Notes for 2017-04-14

1 Taylor revisited

Though we have mentioned optimization problems intermittently through the last few lectures, we have focused mainly on the problem of finding the solutions of nonlinear equations. We now turn to the problem of finding (local) optima of functions with at least two continuous derivatives.

Recall the basic Taylor expansion that we outlined before the break; if $\phi : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^2$ (i.e. if it is at least twice continuously differentiable) then we have the expansion

$$\phi(x+u) = \phi(x) + \phi'(x)u + \frac{1}{2}u^T H_\phi(x)u + o(\|u\|^2)$$

where $\phi'(x) \in \mathbb{R}^{1\times n}$ is the derivative of ϕ and H_ϕ is the *Hessian matrix* consisting of second derivatives:

$$(H_\phi)_{ij} = \frac{\partial \phi}{\partial x_i \partial x_j}.$$

The *gradient* $\nabla\phi(x) = \phi'(x)^T$ is a column vector (rather than a row vector).

If $\nabla\phi(x) \neq 0$ then $\nabla\phi(x)$ and $-\nabla\phi(x)$ are the directions of steepest ascent and descent, respectively. If $\nabla\phi(x) = 0$, then we say x is a *stationary point* or *critical point*. The first derivative test says that if x minimizes ϕ (and ϕ is differentiable) then the gradient of x must be zero; otherwise, there is a "downhill" direction, and a point near x achieves a smaller function value.

A stationary point does not need to be a local minimizer; it might also be a maximizer, or a saddle point. The *second* derivative test says that for a critical point x to be a (local) minimizer, the Hessian $H_\phi(x)$ must be at least positive semi-definite at a (local) minimizer. If x is a stationary point and H_ϕ is strictly positive definite, then x must be a local minimizer; in this case, we call x a *strong* local minimizer.

One approach to the problem of minimizing ϕ is to run Newton iteration on the critical point equation $\nabla\phi(x) = 0$. The Jacobian of the function $\nabla\phi(x)$ is simply the Hessian matrix, so Newton's iteration for finding the critical point is just

$$x_{k+1} = x_k - H_\phi(x_k)^{-1}\nabla\phi(x_k).$$

We can derive this in the same way that we derived Newton's iteration for other nonlinear equations; or we can derive it from finding the critical point of a quadratic approximation to ϕ:

$$\hat{\phi}(x_k + p_k) = \phi(x_k) + \phi'(x_k)p_k + \frac{1}{2}p_k^T H_\phi(x_k)p_k.$$

The critical point occurs for $p_k = -H_\phi(x_k)^{-2}\nabla\phi(x_k)$; but this critical point is a strong local minimum iff $H_\phi(x_k)$ is positive definite.

There are a few reasons we might want to dig deeper:

- As with other systems of nonlinear equations, we might prefer to avoid a Newton iteration because of the cost of factoring the Jacobian (in this case, the Hessian matrix, which is the Jacobian of $\nabla\phi$).
- We can take advantage of the fact that this is *not* a general system of nonlinear equations in devising and analyzing methods.
- If we only seek to solve the critical point equation, we might end up finding a maximizer or saddle point as easily as a minimizer.

For this reason, we will discuss a different class of iterations, the (scaled) gradient descent methods and their relatives. At the end of the day, we will see many of the same ideas that we saw when treating nonlinear equations, but we will get to them by a slightly different path.

2 Gradient descent

One of the simplest optimization methods is the *steepest descent* or *gradient descent* method

$$x_{k+1} = x_k + \alpha_k p_k$$

where α_k is a *step size* and $p_k = -\nabla\phi(x_k)$. To understand the convergence of this method, consider gradient descent with a fixed step size α for the quadratic model problem

$$\phi(x) = \frac{1}{2}x^T Ax + b^T x + c$$

where A is symmetric positive definite. We have computed the gradient for a quadratic before:

$$\nabla\phi(x) = Ax + b,$$

which gives us the iteration equation

$$x_{k+1} = x_k - \alpha(Ax_k + b).$$

Subtracting the fixed point equation

$$x_* = x_* - \alpha(Ax_* + b)$$

yields the error iteration

$$e_{k+1} = (I - \alpha A)e_k.$$

If $\{\lambda_j\}$ are the eigenvalues of A, then the eigenvalues of $I - \alpha A$ are $\{1 - \alpha\lambda_j\}$. The spectral radius of the iteration matrix is thus

$$\min\{|1 - \alpha\lambda_j|\}_j = \min\left(|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\right).$$

The iteration converges provided $\alpha < 1/\lambda_{\max}$, and the optimal α is

$$\alpha_* = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

which leads to the spectral radius

$$1 - \frac{2\lambda_{\min}}{\lambda_{\min} + \lambda_{\max}} = 1 - \frac{2}{1 + \kappa(A)}$$

where $\kappa(A) = \lambda_{\max}/\lambda_{\min}$ is the condition number for the (symmetric positive definite) matrix A. If A is ill-conditioned, then, we are forced to take very small steps to guarantee convergence, and convergence may be heart breakingly slow. We will get to the minimum in the long run — but, then again, in the long run we all die.

The behavior of steepest descent iteration on a quadratic model problem is indicative of the behavior more generally: if x_* is a strong local minimizer of some general nonlinear ϕ, then gradient descent with sufficiently small step size will converge locally to x_*. But if $H_\phi(x_*)$ is ill-conditioned, then one has to take small steps, and the rate of convergence can be quite slow.

Not all problems are terrible ill-conditioned, and so in many cases simple gradient descent algorithms can work quite well. For ill-conditioned problems, though, we would like to change something about the algorithm. One approach is to keep the gradient descent direction and adapt the step size in a clever way; the Barzelei-Borwein (BB) method and related approaches follow this approach. These remarkable methods deserve to be better known, but in the interest of fitting the course into the semester, we will turn instead to the problem of choosing better directions.

3 Scaled gradient descent

The *scaled* gradient descent iteration takes the form

$$x_{k+1} = x_k + \alpha_k p_k, \quad M_k p_k = -\nabla\phi(x_k).$$

where α_k and p_k are the step size and direction, as before, and M_k is a symmetric positive definite *scaling matrix*. Positive definiteness of M_k guarantees that p_k is a *descent direction*, i.e.

$$\phi'(x_k)p_k = \nabla\phi(x_k)^T p_k = -\nabla\phi(x_k)^T M_k^{-1} \nabla\phi(x_k) < 0;$$

this in turn guarantees that if α_k is sufficiently small, $\phi(x_{k+1})$ will be less than $\phi(x_k)$ — unless $\phi(x_k)$ is a stationary point (i.e. $\nabla\phi(x_k) = 0$).

How does scaling improve on simple gradient descent? Consider again the quadratic model problem

$$\nabla\phi(x) = Ax + b,$$

and let M and α be fixed. With a little work, we derive the error iteration

$$e_{k+1} = (I - \alpha MA)e_k$$

If $\alpha M = A^{-1}$, the iteration converges in a single step! Going beyond the quadratic model problem, if $H_\phi(x_k)$ is positive definite, we might choose $M_k = H_\phi(x_k)$ — which would correspond to a Newton step.

Of course, $H_\phi(x_k)$ does not have to be positive definite everywhere! Thus, most minimization codes based on Newton scaling use $M_k = H_\phi(x_k)$ when it is positive definite, and otherwise use some modification. One possible modification is to choose a diagonal shift $M_k = H_\phi(x_k) + \beta I$ where β is sufficiently large to guarantee positive definiteness. Another common approach is to compute a *modified Cholesky* factorization of $H_\phi(x_k)$. The modified Cholesky algorithm looks like ordinary Cholesky, and is identical to ordinary Cholesky when $H_\phi(x_k)$ is positive definite. But rather than stopping when it encounters a negative diagonal in a Schur complement, the modified Cholesky approach replaces that element with something else and proceeds.

4 Modified and quasi-Newton optimizers

In the last two lectures, we described a variety of Newton-like methods for solving nonlinear equations that might involve fewer derivative computations

and lower cost less per step than Newton iteration. Most of these ideas carry over to optimization problems as well, but with some twists to ensure that steps always move in a descent direction. Some of the variants are:

- **Scaling with an approximate Hessian**: Here we choose M_k to be a symmetric positive definite matrix that in some sense approximates $H_\phi(x_k)$ (at least when the Hessian is positive definite) and for which it is easy to solve linear systems.

- **Inexact Newton steps**: Here we compute a (modified) Newton step, but inexactly (e.g. using a Krylov subspace method like PCG).

- **Quasi-Newton steps**: The most popular quasi-Newton method for optimization is the BFGS method (Broyden-Fletcher-Goldfarb-Shanno). The Broyden in BFGS is the same as the Broyden we saw in the popular Broyden quasi-Newton method for nonlinear equation solving, but the update formula itself is a little different. We still seek to satisfy a secant condition, but in BFGS we also seek to retain symmetry and positive definiteness of the approximate Hessian matrix. This is done with a rank two update.

  The limited memory BFGS (L-BFGS) method uses only a fixed set of previous iterates in the Hessian approximation, rather than considering the entire past convergence history. L-BFGS is one of the most popular methods for large scale optimization.

There are also "Krylov-like" methods for choosing update directions, most prominent of which are the nonlinear conjugate gradient methods. There are many potential nonlinear CG methods; for quadratic objective functions, they are all equivalent, but they differ when applied to more general functions. Newton-like methods may be more effective, but usually require more memory and computation per step. Anderson acceleration (discussed in the last lecture) has also been successfully applied to optimization methods.