_{i=1}^n y_i!}\prod_{i=1}^n p_i(\theta)^{y_i}.$$

The log-likelihood, ignoring the constant, is

$$\begin{aligned} \ell(\theta) &= \log \left\{ \prod_{i=1}^{n} p_i(\theta)^{y_i} \right\} \\ &= \sum_{i}^{n} y_i \log(p_i(\theta)). \end{aligned}$$

The MLE is obtained by solving the likelihood equation

$$\begin{aligned} \dot{\ell}(\theta) &= \sum_{i}^{n} y_i \frac{\partial}{\partial \theta} \log(p_i(\theta)) \\ &= \sum_{i}^{n} y_i \frac{p_i'(\theta)}{p_i(\theta)} \qquad \text{where } p_i'(\theta) = \frac{\partial p_i(\theta)}{\partial \theta}. \end{aligned}$$

Rather than do things in general, I will consider the following genetics problem[1] as a classic example of maximum likelihood estimation due to Fisher (1925). Let $n = 4$ cells have class probabilities given by

$$\begin{aligned} p_1(\theta) &= (2+\theta)/4 \\ p_2(\theta) = p_3(\theta) &= (1-\theta)/4 \\ p_4(\theta) &= \theta/4 \end{aligned}$$

where $0 < \theta < 1$. The parameter θ is to be estimated from the observed frequencies $\underset{\sim}{Y} = [1997, 906, 904, 32]^\top$ from a sample of size $m = 3839$.

The log-likelihood function is

$$\begin{aligned} \ell(\theta) &= \sum_{i}^{4} y_i \log(p_i(\theta)) \\ &= 4\log(0.25) + y_1 \log(2+\theta) + (y_2 + y_3)\log(1-\theta) + y_4 \log(\theta), \end{aligned}$$

[1] Ronald Thisted (1988) Elements of Statistical Computing. pp. 175–6.

so its derivatives are given by

$$\begin{aligned}\dot{\ell}(\theta) &= \frac{y_1}{2+\theta} - \frac{y_2+y_3}{1-\theta} + \frac{y_4}{\theta} \\ \ddot{\ell}(\theta) &= -\frac{y_1}{(2+\theta)^2} - \frac{y_2+y_3}{(1-\theta)^2} - \frac{y_4}{\theta^2}.\end{aligned}$$

To get $\mathbf{I}(\theta)$, treat y_is as random variables in $\ddot{\ell}(\theta)$ and recall that $Y_i \sim$ Binomial$(m, p_i(\theta))$. So, $\mathrm{E}[Y_i] = mp_i(\theta)$, which implies

$$\begin{aligned}\mathbf{I}(\theta) &= \mathrm{E}[-\ddot{\ell}(\theta)] \\ &= m\left\{\frac{p_1(\theta)}{(2+\theta)^2} + \frac{p_2(\theta)+p_3(\theta)}{(1-\theta)^2} + \frac{p_4(\theta)}{\theta^2}\right\} \\ &= 0.25m\left\{\frac{1}{2+\theta} + \frac{2}{1-\theta} + \frac{1}{\theta}\right\}.\end{aligned}$$

Note that for this example

$$\begin{aligned}\dot{\ell}(\theta) &= \frac{y_1}{2+\theta} - \frac{y_2+y_3}{1-\theta} + \frac{y_4}{\theta} \\ &= \frac{y_1(1-\theta)\theta - (y_2+y_3)(2+\theta)\theta + y_4(2+\theta)(1-\theta)}{(2+\theta)(1-\theta)\theta}.\end{aligned}$$

The numerator is a quadratic function of θ, so the likelihood equation

$$\dot{\ell}(\theta) = 0$$

has two roots. It can also be shown that one root is negative, so the only candidate for the MLE is the positive root. Although the roots can be found analytically, it is informative to see whether NR and Scoring converge. Note that even though the score function is defined for $\theta < 0$, the log-likelihood function is not.

The coding for this example is very simple because we have a single parameter. Thus, NR iterates as follows

$$\hat{\theta}_{i+1} \;=\; \hat{\theta}_i - \frac{\dot{\ell}(\hat{\theta}_i)}{\ddot{\ell}(\hat{\theta}_i)}, \quad i = 0, 1, \ldots,$$

while Fisher scoring iterates via

$$\hat{\theta}_{i+1} \;=\; \hat{\theta}_i + \frac{\dot{\ell}(\hat{\theta}_i)}{\mathbf{I}(\hat{\theta})}, \quad i = 0, 1, \ldots.$$

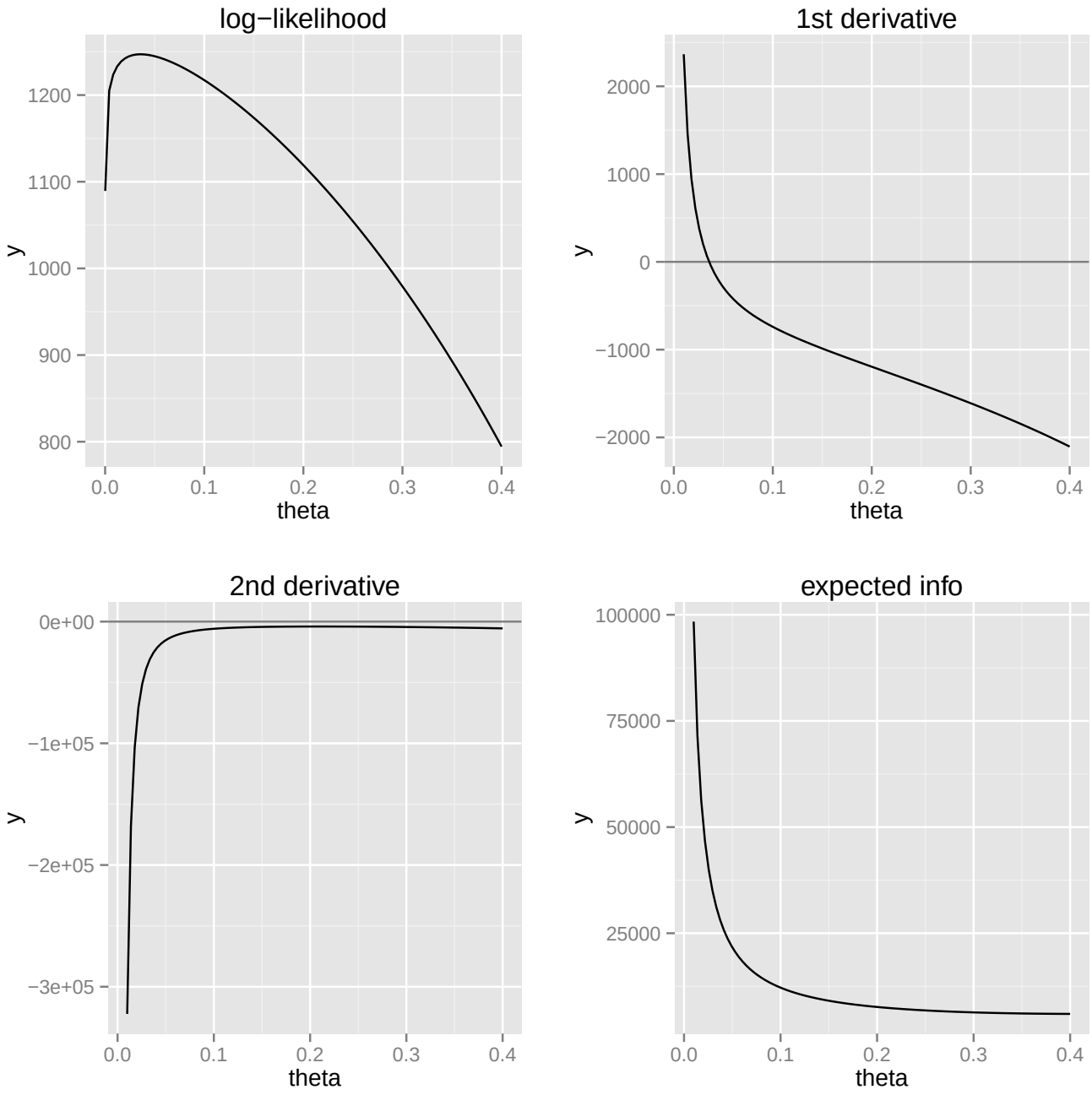
Implementation, NR The script below defines the likelihood function, and related derivatives for Thisted's multinomial example. I have placed `theta` as the first argument so the `ggplot` function `stat_function` can use that variable as the x-axis with the `y` variable passed as an additional argument.

```
# maximizing multinomial likelihood
y <- c(1997, 906, 904, 32)
m <- sum(y)

# functions: log-likelihood, 1st derivative, 2nd derivative, and expected info
f.l <- function(theta, y) {
  temp <- y[1] * log(2 + theta) +
          (y[2] + y[3]) * log(1 - theta) +
          y[4] * log(theta)
  return(temp)
}
f.dl <- function(theta, y) {
  temp <- y[1] / (2+theta) +
          - (y[2] + y[3]) / (1 - theta) +
          y[4] / theta
  return(temp)
}
f.ddl <- function(theta, y) {
  temp <- - (y[1] / (2 + theta)^2 +
             (y[2] + y[3]) / (1-theta)^2 +
             y[4] / theta^2
```

```
            )
  return(temp)
}
f.info <- function(theta, y) {
  temp <- 0.25 * sum(y) * (1 / (2 + theta) +
                           2 / (1 - theta) +
                           1 / theta )
  return(temp)
}
```

Notice that the log-likelihood increases rapidly from zero to a maximum at approximately 0.05.

```
# plot functions
library(ggplot2)
p1 <- ggplot(data.frame(theta = c(0.0001, 0.4)), aes(theta))
p1 <- p1 + stat_function(fun = f.l, args = list(y))
p1 <- p1 + labs(title = "log-likelihood")
#print(p1)

p2 <- ggplot(data.frame(theta = c(0.01, 0.4)), aes(theta))
p2 <- p2 + geom_hline(yintercept = 0, alpha = 0.5)
p2 <- p2 + stat_function(fun = f.dl, args = list(y))
p2 <- p2 + labs(title = "1st derivative")
#print(p2)

p3 <- ggplot(data.frame(theta = c(0.01, 0.4)), aes(theta))
p3 <- p3 + geom_hline(yintercept = 0, alpha = 0.5)
p3 <- p3 + stat_function(fun = f.ddl, args = list(y))
p3 <- p3 + labs(title = "2nd derivative")
#print(p3)

p4 <- ggplot(data.frame(theta = c(0.01, 0.4)), aes(theta))
p4 <- p4 + stat_function(fun = f.info, args = list(y))
p4 <- p4 + labs(title = "expected info")
#print(p4)

library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol=2)
```



Because we have an explicit and relatively simple expression for the derivative of the score function, NR is a logical candidate for the iterative method. The function below is effectively the same as the NR function from the chapter on Function Maximization, except for the `y` argument for the observed frequencies. Note that the functions `g` and `gp` defined in the function are replaced by the functions `f.dl` and `f.ddl` passed to the

`f.NR()` function.

```
# NR routine for finding root of g(x) = 0.
# Requires predefined g(x) and gp(x) = deriv of g(x)
# The iteration is controlled by:
#   eps   = absolute convergence criterion
#   maxit = maximum allowable number of iterations
# Input:  xnew = user prompted starting value
# Output: number of root, steps, and note
f.NR <- function(g, gp, xnew = 1, eps = 0.001, maxit = 35, y = c(1,1,1,1)) {
  xold <- -Inf   # needed so argument in while() loop is defined

  i <- 1;   # initial iteration index

  NR.hist <- data.frame(i, xnew, diff = abs(xnew - xold)) # iteration history
  while ((i <= maxit) & (abs(xnew - xold) > eps)) {
    i <- i + 1                          # increment iteration
    xold <- xnew                        # old guess is current guess
    xnew <- xold - g(xold, y) / gp(xold, y)  # new guess

    NR.hist <- rbind(NR.hist, c(i, xnew, abs(xnew - xold))) # iteration history
  }

  out <- list()
  out$root <- xnew
  out$iter <- i
  out$hist <- NR.hist
  if (abs(xnew - xold) <= eps) {
    out$note <- paste("Absolute convergence of", eps, "satisfied")
  }
  if (i > maxit) {
    out$note <- paste("Exceeded max iterations of ", maxit)
  }
  return(out)
}
```

A few illustrations of our NR function follow.

```
out0.01 <- f.NR(f.dl, f.ddl, xnew = 0.01, y = y)
out0.01


## $root
## [1] 0.03571
```

```
##
## $iter
## [1] 6
##
## $hist
##   i    xnew      diff
## 1 1 0.01000       Inf
## 2 2 0.01734 0.0073377
## 3 3 0.02647 0.0091313
## 4 4 0.03344 0.0069732
## 5 5 0.03558 0.0021373
## 6 6 0.03571 0.0001323
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"
```

```
out0.05 <- f.NR(f.dl, f.ddl, xnew = 0.05, y = y)
out0.05
```

```
## $root
## [1] 0.0357
##
## $iter
## [1] 4
##
## $hist
##   i    xnew      diff
## 1 1 0.05000       Inf
## 2 2 0.03095 0.0190512
## 3 3 0.03512 0.0041720
## 4 4 0.03570 0.0005826
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"
```

```
out0.20 <- f.NR(f.dl, f.ddl, xnew = 0.20, y = y)
out0.20
```

```
## $root
## [1] -0.4668
##
## $iter
## [1] 6
##
```

```
## $hist
##   i     xnew      diff
## 1 1  0.20000       Inf
## 2 2 -0.09568 0.2956825
## 3 3 -0.26453 0.1688450
## 4 4 -0.44285 0.1783252
## 5 5 -0.46669 0.0238361
## 6 6 -0.46681 0.0001253
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"

out0.40 <- f.NR(f.dl, f.ddl, xnew = 0.40, y = y)
out0.40

## $root
## [1] 0.0357
##
## $iter
## [1] 5
##
## $hist
##   i    xnew      diff
## 1 1 0.40000       Inf
## 2 2 0.02246 0.3775390
## 3 3 0.03098 0.0085169
## 4 4 0.03513 0.0041502
## 5 5 0.03570 0.0005755
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"

out0.50 <- f.NR(f.dl, f.ddl, xnew = 0.50, y = y)
out0.50

## $root
## [1] -0.4668
##
## $iter
## [1] 7
##
## $hist
##   i    xnew      diff
## 1 1  0.5000       Inf
```

```
## 2 2  0.1413 0.3586592
## 3 3 -0.0699 0.2112391
## 4 4 -0.1985 0.1286382
## 5 5 -0.4080 0.2094407
## 6 6 -0.4659 0.0578853
## 7 7 -0.4668 0.0009514
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"
```

We see that if the starting value is less than 0.05 that the routine converges to 0.0357. If the start value is 0.2 or above, the routine sometimes converges to 0.0357, and sometimes converges to a negative root -0.4668.

Using the positive root as the MLE, the estimated standard deviation of the MLE is approximately 0.006.

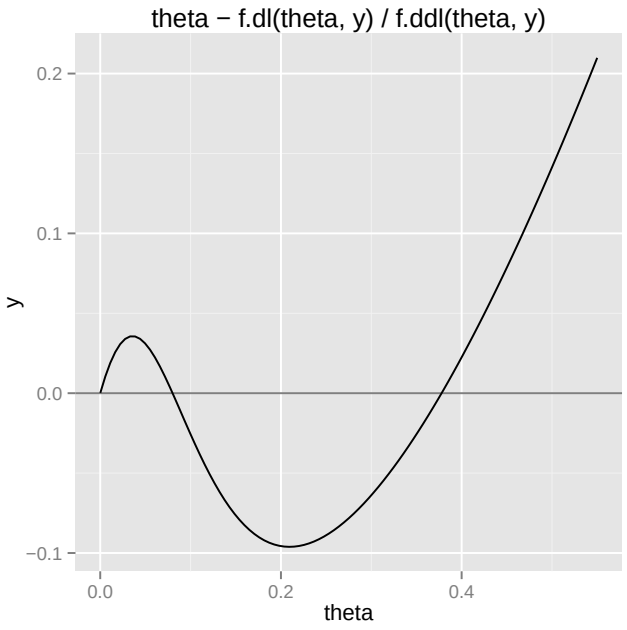
```
# estimated standard deviation via Fisher's information
sqrt(1/f.info(out0.05$root, y))

## [1] 0.005838

# estimated standard deviation via second derivative
sqrt(-1/f.ddl(out0.05$root, y))

## [1] 0.006027
```

To understand why the routine converges from a starting point of 0.4, but not from a starting point of 0.2, I plotted the function $\theta - \dot{\ell}(\theta)/\ddot{\ell}(\theta)$. This function gives the next guess for the root when the current guess is θ. Looking at this function, we see that for starting values between about 0.08 and 0.38 the next guess is negative, while for starting values near zero or 0.4 the next guess is near the root. If the starting value is near 0.5 or greater, the next guess is near 0.15, so the third guess will be negative!

One might be led to a simple choice of $\theta = 0.5$ by simply noting that θ must be in $(0, 1)$ and by taking the midpoint of that interval.

This "easy way out" of the starting-value problem leads to disaster for Newton's method, which converges to the wrong root! This difficulty is easily avoided by plotting the log-likelihood before selecting a starting value, as we have done.

```
# plot functions
library(ggplot2)
p <- ggplot(data.frame(theta = c(0.0001, 0.55)), aes(theta))
p <- p + geom_hline(yintercept = 0, alpha = 0.5)
p <- p + stat_function(fun = function(theta, y)
                           {theta - f.dl(theta, y) / f.ddl(theta, y)}
                       , args = list(y))
p <- p + labs(title = "theta - f.dl(theta, y) / f.ddl(theta, y)")
print(p)
```



Implementation, Fisher's Scoring For comparison, we shall also show how the method of scoring performs.

To perform the Fisher's Scoring, a simple replacement from `-f.ddl` to `f.info` is needed.

```
# Fisher's scoring routine for finding root of g(x) = 0.
# Requires predefined g(x) and gp(x) = deriv of g(x)
# The iteration is controlled by:
#   eps   = absolute convergence criterion
#   maxit = maximum allowable number of iterations
# Input:  xnew = user prompted starting value
# Output: number of root, steps, and note
f.FS <- function(g, gp, xnew = 1, eps = 0.001, maxit = 35, y = c(1,1,1,1)) {
  xold <- -Inf   # needed so argument in while() loop is defined

  i <- 1;   # initial iteration index

  NR.hist <- data.frame(i, xnew, diff = abs(xnew - xold)) # iteration history
  while ((i <= maxit) & (abs(xnew - xold) > eps)) {
    i <- i + 1                          # increment iteration
    xold <- xnew                        # old guess is current guess
    xnew <- xold + g(xold, y) / gp(xold, y)  # new guess

    NR.hist <- rbind(NR.hist, c(i, xnew, abs(xnew - xold))) # iteration history
  }

  out <- list()
  out$root <- xnew
  out$iter <- i
  out$hist <- NR.hist
  if (abs(xnew - xold) <= eps) {
    out$note <- paste("Absolute convergence of", eps, "satisfied")
  }
  if (i > maxit) {
    out$note <- paste("Exceeded max iterations of ", maxit)
  }
  return(out)
}
```

A few illustrations of our Fisher's Scoring follow.

```
out0.01 <- f.FS(f.dl, f.info, xnew = 0.01, y = y)
out0.01


## $root
## [1] 0.03571
##
## $iter
```

```
## [1] 4
##
## $hist
##   i    xnew      diff
## 1 1 0.01000       Inf
## 2 2 0.03404 2.404e-02
## 3 3 0.03561 1.569e-03
## 4 4 0.03571 9.753e-05
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"
```

```
out0.05 <- f.FS(f.dl, f.info, xnew = 0.05, y = y)
out0.05
```

```
## $root
## [1] 0.03577
##
## $iter
## [1] 3
##
## $hist
##   i    xnew      diff
## 1 1 0.05000       Inf
## 2 2 0.03657 0.0134256
## 3 3 0.03577 0.0008088
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"
```

```
out0.20 <- f.FS(f.dl, f.info, xnew = 0.20, y = y)
out0.20
```

```
## $root
## [1] 0.03574
##
## $iter
## [1] 4
##
## $hist
##   i    xnew      diff
## 1 1 0.20000       Inf
## 2 2 0.04350 0.1564991
## 3 3 0.03619 0.0073130
```

```
## 4 4 0.03574 0.0004461
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"

out0.40 <- f.FS(f.dl, f.info, xnew = 0.40, y = y)
out0.40

## $root
## [1] 0.03576
##
## $iter
## [1] 4
##
## $hist
##   i    xnew      diff
## 1 1 0.40000       Inf
## 2 2 0.04914 0.3508553
## 3 3 0.03652 0.0126207
## 4 4 0.03576 0.0007615
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"

out0.50 <- f.FS(f.dl, f.info, xnew = 0.50, y = y)
out0.50

## $root
## [1] 0.03577
##
## $iter
## [1] 4
##
## $hist
##   i    xnew      diff
## 1 1 0.50000       Inf
## 2 2 0.05112 0.4488799
## 3 3 0.03664 0.0144800
## 4 4 0.03577 0.0008704
##
## $note
## [1] "Absolute convergence of 0.001 satisfied"
```

In all cases, Fisher's Scoring method converged quickly to the correct root. This can be understood by looking at a plot of $\theta + \dot{\ell}(\theta)/\mathbf{I}(\theta)$, which gives the next guess in the Scoring routine from the current guess at θ. For θ between about 0 and 0.5, the next guess is very close to the root, so the method converges rapidly!

```
# plot functions
library(ggplot2)
p <- ggplot(data.frame(theta = c(0.0001, 0.55)), aes(theta))
p <- p + geom_hline(yintercept = 0, alpha = 0.5)
p <- p + stat_function(fun = function(theta, y)
                           {theta + f.dl(theta, y) / f.info(theta, y)}
                       , args = list(y))
p <- p + labs(title = "theta + f.dl(theta, y) / f.info(theta, y)")
print(p)
```