Unit 11: Optimization

November 5, 2015

References:

- Gentle: *Computational Statistics*
- Lange: *Optimization*
- Monahan: *Numerical Methods of Statistics*
- Givens and Hoeting: *Computational Statistics*
- Materials online from Stanford's EE364a course on convex optimization, including Boyd and Vandenberghe's (online) book Convex Optimization, which is also linked to from the course webpage.

1 Notation

We'll make use of the first derivative (the gradient) and second derivative (the Hessian) of functions. We'll generally denote univariate and multivariate functions (without distinguishing between them) as $f(x)$ with $x = (x_1, \ldots, x_p)$. The (column) vector of first partial derivatives (the gradient) is $f'(x) = \nabla f(x) = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_p})^\top$ and the matrix of second partial derivatives (the Hessian) is

$$f''(x) = \nabla_f^2(x) = H_f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_p} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_p} \\ \vdots & \vdots & \ddots & \\ \frac{\partial^2 f}{\partial x_1 \partial x_p} & \frac{\partial^2 f}{\partial x_2 \partial x_p} & \cdots & \frac{\partial^2 f}{\partial x_p^2} \end{pmatrix}.$$

In considering iterative algorithms, I'll use x_0, $x_1, \ldots, x_t$, x_{t+1} to indicate the sequence of values as we search for the optimum, denoted x^*. x_0 is the starting point, which we must choose (often

carefully). If it's unclear at any point whether I mean a value of x in the sequence or a sub-element of the x vector, let me know, but hopefully it will be clear from context most of the time.

I'll try to use x (or if we're talking explicitly about a likelihood, θ) to indicate the argument with respect to which we're optimizing and Y to indicate data involved in a likelihood. I'll try to use z to indicate covariates/regressors so there's no confusion with x.

2 Overview

The basic goal here is to optimize a function numerically when we cannot find the maximum (or minimum) analytically. Some examples:

1. Finding the MLE for a GLM
2. Finding least squares estimates for a nonlinear regression model,

$$Y_i \sim \mathcal{N}(g(z_i; \beta), \sigma^2)$$

   where $g(\cdot)$ is nonlinear and we seek to find the value of $\theta = (\beta, \sigma^2)$ that best fits the data.

3. Maximizing a likelihood under constraints

Maximum likelihood estimation and variants thereof is a standard situation in which optimization comes up.

We'll focus on **minimization**, since any maximization of f can be treated as minimization of $-f$. The basic setup is to find the *argument*, x, that minimizes $f(x)$:

$$\arg\min_{x \in D} f(x)$$

where D is the domain. Sometimes $D = \Re^p$ but other times it imposes constraints on x. When there are no constraints, this is unconstrained optimization, where any x for which $f(x)$ is defined is a possible solution. We'll assume that f is continuous as there's little that can be done systematically if we're dealing with a discontinuous function.

In one dimension, minimization is the same as root-finding with the derivative function, since the minimum of a differentiable function can only occur at a point at which the derivative is zero. So with differentiable functions we'll seek to find x s.t. $f'(x) = \nabla f(x) = 0$. To ensure a minimum, we want that for all y in a neighborhood of x^*, $f(y) \leq f(x^*)$, or (for twice differentiable functions) $f''(x^*) = \nabla^2 f(x^*) = H_f(x^*) \geq 0$, i.e., that the Hessian is positive semi-definite.

Different strategies are used depending on whether D is discrete and countable, or continuous, dense and uncountable. We'll concentrate on the continuous case but the discrete case can arise in statistics, such as in doing variable selection.

In general we rely on the fact that we can evaluate f. Often we make use of analytic or numerical derivatives of f as well.

To some degree, optimization is a solved problem, with good software implementations, so it raises the question of how much to discuss in this class. The basic motivation for going into some of the basic classes of optimization strategies is that the function being optimized changes with each problem and can be tricky to optimize, and I want you to know something about how to choose a good approach when you find yourself with a problem requiring optimization. Finding global, as opposed to local, minima can also be an issue.

Note that I'm not going to cover MCMC (Markov chain Monte Carlo) methods, which are used for approximating integrals and sampling from posterior distributions in a Bayesian context and in a variety of ways for optimization. If you take a Bayesian course you'll cover this in detail, and if you don't do Bayesian work, you probably won't have much need for MCMC, though it comes up in MCEM (Monte Carlo EM) and simulated annealing, among other places.

Goals for the unit Optimization is a big topic. Here's what I would like you to get out of this:

1. an understanding of line searches,
2. an understanding of multivariate derivative-based optimization and how line searches are useful within this,
3. an understanding of derivative-free methods,
4. an understanding of the methods used in R's optimization routines, their strengths and weaknesses, and various tricks for doing better optimization in R, and
5. a basic idea of what convex optimization is and when you might want to go learn more about it.

3 Univariate function optimization

We'll start with some strategies for univariate functions. These can be useful later on in dealing with multivariate functions.

3.1 Golden section search

This strategy requires only that the function be unimodal.

Assume we have a single minimum, in $[a, b]$. We choose two points in the interval and evaluate them, $f(x_1)$ and $f(x_2)$. If $f(x_1) < f(x_2)$ then the minimum must be in $[a, x_2]$, and if the converse in $[x_1, b]$. We proceed by choosing a new point in the new, smaller interval and iterate. At each step we reduce the length of the interval in which the minimum must lie. The primary question involves what is an efficient rule to use to choose the new point at each iteration.

Suppose we start with x_1 and x_2 s.t. they divide $[a, b]$ into three equal segments. Then we use $f(x_1)$ and $f(x_2)$ to rule out either the leftmost or rightmost segment based on whether $f(x_1) < f(x_2)$. If we have divided equally, we cannot place the next point very efficiently because either x_1 or x_2 equally divides the remaining space, so we are forced to divide the remaining space into relative lengths of 0.25, 0.25, and 0.5. The next time around, we may only rule out the shorter segment, which leads to inefficiency.

The efficient strategy is to maintain the *golden ratio* between the distances between the points using $\phi = (\sqrt{5} - 1)/2 \approx .618$, the golden ratio. We start with $x_1 = a + (1 - \phi)(b - a)$ and $x_2 = a + \phi(b - a)$. Then suppose $f(x_1) < f(x_2)$. We now choose to place x_3 s.t. it uses the golden ratio in the interval $[a, x_1]$: $x_3 = a + (1 - \phi)(x_2 - a)$. Because of the way we've set it up, we once again have the third subinterval, $[x_1, x_2]$, of equal length as the first subinterval, $[a, x_3]$. The careful choice allows us to narrow the search interval by an equal proportion,$1 - \phi$, in each iteration. Eventually we have narrowed the minimum to between x_{t-1} and x_t, where the difference $|x_t - x_{t-1}|$ is sufficiently small (within some tolerance - see Section 4 for details), and we report $(x_t + x_{t-1})/2$. We'll see an example of this on the board in class.

3.2 Bisection method

The bisection method requires the existence of the first derivative but has the advantage over the golden section search of halving the interval at each step. We again assume unimodality.

We start with an initial interval (a_0, b_0) and proceed to shrink the interval. Let's choose a_0 and b_0, and set x_0 to be the mean of these endpoints. Now we update according to the following algorithm, assuming our current interval is $[a_t, b_t]$:

$$[a_{t+1}, b_{t+1}] = \begin{cases} [a_t, x_t] & \text{if} f'(a_t)f'(x_t) < 0 \\ [x_t, b_t] & \text{if} f'(a_t)f'(x_t) > 0 \end{cases}$$

and set x_{t+1} to the mean of a_{t+1} and b_{t+1}. The basic idea is that if the derivative at both a_t and x_t is negative, then the minimum must be between x_t and b_t, based on the intermediate value theorem.

If the derivatives at a_t and x_t are of different signs, then the minimum must be between a_t and x_t.

Since the bisection method reduces the size of the search space by one-half at each iteration, one can work out that each decimal place of precision requires 3-4 iterations. Obviously bisection is more efficient than the golden section search because we reduce by $0.5 > 0.382 = 1 - \phi$, so we've gained information by using the derivative. It requires an evaluation of the derivative however, while golden section just requires an evaluation of the original function.

Bisection is an example of a *bracketing* method, in which we trap the minimum within a nested sequence of intervals of decreasing length. These tend to be slow, but if the first derivative is continuous, they are robust and don't require that a second derivative exist.

3.3 Newton-Raphson (Newton's method)

3.3.1 Overview

We'll talk about Newton-Raphson (N-R) as an optimization method rather than a root-finding method, but they're just different perspectives on the same algorithm.

For N-R, we need two continuous derivatives that we can evaluate. The benefit is speed, relative to bracketing methods. We again assume the function is unimodal. The minimum must occur at x^* s.t. $f'(x^*) = 0$, provided the second derivative is non-negative at x^*. So we aim to find a zero (a root) of the first derivative function. Assuming that we have an initial value x_0 that is close to x^*, we have the Taylor series approximation

$$f'(x) \approx f'(x_0) + (x - x_0)f''(x_0).$$

Now set $f'(x) = 0$, since that is the condition we desire (the condition that holds when we are at x^*), and solve for x to get

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

and iterate, giving us updates of the form $x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$. What are we doing intuitively? Basically we are taking the tangent to $f(x)$ at x_0 and extrapolating along that line to where it crosses the x-axis to find x_1. We then reevaluate $f(x_1)$ and continue to travel along the tangents.

One can prove that if $f'(x)$ is twice continuously differentiable, is convex, and has a root, then N-R converges from any starting point.

Note that we can also interpret the N-R update as finding the analytic minimum of the quadratic Taylor series approximation to $f(x)$.

Newton's method converges very quickly (as we'll discuss in Section 4), but if you start too far from the minimum, you can run into serious problems.

3.3.2 Secant method variation on N-R

Suppose we don't want to calculate the second derivative required in the divisor of N-R. We might replace the analytic derivative with a discrete difference approximation based on the secant line joining $(x_t, f'(x_t))$ and $(x_{t-1}, f'(x_{t-1}))$, giving an approximate second derivative:

$$f''(x_t) \approx \frac{f'(x_t) - f'(x_{t-1})}{x_t - x_{t-1}}.$$

For this variant on N-R, we need two starting points, x_0 and x_1.

An alternative to the secant-based approximation is to use a standard discrete approximation of the derivative such as

$$f''(x_t) \approx \frac{f'(x_t + h) - f'(x_t - h)}{2h}.$$

3.3.3 How can Newton's method go wrong?

Let's think about what can go wrong - namely when we could have $f(x_{t+1}) > f(x_t)$? Basically, if $f'(x_t)$ is relatively flat, we can get that $|x_{t+1} - x^*| > |x_t - x^*|$. We'll see an example on the board and the demo code . Newton's method can also go uphill when the second derivative is negative, with the method searching for a maximum.

```
options(width=55)
par(mfrow = c(1,2))
fp <- function(x, theta = 1){
        exp(x*theta)/(1+exp(x*theta)) - .5
}
fpp <- function(x, theta = 1){
        exp(x*theta)/((1+exp(x*theta))^2)
}
xs <- seq(-15, 15, len = 300)
plot(xs, fp(xs), type = 'l')
lines(xs, fpp(xs), lty = 2)
# good starting point
x0 <- 2
xvals <- c(x0,rep(NA,9))
for(t in 2:10){
        xvals[t]=xvals[t-1] - fp(xvals[t-1]) / fpp(xvals[t-1])
}
```