Notes for 2016-04-13

Broadening the Basin

If $f(x^*) = 0$ is a solution to a nonlinear equation, the *basis of convergence* for x^* for a given iteration is the set of initial guesses for which the iteration converges to x^*. In the previous lecture, we gave some indication of features that can lead to small convergence basins for Newton-like iterations: we expect problems if f' is nearly singular in the vicinity of x^*, or if f' can change rapidly (i.e. it is not controlled by a modest Lipschitz constant). One way to deal with this problem is to find a very good initial guess; another approach, known as *globalization*, changes the iteration in simple ways in order to expand the basin of convergence. Globalization does not free us from getting a good initial guess, but it does make the quality of the initial guess slightly less crucial to guaranteeing convergence.

So far, we focused on nonlinear equation solving. To understand globalization, though, it will help to focus on optimization problems.

All Downhill from Here

Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be a C^1 function. At a point $x \in \mathbb{R}^n$, a vector $0 \neq p \in \mathbb{R}^k$ points in a *descent* direction if

$$\phi'(x)p < 0.$$

If we move from x by a small amount in a descent direction, we decrease the function value; that is, for small enough ϵ,

$$\phi(x + \epsilon p) = \phi(x) + \epsilon\phi'(x)p + o(\epsilon) < \phi(x).$$

The most familiar descent direction is the *steepest descent* direction $-\nabla\phi$. More generally, if A is a positive definite matrix, then $p = -A\nabla\phi(x)$ is a descent direction whenever x is not a stationary point, since

$$\phi'(x)p = -u^T Au, \quad u = \nabla\phi(x)$$

and positive definiteness guarantees $u^T Au > 0$. In particular, if x is near a strong local minimum and ϕ is C^2, then we expect the Hessian to be

positive definite, so that the Newton update $p = -H(x)^{-1}\nabla\phi(x)$ is a descent direction.

Of course, sometimes the Hessian could be indefinite or negative definite! In this case, even moving a little in the Newton direction could increase ϕ. We would usually consider this a Bad Thing. Therefore, in globalized optimization methods, we usually consider steps that are proportional to

$$p = -\hat{H}^{-1}\nabla\phi(x)$$

where $\hat{H}$ is some positive definite matrix. If $\hat{H} = I$, this gives us the steepest descent direction. For globalized Newton methods, we let $\hat{H}$ be equal to the Hessian matrix when the Hessian is sufficiently positive definite, and otherwise take $\hat{H}$ to be some positive definite matrix.

What if we are interested not in optimization, but in finding zeros of a nonlinear function $f : \mathbb{R}^n \to \mathbb{R}^n$? Recall that finding zeros of f is equivalent to finding minima of $\phi(x) = \|f(x)\|^2$, for which descent directions satisfy

$$\phi'(x)p = 2f(x)^T f'(x)p < 0.$$

If the Newton direction $p = -f'(x)^{-1}f(x)$ is well defined and nonzero, it satisfies

$$\phi'(x)p = -2\|f(x)\|^2,$$

and so the Newton step for f is certainly in a descent direction for ϕ.

Line Up!

If $p \in \mathbb{R}^n$ is a descent direction for ϕ, then we know

$$\phi(x + \epsilon p) = \phi(x) + \epsilon\phi'(x)p + o(\epsilon) < \phi(x)$$

for small enough ϵ. We can be even sharper and say that for any $\eta < 1$ and for small enough ϵ, we have

$$\phi(x + \epsilon p) \leq \phi(x) + \epsilon\eta\phi'(x)p.$$

Alas, unless we have more information (e.g. a Lipschitz constant for the derivative of ϕ), we cannot tell in advance how small is "small enough."

In a *line search* procedure, we consider first choose a descent direction p and then consider new points of the form $x + \sigma p$ for some scalar σ. A natural

choice is to consider *backtracking*: we look at points $x + \alpha^l p$ where $0 < \alpha < 1$ (a typical choice is $\alpha = 0.5$) and choose the smallest $l \geq 0$ such that

$$\phi(x + \alpha^l p) \leq \phi(x) + \eta \alpha^l \phi'(x) p.$$

An alternative to this geometric line search is to use *exact* line search: that is, find s to minimize $f(x + sp)$ along the ray. In practice, the cost of an exact line search is rarely worthwhile.

A successful *monotone* line search means that we will move from one guess at the minimizer to a new guess with a lower objective value. Moreover, if p is a descent direction, then we should always be able to conduct the line search successfully. There is, however, a caveat. Even putting aside the possibility that a programming error will lead us to propose a step in a non-descent direction (which we can and should guard against by careful programming), we might find that a proposed direction is simply not a very *good* descent direction, either because it is nearly orthogonal to the gradient or because the function is a little crazy. In either case, we might find ourselves cutting the step by what seems like an unreasonable amount. In the worst case, we might find that $|\alpha^l p| < |x|$ componentwise, so that $\mathrm{fl}(x + \alpha^l p) = \mathrm{fl}(x)$; in this case, we keep trying the same value over and over again, and end up in an infinite loop. We guard against this possibility in two ways. First, we will make sure that we bound the number of steps that we allow in a line search — a good idea for any iterative procedure! Second, we will try to rule out directions that form too acute an angle with the gradient, and thus are not "sufficiently downhill" for the algorithm to make acceptable progress.

There has also been some work on *non-monotone* line search algorithms that allow increases in the function values, as long as progress is made in some more averaged sense (e.g. the new point has a objective function value smaller than the maximum objective function for the past few points). This is useful for improving convergence speed on some hard problems, and is useful in the context of particular classes of methods such as spectral projected gradient (about which we will say nothing in this class other than the name).

Decent Descent

Choosing descent directions and employing line search is enough to strongly suggest convergence, but there are still some pathological ways to get into trouble. In particular:

- Our proposed steps p^k might blow up. To avoid this, we will usually insist on algorithms where $\|p^k\|$ remains bounded for all k.
- Our proposed steps p^k might shrink too quickly, so that we converge before actually reaching a minimizer. To avoid this, we insist on algorithms where $\|p^k\| \geq m\|\nabla\phi(x^k)\|$ for all k (such steps are called *gradient related*).
- Our proposed steps might remain a reasonable length, but come closer and closer to orthogonal to the gradient vector, so that progress slows too quickly for us to converge to a minimizer.

In general, if our objective is C^1 with a Lipschitz first derivative, and if we use the line search algorithm sketched above (backtracking line search where we pick the longest step that gives sufficient decrease) and a sequence of proposed steps p^k that are gradient related, form acute angles with the gradient that are bounded away from right angles, then we are guaranteed to converge to a stationary point whenever the set of points less than the initial value $\phi(x^0)$ is bounded. Note that this last condition is important! For instance, a function like

$$\phi(x) = \exp(-x^2/2) - exp(-x^4/2)$$

has two global minima close to zero, but for any starting guess outside the interval $[-3, 3]$ (or even a bit more), any iteration based on descent directions can only move *away* from those mimina. Also, we are only guaranteed to converge to a stationary point; it could be a saddle, or a local minimizer that is not the one we care about. For all of these reasons, good initial guesses remain important even with globalization.

Newton's method has many attractive properties, particularly when we combine it with a globalization strategy. Unfortunately, Newton steps are not cheap. At each step, we need to:

- Form the function f *and* the Jacobian. This involves not only computational work, but also analytical work – someone needs to figure out those derivatives!
- Solve a linear system with the Jacobian. This is no easier than any other linear solve problem! Indeed, it may be rather expensive for large systems, and factorization costs cannot (in general) be amortized across Newton steps.

The Jacobian (or the Hessian if we are looking at optimization problems) is the main source of difficulty. Today we consider several iterations that deal with this difficulty in one way or the other.

Trust but Verify

The line search paradigm for globalized optimization involves two phase:

- Propose a step in a descent direction (with some conditions mentioned above).
- Potentially cut the step back to get sufficient decrease.

But if we choose a step according to Newton's method, there is something odd about this two-stage process. Why should we choose a direction based on a model that we think will be good globally, then keep using that direction after we find out it was not a good direction? An alternative approach is to define a *trust region* where we believe the model to be good, and choose the next iterate to lie within the trust region. That is, at each step we minimize an approximation

$$f(x^k + p^k) \approx f(x^k) + f'(x^k)p^k + \frac{1}{2}(p^k)^T H(x^k)p^k$$

subject to the constraint that $\|p^k\| \leq \rho$. This leads to the subproblem

$$(H + \lambda I)p^k = -\nabla f(x^k)$$

where λ is a multiplier that enforces the constraint $\|p^k\| \leq \rho$. We then accept or reject the point, and dynamically adjust ρ, depending on how well the objective value at the new point agrees with the model.

Trust region methods are generally more complex to implement than line search methods, particularly since one often approximates the constrained minimization problem at the heart of the method. On the other hand, there are sometimes reasons to prefer trust region approaches.

Gauss-Newton and Levenberg-Marquardt

The trust region approach was first used in the context of nonlinear least squares problems, where the *Gauss-Newton* iteration involves

$$\min_{p^k} \|f(x^k) + f'(x^k)p^k\|.$$

The Gauss-Newton iteration is *not* the same as Newton, since the Hessian of $\phi(x) = \|f(x)\|^2/2$ is the matrix with elements

$$\phi_{,ij}(x) = \sum_k [f_{k,i}(x) f_{k,j}(x) + f_k(x) f_{k,ij}(x)],$$

or, equivalently,

$$H(x) \equiv \nabla^2 \phi(x) = f'(x)^T f'(x) + \sum_k f_k(x) \nabla^2 f_k(x).$$

The first term is the matrix that appears in the Gauss-Newton step, but Gauss-Newton lacks the second term. On the other hand, if the residual is small at the minimizer, the second term often contributes only a little.

The *Levenberg-Marquardt* iteration can be seen as a regularized version of the Gauss-Newton iteration, i.e.

$$\min_{p^k} \|f(x^k) + f'(x^)p^k\| + \lambda_k \|p^k\|.$$

In Levenberg-Marquardt, one typically works with λ_k directly, rather than treating it as a multiplier that enforces a constraint that $\|p^k\|$ lie in some bounded domain. However, it is possible to treat the method either from the perspective of trust regions or from the perspective of regularization.