Quasi-Newton Methods

MA 348

Kurt Bryan

Newton's Method Pros and Cons

Newton's method has some very nice properties: It's extremely fast, at least once it gets near the minimum, and with the simple modifications we made it becomes a descent method that seeks out minima instead of general critical points.

But Newton's method has some drawbacks, too. The most annoying is that we need all of the second partial derivatives, which may be difficult to compute. Also, standard solvers require $O(n^3)$ operations to solve $\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$, which could be time consuming. And of course, we also have to store the Hessian matrix, which requires $O(n^2)$ memory.

Conjugate gradient techniques overcame these problems, although there is a price: Newton's Method is quadratically convergent, but in general conjugate gradient methods do NOT converge quadratically.

We'll now look at yet another popular class of unconstrained nonlinear optimization methods, so-called *Quasi-Newton* methods. Like conjugate gradients, these methods are sequential "line search" algorithms which do a good job on quadratic objective functions (and hence, we hope, on smooth non-quadratic functions). They can be viewed as a variation of Newton's method that retain good convergence properties, but don't require difficult Hessian computations and linear solves.

Quasi-Newton Methods

Like the other algorithms we've seen, Quasi-Newton methods are iterative, involving a series of line searches, and generally involve computations only of f and ∇f at each iteration. In fact the only thing that distinguishes them (or any of the algorithms we've looked at) is how the search direction is chosen. At the kth stage of the algorithm we will compute a certain matrix $\mathbf{H}_k$; this computation will be fast and easy. The matrix $\mathbf{H}_k$ is supposed to be an approximation to $\mathbf{H}^{-1}(\mathbf{x}_k)$, the *inverse* of the Hessian of f at the current iterate, although some approaches approximate the Hessian itself. By approximating $\mathbf{H}^{-1}$ we don't need to solve the linear system $\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$, but instead simply multiply to find search direction $\mathbf{h}_k = -\mathbf{H}_k\nabla f(\mathbf{x}_k)$.

The basic form of the algorithm is much like Newton's method, and goes like this:

1. Make an initial guess $\mathbf{x}_0$ at the minimum; set $k = 0$. Initialize $\mathbf{H}_0 = \mathbf{I}$, the n by n identity matrix (or maybe some other appropriate positive definite matrix).

2. Compute $\nabla f(\mathbf{x}_k)$ and take the search direction as $\mathbf{h}_k = -\mathbf{H}_k\nabla f(\mathbf{x}_k)$ (in straight Newton's method we'd take $\mathbf{h}_k = \mathbf{H}^{-1}(\mathbf{x}_k)\nabla f(\mathbf{x}_k)$).

3. Do a line search from $\mathbf{x}_k$ in the direction $\mathbf{h}_k$ and take $\mathbf{x}_{k+1} = \mathbf{x}_k + t^*\mathbf{h}_k$, where t^* minimizes $f(\mathbf{x}_k + t\mathbf{h}_k)$. (In straight Newton we'd use $t^* = 1$ always, but we could also do a line search). Do some termination test to see if we're done.

4. Compute $\mathbf{H}_{k+1}$ by modifying $\mathbf{H}_k$ appropriately. This is usually done by setting $\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{U}_k$, where $\mathbf{U}_k$ is some easy to compute "updating" matrix. Set $k = k + 1$ and go to step 2.

Of course, the key step is the update from $\mathbf{H}_k$ to $\mathbf{H}_{k+1}$.

As mentioned above, the idea is that $\mathbf{H}_k$ should be an approximation to $\mathbf{H}^{-1}(\mathbf{x}_k)$. We want the matrices $\mathbf{H}_k$ to contain the "essential features" of $\mathbf{H}^{-1}(\mathbf{x}_k)$ without the expense of actually computing or inverting the Hessian. These essential features are described below.

- **Symmetry:** The Hessian matrix is symmetric, and hence so is the inverse (try proving this), so it seems reasonable to insist that $\mathbf{H}_k$ should be symmetric. If the update $\mathbf{U}_k$ is symmetric then $\mathbf{H}_{k+1}$ will inherit symmetry from $\mathbf{H}_k$.

- **Quasi-Newton Condition:** Consider for a moment the special case in which f is quadratic with constant Hessian $\mathbf{H}$, hence of the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{b} + c$. In this case $\nabla f(\mathbf{x}) = \mathbf{H}\mathbf{x} + \mathbf{b}$. It's easy to check that for any vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ we have

$$\mathbf{x} - \mathbf{y} = \mathbf{H}^{-1}(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})). \tag{1}$$

  We will require the matrix $\mathbf{H}_{k+1}$ that we construct to satisfy the condition (1) as if $\mathbf{H}_{k+1}$ were itself the inverse of the Hessian. However, we won't require equation (1) to hold for every possible pair of $\mathbf{x}$ and $\mathbf{y}$ (probably no matrix satisfies this if f is non-quadratic, and only the true inverse of $\mathbf{H}$ is f is quadratic). Instead, we'll require that $\mathbf{H}_{k+1}$ satisfy equation (1) in the case that $\mathbf{x} = \mathbf{x}_{i+1}$ and $\mathbf{y} = \mathbf{x}_i$ for $i = 0$ to $i = k$, that is,

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{H}_{k+1}(\nabla f(\mathbf{x}_{i+1}) - \nabla f(\mathbf{x}_i))$$

  for $0 \le i \le k$. This is the *quasi-Newton* condition.

  Note that at the time we want to compute $\mathbf{H}_{k+1}$ we already have in hand the iterates $\mathbf{x}_0, \ldots, \mathbf{x}_k$.

  In what follows let's define

$$\Delta\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i, \quad \Delta\mathbf{g}_i = \nabla f(\mathbf{x}_{i+1}) - \nabla f(\mathbf{x}_i).$$

  The quasi-Newton condition can then be written as

$$\Delta\mathbf{x}_i = \mathbf{H}_{k+1}\Delta\mathbf{g}_i \tag{2}$$

  for $0 \le i \le k$.

- **Positive-Definiteness:** It would be highly desirable if the matrices $\mathbf{H}_k$ were positive definite (which also guarantees that $\mathbf{H}_k^{-1}$ is positive definite). By the reasoning we've used before, $\mathbf{h}_k$ will then be a descent direction.

So that's the task: Construct $\mathbf{H}_k$ so that it's symmetric and satisfies the quasi-Newton condition; as it turns out, we'll get positive definiteness for free! We'd like to use only function and gradient evaluations at $\mathbf{x}_k$ (or maybe prior points too) to do this.

Quadratic Functions; An Example

If the function f in question is in fact quadratic in n variables with Hessian matrix $\mathbf{H}$ then it can be shown that any method for computing a symmetric $\mathbf{H}_k$ in accordance with the quasi-Newton condition (2) yields an algorithm which (with exact line searches) will minimize f in at most n line searches. In fact, the directions $\mathbf{h}_k$ generated by the algorithm will be $\mathbf{H}$-conjugate, so the quasi-Newton algorithm is also a conjugate direction algorithm. From Theorem 2 in the Conjugate Gradients handout this proves that n line searches suffice to locate the minimum. Since quasi-Newton methods perform well on quadratic functions, they should also perform well on more general functions.

Here's an example in the case $n = 2$. Let the function f be given by $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}+\mathbf{x}^T\mathbf{b}$ with

$$\mathbf{H} = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

We begin with initial guess $\mathbf{x}_0 = [2, 1]^T$ and $\mathbf{H}_0$ as the two by two identity. We then obtain search direction $\mathbf{h}_0 = -\mathbf{H}_0\nabla f(\mathbf{x}_0) = [-8, -6]^T$. A line search to minimize $f(\mathbf{x}_0 + t\mathbf{h}_0) = 180t^2 - 100t + 13$ in t yields minimum at $t = 5/18$, and the next iterate is $\mathbf{x}_1 = \mathbf{x}_0 + \frac{5}{18}\mathbf{h}_0 = [-2/9, -2/3]^T$.

Up to this point it's just steepest descent. Now we need to find an updated Hessian $\mathbf{H}_1$. It must be symmetric, so that

$$\mathbf{H}_1 = \begin{bmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{bmatrix}.$$

The Quasi-Newton condition also comes into play. We have

$$\Delta\mathbf{x}_0 = \mathbf{x}_1 - \mathbf{x}_0 = \begin{bmatrix} -20/9 \\ -5/3 \end{bmatrix}, \quad \Delta\mathbf{g}_0 = \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_0) = \begin{bmatrix} -25/3 \\ -50/9 \end{bmatrix}$$

The Quasi-Newton condition (note $k = 0$ still, so we only need to consider the case $i = 0$ in equation (2)) yields equations

$$-\frac{25}{3}h_{11} - \frac{50}{9}h_{12} = -\frac{20}{9}, \quad -\frac{25}{3}h_{12} - \frac{50}{9}h_{22} = -\frac{5}{3}.$$

This is two equations in three unknowns, so we ought to have a free variable. Indeed, we may take h_{11} arbitrarily. With $h_{11} = 1$ we find $h_{12} = -11/10$ and $h_{22} = 39/20$. We then have

$$\mathbf{H}_1 = \begin{bmatrix} 1 & -11/10 \\ -11/10 & 39/20 \end{bmatrix}.$$

The next iteration proceeds as follows: The new search direction is $\mathbf{h}_1 = -\mathbf{H}_1 \nabla f(\mathbf{x}_1)$, and we find that $\mathbf{h}_1 = [37/45, -37/30]^T$. Then $f(\mathbf{x}_1 + t\mathbf{h}_1) = \frac{1369}{900}t^2 - \frac{37}{45}t - \frac{8}{9}$ has minimum at $t = 10/37$. The next estimate of the minimum is $\mathbf{x}_2 = \mathbf{x}_1 + \frac{10}{37}\mathbf{h}_1 = [0, -1]^T$, which IS the exact minimum.

The example above illustrates the fact that the Quasi-Newton approach minimizes a quadratic in n variables using n (or fewer) line searches. It's also worth noting that although we chose h_{11} in $\mathbf{H}_1$ arbitrarily, it wouldn't have made any difference. We'd still get to the minimum on the next iteration.

Problems 1:

1. Run through the example above but make a different choice for h_{11}.

2. Modify the proof of Theorem 3 in the Conjugate Gradients handout to show that the directions generated by a quasi-Newton method (for a quadratic f with Hessian matrix $\mathbf{H}$) are in fact $\mathbf{H}$-conjugate.

Some Specific Quasi-Newton Methods and the Non-Quadratic Case

The Quasi-Newton condition looks like a pain to implement, for even in the quadratic case at the kth stage of the algorithm we obtain (possibly) nk equations which must be satisfied by the entries of $\mathbf{H}_{k+1}$. By the time $k \approx n$ solving these equations is about $O(n^3)$ work.

But in fact many Quasi-Newton strategies show how to construct $\mathbf{H}_{k+1}$ "painlessly", with no solves, so that the Quasi-Newton condition is satisfied. One of the most famous is the DFP (Davidson, Fletcher, and Powell) update, in which we take $\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{U}_k$ where

$$\mathbf{U}_k = \frac{\Delta\mathbf{x}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} - \frac{(\mathbf{H}_k \Delta\mathbf{g}_k)(\mathbf{H}_k \Delta\mathbf{g}_k)^T}{\Delta\mathbf{g}_k^T \mathbf{H}_k \Delta\mathbf{g}_k}. \tag{3}$$

Despite the complicated appearance of the formula, it's just mindless algebra and a bit of induction to show that in the quadratic case DFP yields a matrix $\mathbf{H}_{k+1}$ which satisfies the Quasi-Newton condition (2).

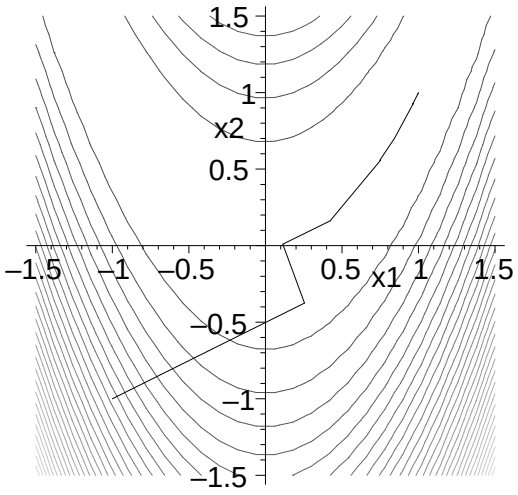
Problem 2: Show the DFP update satisfies the Quasi-Newton condition (2) in the special case $i = k$, that is, show

$$\Delta\mathbf{x}_k = \mathbf{H}_{k+1} \Delta\mathbf{g}_k$$

with $\mathbf{U}_k$ chosen as above and $\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{U}_k$. By the way, it doesn't matter whether or not f is quadratic here.

I won't give the induction proof that DFP yields matrices $\mathbf{H}_{k+1}$ that satisfy (2) for $0 \leq i \leq k$ in the quadratic case. See *An Introduction to Optimization* by Chong and Zak for the proof.

Note, however, that the DFP update (3) makes perfect sense for non-quadratic functions. We may thus apply it to such functions in the hope that (since such functions are locally quadratic) the method will converge like Newton's method. Below is an example to show the results of using the DFP algorithm on Rosenbrock's function, with initial guess $(-1, -1)$ and exact line searches (Golden section). Notice that by taking $\mathbf{H}_0 = \mathbf{I}$, the first step will be simply steepest descent. In this case the algorithm converges in 12 iterations.



Another formula, generally favored over DFP, is the BFGS (Broyden, Fletcher, Goldfarb, and Shanno) update, which is

$$\mathbf{U}_k = \left(1 + \frac{\Delta g_k^T \mathbf{H}_k \Delta g_k}{\Delta \mathbf{x}_k^T \Delta g_k}\right) \frac{\Delta \mathbf{x}_k \Delta \mathbf{x}_k^T}{\Delta \mathbf{x}_k^T \Delta g_k} - \frac{\mathbf{H}_k \Delta g_k \Delta \mathbf{x}_k^T + (\mathbf{H}_k \Delta g_k \Delta \mathbf{x}_k^T)^T}{\Delta \mathbf{x}_k^T \Delta g_k}. \tag{4}$$

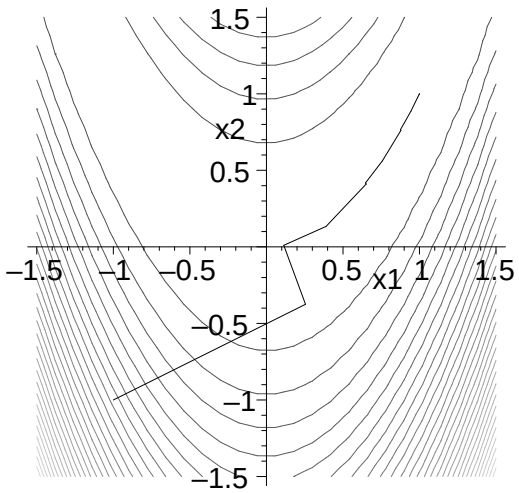
Again, it's just mind-numbing algebra to verify that this satisfies the quasi-Newton condition in the quadratic case.

Problems 3:

1. Verify that the BFGS formula satisfies the quasi-Newton condition in the case $i = k$.
2. Show that $\mathbf{U}_k$ is symmetric for both the DFP and BFGS methods, so that $\mathbf{H}_{k+1}$ is symmetric if $\mathbf{H}_k$ is.

It can be shown for both DFP and BFGS that if the line searches are done exactly then $\mathbf{H}_{k+1}$ is positive definite if $\mathbf{H}_k$ is, so the search directions will always be descent directions. Again, see Chong and Zak for a proof.

Here is an example showing the BFGS algorithm under the same conditions:



It converges in 14 iterations.

More Stuff

The BFGS update is the most favored approach to quasi-Newton methods. It has some subtly better properties as far as round-off is concerned and is better behaved if the line searches are not exact.

Notice that the quasi-Newton methods as presented above DON'T get rid of the $O(n^2)$ storage problem for the Hessian, but do get rid of the $O(n^3)$ Cholesky factorization issue.

However, as mentioned above, many approaches to quasi-Newton methods approximate $\mathbf{H}(\mathbf{x}_k)$ directly, instead of the inverse. As a result we still have to solve an n by n linear system $\mathbf{H}_k\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$ at every iteration; it seems like such an approach gives away one of the main advantages of quasi-Newton methods. However, because the matrix $\mathbf{H}_{k+1}$ is constructed in a simple way from a prior matrix $\mathbf{H}_k$, and assuming we have done a Cholesky factorization $\mathbf{H}_k = \mathbf{L}_k^T\mathbf{D}_k\mathbf{L}_k$, we can relatively easily (in $O(n^2)$ operations) use this to compute a Cholesky factorization of $\mathbf{H}_{k+1}$, and so use previous work to more easily solve $\mathbf{H}_k\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$. We won't discuss such issues here.