Lab02: Review of Optimization with Newton's Method

Newton's Method for root finding

One example of a root finding algorithm

- Given $f(x)$ and an initial guess for the root, x_0, a better approximation for the root is:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

- More generally, iterate until convergence:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Note: f needs to be differentiable and x_0 needs to be close to a true root. Newton's Method can fail if initial point is chosen poorly.

When to stop iterating?

One decision rule you could use. Stop if

$$|x_{n+1} - x_n| < \epsilon$$

An example

Write our the first 3 iterations of Newton's method for $f(x) = x^2 + 5x - 10$ with initial guess $x_0 = 4$

Answer

Need derivative: $f'(x) = 2x + 5$

So,

$x_1 = 4 - (26/13) = 2$

$x_2 = 2 - (4/9) = 1.555$

$x_3 = 1.555 - (.193/8.11) = 1.5312$

Let's draw it out

One iteration with iterate x_n:

1. Find $f(x_n)$
2. Draw tangent
3. x_{n+1} is where tangent intersects x-axis

Newton's Method for optimization

Convert our root-finding problem into an *optimization problem* by considering the *derivative* of a one dimentional function $f'(x)$.

- Basic idea: find where $f'(x) = 0$ to identify stationary point (e.g., minima & maxima).
- Our updates are now of the form:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

- Should we draw this out?

Note f must be twice differentiable.

What we really care about: the multivariable case

For maximum likelihood estimation, we need to find the roots of the first derivative of the loglikelihood $\ell(\theta)$.

$$\theta_{n+1} = \theta_n - \frac{\ell'(\theta_n)}{\ell''(\theta_n)}$$

- Usually for us, θ is a vector. For example, $\theta = (\beta_0, ..., \beta_p)$.
- In that case, we need to consider the *gradient* vector and the *Hessian* matrix of second partial derivatives:

$$\theta_{n+1} = \theta_n - [\nabla^2 f(\theta_n)]^{-1} \nabla f(\theta_n)$$

Disclaimer

A major downside to Newton's method is you have to invert a matrix. (Note for your homework it is fine to use the `solve` function in R.)

Other optimization methods like Fisher Scoring get around this, but that's beyond today's lab.

How would do this?

Consider $X_1, ..., X_n \overset{\text{iid}}{\sim} Gamma(\alpha, \beta)$

$$f(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

- Let's assume $\alpha = 1$, so our concern is β:

$$f(x; \alpha = 1, \beta) = \beta e^{-\beta x}$$

- Our goal is to find the MLE of β using Newton's Method.

Derive the update

$$L(\beta) = \beta^n e^{-\beta \sum_{i=1}^n x_i}$$

$$\ell(\beta) = n \log(\beta) - \beta \sum_{i=1}^n x_i$$

$$\ell'(\beta) = \frac{n}{\beta} - \sum_{i=1}^n x_i$$

$$\ell''(\beta) = -\frac{n}{\beta^2}$$

Code it up

```r
l_p <- function(beta, X) # what goes here??

l_pp <- function(beta, X) # what goes here??

newtons_method <- function(beta, X, eps){
    abs_change <- 2*eps #initial value to start while loop
    beta_old <- beta #initial value to start while loop

    # continue while absolute change in interates is still large
    while(abs_change > eps){
        # update
        beta <- # what goes here??
        # calculate change
        abs_change <- abs(beta - beta_old)
        # prep next iteration
        beta_old <- beta
        # let's check out our successive iterates
        print(beta)
    }
    return(beta)
}
```

Check our code with simulated data

```
X <- rgamma(n = 100, shape = 1, rate = 3)
newtons_method(beta = 1, X = X, eps = .001)
```