

# The Newton Raphson Algorithm for Function Optimization

Kevin Quinn

Assistant Professor

Department of Political Science and

The Center for Statistics and the Social Sciences

Box 354322, Padelford Hall

University of Washington

Seattle, WA 98195-4322

October 25, 2001

# 1 Introduction

Throughout this course we will be interested in calculating maximum likelihood estimates (MLEs). Such estimates are often extremely complicated nonlinear functions of the observed data. As a result, closed form expressions for the MLEs will generally not exist for the models we are working with.

The Newton Raphson algorithm is an iterative procedure that can be used to calculate MLEs. The basic idea behind the algorithm is the following. First, construct a quadratic approximation to the function of interest around some initial parameter value (hopefully close to the MLE). Next, adjust the parameter value to that which maximizes the quadratic approximation. This procedure is iterated until the parameter values stabilize.

These notes begin with the easy to visualize case of maximizing a function of one variable. After this case is developed, we turn to the more general case of maximizing a function of  $k$  variables.

## 2 The Newton Raphson Algorithm for Finding the Maximum of a Function of 1 Variable

### 2.1 Taylor Series Approximations

The first part of developing the Newton Raphson algorithm is to devise a way to approximate the likelihood function with a function that can be easily maximized analytically. To do this we need to make use of Taylor's Theorem.

**Theorem 1 (Taylor's Theorem (1 Dimension)).** *Suppose the function  $f$  is  $k + 1$  times differentiable on an open interval  $I$ . For any points  $x$  and  $x + h$  in  $I$  there exists a point  $w$  between  $x$  and  $x + h$  such that*

$$f(x + h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \dots + \frac{1}{k!}f^{[k]}(x)h^k + \frac{1}{(k + 1)!}f^{[k+1]}(w)h^{k+1}. \quad (1)$$

It can be shown that as  $h$  goes to 0 the higher order terms in equation 1 go to 0 much faster than  $h$  goes to 0. This means that (for small values of  $h$ )

$$f(x + h) \approx f(x) + f'(x)h$$

This is referred to as a *first order Taylor approximation of  $f$  at  $x$* . A more accurate approximation to  $f(x + h)$  can be constructed for small values of  $h$  as:

$$f(x + h) \approx f(x) + f'(x)h + \frac{1}{2}f''(x)h^2$$

This is known as a *second order Taylor approximation of  $f$  at  $x$* .

Note that the first order Taylor approximation can be rewritten as:

$$f(x + h) \approx a + bh$$

where  $a = f(x)$  and  $b = f'(x)$ . This highlights the fact that the first order Taylor approximation is a linear function in  $h$ .

Similarly, the second order Taylor approximation can be rewritten as:

$$f(x+h) \approx a + bh + \frac{1}{2}ch^2$$

where  $a = f(x)$ ,  $b = f'(x)$ , and  $c = f''(x)$ . This highlights the fact that the second order Taylor approximation is a second order polynomial in  $h$ .

## 2.2 Finding the Maximum of a Second Order Polynomial

Suppose we want to find the value of  $x$  that maximizes

$$f(x) = a + bx + cx^2.$$

First, we calculate the first derivative of  $f$ :

$$f'(x) = b + 2cx$$

We know that  $f'(\hat{x}) = 0$ , where  $\hat{x}$  is the value of  $x$  at which  $f$  attains its maximum. In other words, we know that

$$0 = b + 2c\hat{x}.$$

Solving for  $\hat{x}$  we find that  $\hat{x} = -\frac{b}{2c}$ . The second order condition is  $f''(x) = 2c < 0$ . This implies that  $f(-\frac{b}{2c})$  will be a maximum whenever  $c < 0$ .

## 2.3 The Newton Raphson Algorithm

Suppose we want to find the value of  $x$  that maximizes some twice continuously differentiable function  $f(x)$ .

Recall

$$f(x+h) \approx a + bh + \frac{1}{2}ch^2$$

where  $a = f(x)$ ,  $b = f'(x)$ , and  $c = f''(x)$ . This implies

$$f'(x+h) \approx b + ch.$$

The first order condition for the value of  $h$  (denoted  $\hat{h}$ ) that maximizes  $f(x+h)$  is

$$0 = b + c\hat{h}$$

Which implies  $\hat{h} = -\frac{b}{c}$ . In other words, the value that maximizes the second order Taylor approximation to  $f$  at  $x$  is

$$\begin{aligned} x + \hat{h} &= x - \frac{b}{c} \\ &= x - \frac{1}{f''(x)} f'(x) \end{aligned}$$

With this in mind we can specify the Newton Raphson algorithm for 1 dimensional function optimization.

**Algorithm 2.1:** NEWTONRAPHSO1D( $f, x_0, tolerance$ )

**comment:** Find the value  $\hat{x}$  of  $x$  that maximizes  $f(x)$

$i \leftarrow 0$

**while**  $|f'(x_i)| > tolerance$

**do**  $\begin{cases} i \leftarrow i + 1 \\ x_i \leftarrow x_{i-1} - \frac{1}{f''(x_{i-1})} f'(x_{i-1}) \end{cases}$

$\hat{x} \leftarrow x_i$

**return** ( $\hat{x}$ )

*Caution:* Note that the Newton Raphson Algorithm doesn't check the second order conditions necessary for  $\hat{x}$  to be a maximizer. This means that if you give the algorithm a bad starting value for  $x_0$  you may end up with a min rather than a max.

## 2.4 Example: Calculating the MLE of a Binomial Sampling Model

To see how the Newton Raphson algorithm works in practice lets look at a simple example with an analytical solution– a simple model of binomial sampling.

Our log-likelihood function is:

$$\ell(\pi|y) = y \ln(\pi) + (n - y) \ln(1 - \pi)$$

where  $n$  is the sample size,  $y$  is the number of successes, and  $\pi$  is the probability of a success. The first derivative of the log-likelihood function is

$$\ell'(\pi|y) = \frac{y}{\pi} + (n - y) \frac{-1}{1 - \pi}$$

and the second derivative of the log-likelihood function is

$$\ell''(\pi|y) = -\frac{y}{\pi^2} - \frac{n - y}{(1 - \pi)^2}.$$

Analytically, we know that the MLE is  $\hat{\pi} = \frac{y}{n}$ .

For the sake of example, suppose  $n = 5$  and  $y = 2$ . Analytically, we know that the MLE is  $\hat{\pi} = \frac{y}{n} = 0.4$ . Let's see how the Newton Raphson algorithm works in this situation.

We begin by setting a tolerance level. In this case, let's set it to 0.01 (In practice you probably want something closer to 0.00001). Next we make an initial guess (denoted  $\pi_0$ ) as to the MLE. Suppose  $\pi_0 = 0.55$ .  $\ell'(\pi_0|y) \approx -3.03$  which is larger in absolute value than our tolerance of 0.01. Thus we set

$$\pi_1 \leftarrow \pi_0 - \frac{1}{\ell''(\pi_0|y)} \ell'(\pi_0|y) \approx 0.40857.$$

Now we calculate  $\ell'(\pi_1|y) \approx -0.1774$  which is still larger in absolute value than our tolerance of 0.01. Thus we set

$$\pi_2 \leftarrow \pi_1 - \frac{1}{\ell''(\pi_1|y)} \ell'(\pi_1|y) \approx 0.39994.$$

$\ell'(\pi_2|y)$  is approximately equal to 0.0012 which is smaller in absolute value than our tolerance of 0.01 so we can stop. The Newton Raphson algorithm here returns a value of  $\hat{p}_i$  equal to 0.39994 which is reasonably close to the analytical value of 0.40. Note we can make the Newton Raphson procedure more accurate (within machine precision) by setting the tolerance level closer to 0.

### 3 The Newton Raphson Algorithm for Finding the Maximum of a Function of $k$ Variables

#### 3.1 Taylor Series Approximations in $k$ Dimensions

Consider a function  $f : \mathbb{R}^k \rightarrow \mathbb{R}$  that is at least twice continuously differentiable. Suppose  $\mathbf{x} \in \mathbb{R}^k$  and  $\mathbf{h} \in \mathbb{R}^k$ . Then the first order Taylor approximation to  $f$  at  $\mathbf{x}$  is given by

$$f(\mathbf{x} + \mathbf{h}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})' \mathbf{h}$$

and the second order Taylor approximation to  $f$  at  $\mathbf{x}$  is given by

$$f(\mathbf{x} + \mathbf{h}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})' \mathbf{h} + \frac{1}{2} \mathbf{h}' D^2 f(\mathbf{x}) \mathbf{h}$$

where  $\nabla f(\mathbf{x})$  is the gradient (vector of first derivatives)  $f$  at  $\mathbf{x}$ , and  $D^2 f(\mathbf{x})$  is the Hessian (matrix of second derivatives) of  $f$  at  $\mathbf{x}$ .

#### 3.2 Finding the Maximum of a Second Order Polynomial in $k$ Variables

Consider

$$f(\mathbf{x}) = a + \mathbf{b}' \mathbf{x} + \mathbf{x}' \mathbf{C} \mathbf{x}$$

where  $a$  is a scalar,  $\mathbf{b}$  and  $\mathbf{x}$  are  $k$ -vectors, and  $\mathbf{C}$  is a  $k \times k$  symmetric, negative definite matrix. The gradient of  $f$  at  $\mathbf{x}$  is

$$\nabla f(\mathbf{x}) = \mathbf{b} + 2\mathbf{C}\mathbf{x}$$

Since the gradient at the value that maximizes  $f$  will be a vector of zeros we know that the maximizer  $\hat{\mathbf{x}}$  satisfies

$$\mathbf{0} = \mathbf{b} + 2\mathbf{C}\hat{\mathbf{x}}$$

Solving for  $\hat{\mathbf{x}}$  we find that

$$\hat{\mathbf{x}} = -\frac{1}{2} \mathbf{C}^{-1} \mathbf{b}.$$

Since  $\mathbf{C}$  is assumed to be negative definite we know that this is a maximum.

### 3.3 The Newton Raphson Algorithm in $k$ Dimensions

Suppose we want to find the  $\hat{\mathbf{x}} \in \mathbb{R}^k$  that maximizes the twice continuously differentiable function  $f : \mathbb{R}^k \rightarrow \mathbb{R}$ .

Recall

$$f(\mathbf{x} + \mathbf{h}) \approx a + \mathbf{b}'\mathbf{h} + \frac{1}{2}\mathbf{h}'\mathbf{C}\mathbf{h}$$

where  $a = f(\mathbf{x})$ ,  $\mathbf{b} = \nabla f(\mathbf{x})$ , and  $\mathbf{C} = D^2 f(\mathbf{x})$ . Note that  $\mathbf{C}$  will be symmetric. This implies

$$\nabla f(\mathbf{x} + \mathbf{h}) \approx \mathbf{b} + \mathbf{C}\mathbf{h}.$$

Once again, the first order condition for a maximum is

$$\mathbf{0} = \mathbf{b} + \mathbf{C}\hat{\mathbf{h}}$$

which implies that

$$\hat{\mathbf{h}} = \mathbf{C}^{-1}\mathbf{b}$$

In other words, the vector that maximizes the second order Taylor approximation to  $f$  at  $\mathbf{x}$  is

$$\begin{aligned}\mathbf{x} + \hat{\mathbf{h}} &= \mathbf{x} - \mathbf{C}^{-1}\mathbf{b} \\ &= \mathbf{x} - (D^2 f(\mathbf{x}))^{-1} \nabla f(\mathbf{x})\end{aligned}$$

With this in mind we can specify the Newton Raphson algorithm for  $k$ -dimensional function optimization.

**Algorithm 3.1:** NEWTONRAPHSONKD( $f, \mathbf{x}_0, tolerance$ )

**comment:** Find the value  $\hat{\mathbf{x}}$  of  $\mathbf{x}$  that maximizes  $f(\mathbf{x})$

$i \leftarrow 0$

**while**  $\|\nabla f(\mathbf{x}_i)\| > tolerance$

**do**  $\begin{cases} i \leftarrow i + 1 \\ \mathbf{x}_i \leftarrow \mathbf{x}_{i-1} - (D^2 f(\mathbf{x}_{i-1}))^{-1} \nabla f(\mathbf{x}_{i-1}) \end{cases}$

$\hat{\mathbf{x}} \leftarrow \mathbf{x}_i$

**return** ( $\hat{\mathbf{x}}$ )