Quasi-Newton optimization: Origin of the BFGS update

Steven G. Johnson, notes for 18.335 at MIT

April 25, 2019

Abstract

In a typical optimization setting we are provided with an objective function $f(x)$ and its gradient ∇f only. However, as these are evaluated for many different points x we can infer something about the second derivative ("Hessian") by watching how ∇f changes, and by incorporating that information into our optimization algorithm we can accelerate convergence. This approach leads to "quasi-Newton" or "variable-metric" methods, so-called because they approximate an exact Newton step for $\nabla f = 0$. The most widely used method to approximate the Hessian is a BFGS update, and in these notes we survey the basic ideas underlying this important algorithm.

1 Newton steps and backtracking

Suppose that we are trying to solve

$$\min_{x \in \mathbb{R}^n} f(x)$$

and we are supplied a method to efficiently compute both $f(x)$ and ∇f (e.g. by an adjoint method).

On step k of optimization, let x^k be our current iterate, and let $g^k = \nabla f|_{x^k}$. If we had the **second derivative "Hessian" matrix** H^k as well ($H^k_{ij} = \frac{\partial f}{\partial x_i \partial x_j}\Big|_{x^k}$), then we could try to make progress via second-order (quadratic) Taylor expansion

$$f(x^k + \delta) \approx f(x^k) + \delta^T g^k + \frac{1}{2}\delta^T H^k \delta = q(\delta).$$

Near a local minimum, H is positive-definite, and the minimum of $q(\delta)$ is

$$\delta^k = -(H^k)^{-1} g^k.$$

In fact, this is exactly a **Newton step** in finding a root of $\nabla f = 0$, where we approximate the gradient near x by a *first-order* Taylor expansion

$$\nabla f|_{x+\delta} \approx g^k + H^k \delta.$$

However, we might run into a problem: the Newton step δ might be so large that our Taylor expansion is not accurate, and $f(x^k + \delta)$ might actually get *worse*. There are a couple of common approaches to fix this:

1. **Trust region**: minimize a $q(\delta)$ only for δ sufficiently small, i.e. in a "trust region." For example, a common choice is a spherical trust region $\|\delta\|_2 < r^k$ for some radius r^k, in which case there is a nice result: strong duality holds, and we can optimize a convex dual problem.[1] If the resulting step is not "acceptable" (see below), we can change the trust-region radius and try again.

2. **Line search**: We can minimize $f(x^k + \alpha\delta^k)$ over $\alpha \in \mathbb{R}$, i.e. along the direction of the Newton step δ^k. Usually minimizing this *exactly* is more trouble than it is worth just to take a single optimization step, so it is common to do an **inexact line search**: try different α until the result is "acceptable" (see below).

3. **Backtracking**: Instead of exact line search, we try $f(x^k + \alpha\delta^k)$ for $\alpha = 1, \tau, \tau^2, \tau^3, \ldots$ where $0 < \tau < 1$ is some parameter (e.g. $\tau = 0.5$), until the result is "acceptable" (see below).

For both a trust region and backtracking line search we have to decide whether a given step δ is acceptable. Naively, we can simply check whether $f(x^k + \delta) < f(x^k)$, but in practice it turns out that we want to impose stronger conditions — we only want to take steps δ where our quadratic approximation $q(\delta)$ is reasonably accurate. In practice, we typically impose one or both of the **Wolfe conditions** on the step δ:

1. $f(x^k + \delta) \le f(x^k) + c_1\delta^T g^k$ where $0 < c_1 < 1$, typically $c_1 = 10^{-4}$: f must decrease at least proportional to the prediction of the gradient g^k.

2. $|\delta^T g^{k+1}| \le c_2|\delta^T g^k|$, where $g^{k+1} = \nabla f|_{x^k+\delta}$ and $0 < c_2 < 1$, e.g. $c_2 = 0.9$: the derivative $\delta^T \nabla f$ along the search direction must decrease sufficiently. (Note that for an exact line search we will have $g^{k+1} = 0$.) This condition helps prevent trust-region or inexact line-search methods from taking steps δ that are too small, and it also leads to a nice property of BFGS updates below.

2 Quasi-Newton/Variable-metric methods

The problem with Newton steps is that the exact Hessian is hard to come by when n is large. Even with adjoint methods, evaluating H exactly typically costs at least n times the cost of evaluating f once (it correspond to taking the gradient n times: one more gradient for each component of ∇f). When n is really large, just *storing* the H matrix (n^2 numbers) might be impractical.

[1] This is called the "trust region problem," and is discussed in e.g. Boyd & Vandenberghe section 5.2.

Instead, "quasi" Newton methods (also called "variable-metric" methods) apply the same Newton steps above but use an **approximate Hessian** H^k, often a low-rank approximation (which can be stored and applied efficiently). In fact, since what is needed for the Newton step is $(H^k)^{-1}$, usually one stores a low-rank **approximate inverse Hessian**. To obtain this, we want to **iteratively** construct our approximate H^{k+1} (or $(H^{k+1})^{-1}$) **given only the gradient** (*first* derivative) of f. Some desired properties of H^k are:

1. For a convex quadratic $f(x)$, H^k should approach the exact Hessian as $k \to \infty$ (i.e., as we apply our iterative update for many points and many gradient evaluations, approaching the minimum). In practice, what can typically be proved [6, 5, 3] is that for a convex quadratic $f(x)$, the quasi-Newton method gives the exact minimum and the exact Hessian in n steps (in exact arithmetic).

2. **Secant condition**:

$$\underbrace{g^{k+1} - g^k}_{\gamma} = H^{k+1} \underbrace{(x^{k+1} - x^k)}_{\delta}.$$

   This condition arises because it would be true of the *exact* Hessian for a quadratic f (see the $\nabla f|_{x+\delta}$ Taylor expansion above). Equivalently, H^k must at least predict the change in the gradient on the k-th step.

3. Real-symmetric positive-definite. This makes our $q(n)$ function convex and $\delta^k = -(H^k)^{-1} g^k$ is in the "downhill" direction from x^k.

4. H^k should "remember" as much information from previous steps (i.e. the previous gradient evaluations) as possible. (We *don't* want to impose the secant conditions on all steps simultaneously, however, because this could quickly become impossible: f may not be exactly quadratic.)

The last criterion is rather vague and could lead to many possible quasi-Newton algorithms. However, it turns out that an extremely easy and powerful approach to "remembing" information is to simply **minimize the change in** H^k: we minimize $\|H^{k+1} - H^k\|$ in some norm, or alternatively minimize $\|(H^{k+1})^{-1} - (H^k)^{-1}\|$. In the appropriate choice of norm, the latter leads to the famous "BFGS" update, which has lots of nice properties.

3 BFGS updates

This update, named for **B**royden [1], **F**letcher [2], **G**oldfarb [3], and **S**hanno [4], who wrote four *separate* papers that developed the approach in 1970, is obtained by solving

$$\min_{H \in \mathbb{R}^{n \times n}} \|H^{-1} - (H^k)^{-1}\|_W$$

$$\text{subject to } H^{-1}\gamma = \delta \text{ and } H^T = H$$

That is, we minimize the change in H^{-1} subject to the second condition and require that it be real-symmetric (it will turn out that we get positive-definiteness "for free" below). Here, $\|\cdots\|_W$ is a weighted Frobenius norm

$$\|A\|_W^2 = \frac{1}{2}\operatorname{tr}\left[WAWA^T\right] = \frac{1}{2}\|MAM^T\|_F^2 = \frac{1}{2}\operatorname{tr}\left[MAM^TMA^TM^T\right]$$

where $W = M^TM$ is a positive-definite "weight" matrix to be chosen later (recall the identity $\operatorname{tr} AB = \operatorname{tr} BA$). If we let $E = H^{-1} - (H^k)^{-1}$, require that the previous iterate H^k be symmetric, then this optimization problem equivalently becomes

$$\min_{E\in\mathbb{R}^{n\times n}} \|E\|_W^2$$
$$\text{subject to } Ey = r \text{ and } E^T = E$$

where $y = \gamma$ and $r = \delta - (H^k)^{-1}\gamma$.[2] This optimization problem is, in fact, a **QP**: we are minimizing a convex quadratic objective subject to affine constraints. In consequence, strong duality holds and we can instead solve the Lagrange dual problem. Equivalently, we can solve the KKT conditions. It turns out that this leads to a very nice formula for the update E if we make the right choice of weight matrix W.

Let's apply duality, following Greenstadt [7] and Goldfarb [3]. We define Lagrange multipliers $\lambda \in \mathbb{R}^n$ for the $Ey - r = 0$ constraint and $\Gamma^T \in \mathbb{R}^{n\times n}$ for the $E - E^T = 0$ constraint, and obtain the Lagrangian

$$L(E, \lambda, \Gamma) = \operatorname{tr}\left[\frac{1}{2}WEWE^T + (Ey - r)\lambda^T + \Gamma(E - E^T)\right].$$

Here, note that $\operatorname{tr}\left[(Ey-r)\lambda^T\right] = \operatorname{tr}\left[\lambda^T(Ey-r)\right] = \lambda^T(Ey-r)$ is just the ordinary sum of n Lagrange multipliers λ_i times n constraints, but by re-ordering it into a rank-1 matrix we were able to combine it with the $\|E\|_W^2$ trace. And $\operatorname{tr}\left[\Gamma(E-E^T)\right] = \sum_{i,j}\Gamma_{ji}(E-E^T)_{ji} = \sum_{i,j}(\Gamma^T)_{ij}(E-E^T)_{ij}$ is a "Frobenius inner product" of the n^2 Lagrange multipliers $(\Gamma^T)_{ij}$ with the n^2 constraints from $E^T = E$. Note that

$$\nabla_B \operatorname{tr}(BC) = \nabla_B \sum_{ij} B_{ij}C_{ji} = C^T,$$

where ∇_B denotes the matrix of partial derivatives $\frac{\partial \operatorname{tr}(BC)}{\partial B_{ij}} = C_{ji}$, and similarly $\nabla_B \operatorname{tr}(B^TC) = C$. We can now solve the KKT conditions

$$\begin{aligned}\nabla_E L = 0 &= WEW + \lambda y^T + \Gamma^T - \Gamma,\\ Ey - r &= 0,\\ E^T - E &= 0.\end{aligned}$$

[2] If alternatively we were minimizing $\|H - H^k\|_W$, we would get exactly the same form of minimization problem with $E = H - H^k$, $y = \delta$, and $r = \gamma - H^k\delta$. This leads to an alternative quasi-Newton method, called the Davidon–Fletcher–Powell (DFP) method, that seems not to perform quite as well in practice. Intuitively, since H^{-1} is the quantity that appears in the Newton step, it is not too surprising that it is better to minimize the change in H^{-1} rather than the change in H.

subject to the constraints $Ey = r$ and $E^T = E$. The first equation gives

$$E = -W^{-1}\left(\lambda y^T + \Gamma^T - \Gamma\right)W^{-1}.$$

The requirement that $E = E^T$ then means that $\left(\lambda y^T + \Gamma^T - \Gamma\right) = \left(\lambda y^T + \Gamma^T - \Gamma\right)^T$, or equivalently

$$\Gamma^T - \Gamma = \frac{1}{2}\left(y\lambda^T - \lambda y^T\right)$$

and hence

$$E = -\frac{1}{2}W^{-1}\left(y\lambda^T + \lambda y^T\right)W^{-1}.$$

Finally, the condition $Ey = r$ now implies

$$y\lambda^T W^{-1}y + \lambda\left(y^T W^{-1}y\right) = -2Wr.$$

Since the $(\cdots)$ term is a scalar, we can solve for

$$\lambda = -\frac{2Wr + y\lambda^T W^{-1}y}{y^T W^{-1}y}.$$

At first glance, this doesn't seem immediately helpful since there is a λ^T on the right hand side. But if we multiply both sides by $y^T W^{-1}$ and transpose, we can solve for the unknown scalar $\lambda^T W^{-1}y$:

$$\lambda^T W^{-1}y = -\frac{2r^T y + y^T W^{-1}y\left(\lambda^T W^{-1}y\right)}{y^T W^{-1}y} \implies \lambda^T W^{-1}y = -\frac{r^T y}{y^T W^{-1}y}.$$

Plugging this back into $\lambda = \cdots$, we get

$$\lambda = -\frac{2Wr + -\frac{yr^T y}{y^T W^{-1}y}}{y^T W^{-1}y} = \frac{yy^T r}{\left(y^T W^{-1}y\right)^2} - \frac{2Wr}{y^T W^{-1}y}.$$

Finally, we can substitute this into our E equation to obtain

$$E = \frac{1}{y^T W^{-1}y}\left[ry^T W^{-1} + W^{-1}yr^T - \frac{y^T r}{y^T W^{-1}y}W^{-1}yy^T W^{-1}\right].$$

This looks messy, but it is actually quite nice: a **sum of rank-1 updates** to the inverse Hessian! But we have one trick left up our sleeve: we haven't chosen our weight W yet! Different choices of W lead to different quasi-Newton methods, but it is useful to note that E only involves W via the combination $W^{-1}y$.

To get an E that turns out to have the especially nice property of *preserving positive-definiteness* (if H^k is definite then H^{k+1} is also, as we discuss below), is to choose some W so that $W^{-1}y = \delta$. For example, we can choose $W^{-1} = (H^{k+1})^{-1} = E + (H^k)^{-1}$.[3] We then obtain, after a bit more algebra, the famous

[3]This may seem a bit circular: we choose W based on the *result* of the optimization. One way to think of it is that if you choose W based on the $E = \cdots$ formula, then hold W fix and minimize $\|E\|_W$ in our QP, you recover the same W.

BFGS update:

$$\boxed{(H^{k+1})^{-1} = (H^k)^{-1} - \frac{1}{\gamma^T\delta}\left[(H^k)^{-1}\gamma\delta^T + \delta\gamma^T(H^k)^{-1} - \left(1 + \frac{\gamma^T(H^k)^{-1}\gamma}{\gamma^T\delta}\right)\delta\delta^T\right]}.$$

This may look a little messy. Equivalently, via the Sherman–Morrison formula,[4] we can derive (after a bunch more algebra) the corresponding update of H^k:

$$\boxed{H^{k+1} = H^k + \frac{\gamma\gamma^T}{\gamma^T\delta} - \frac{H^k\delta\delta^T H^k}{\delta^T H^k \delta}},$$

which is easier to analyze, even though in practice it is H^{-1} that we compute and store.

3.1 Positive-definiteness

A key property of the choice of weight W in the BFGS update is that it allows us to ensure positive-definiteness of H^{k+1} assuming H^k is definite. (Typically the algorithm starts with $H^0 = I$ or a similar diagonal positive-definite matrix.) We simply need to check that $x^T H^{k+1} x > 0$ for any $x \neq 0$:

$$\begin{aligned} x^T H^{k+1} x &= x^T H^k x - \frac{(\delta^T H^k x)^2}{\delta^T H^k \delta} + \frac{(x^T\gamma)^2}{\gamma^T\delta} \\ &= \underbrace{\frac{(x^T H^k x)(\delta^T H^k \delta) - (\delta^T H^k x)^2}{\delta^T H^k \delta}}_{\geq 0 \text{ by Cauchy–Schwarz}} + \underbrace{\frac{(x^T\gamma)^2}{\gamma^T\delta}}_{>0 \text{ if } \gamma^T\delta > 0}. \end{aligned}$$

The first term is ≥ 0 by the Cauchy-Schwarz inequality: for any inner product $\langle x, y\rangle$, it is always true that $\langle x, x\rangle\langle\delta, \delta\rangle \geq |\langle x, y\rangle|^2$, and in this case because H^k is positive-definite we have an inner product $\langle x, y\rangle = x^T H^k y$.

The second term is clearly > 0 whenever $\gamma^T\delta = \delta^T\gamma = \delta^T g^{k+1} - \delta^T g^k > 0$, but why should this be? If we did an *exact* line search, then $\delta^T g^{k+1} = 0$, and $-\delta^T g^k = (x^{k+1} - x^k)^T(-g^k) > 0$ because $-g^k$ is the "downhill" direction and x^{k+1} must be "downhill" from x^k. If we did an *inexact* line search, but we imposed the second Wolfe condition $|\delta^T g^{k+1}| < |\delta^T g^k|$, then we still have $\delta^T g^{k+1} - \delta^T g^k > 0$ (the second term is positive and the first term can't be a larger negative magnitude). If we didn't impose the second Wolfe condition and happened to do a step where $\delta^T\gamma \lesssim 0$, then we can just skip the update: let $H^{k+1} = H^k$: violating the second Wolfe condition generally means that we took too small a step, and we want to keep going in the same direction.

[4] The Sherman–Morrison formula $(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1+v^T A^{-1}u}$ shows that a rank-1 update of A corresponds to a rank-1 update of A^{-1} and vice-versa.

4 Low-storage quasi-Newton (L-BFGS)

Applying the BFGS update directly requires $\Theta(n^2)$ storage for $(H^k)^{-1}$ and $\Theta(n^2)$ work on each step to update H^{k+1}. This is fine for n up to a few thousand, but for truly large-scale optimization problems it is prohibitive. Fortunately, the fact that BFGS is made of rank-1 updates (adding rank-1 matrices uv^T to H^k or its inverse), there is a solution: store a set of rank-1 updates, not the matrix. That is, represent

$$(H^k)^{-1} \approx H^0 + \sum_{j=1}^{m} u^j (v^j)^T,$$

where we keep the m most recent rank-1 updates for some m (typically $10 \lesssim m \leq 100$). This is known as an **L-BFGS** method, where "L" stands for "low-storage", and was introduced by Nocedal in 1980 [8].

With this representation of $(H^k)^{-1}$, assuming H^0 is sparse (typically diagonal, e.g. I), the storage cost is $\Theta(mn)$ for the $\{u^j, v^j\}$ vectors, the cost to multiply $(H^k)^{-1}g^k$ for the quasi-Newton step is also $\Theta(mn)$, where as usual we compute $uv^T g$ by $u(v^T g)$ in $\Theta(n)$ operations, and the cost of updating to H^{k+1} is $\Theta(mn)$ for the $(H^k)^{-1}\gamma$ product plus $\Theta(n)$ other operations.

Although for $m \ll n$ this procedure can no longer converge to the exact Hessian, in practice L-BFGS can greatly accelerate optimization (compared to steepest-descent and other first-order methods with "no memory") in many cases, especially optimization to high accuracy, in much the same way as an approximate Krylov method like restarted GMRES or nonlinear conjugate-gradient.

5 BFGS and constrained optimization

For nonlinearly constrained optimization ($\min f_0(x)$ subject to $f_i(x) \leq 0$), the most common utilization of BFGS has been for sequential quadratic programming (SQP): approximate the optimization problem by a sequence of convex QP (convex quadratic objective + affine constraints), typically solved in a trust region to give each optimization step. BFGS is then used to obtain the quadratic term in the QP, but there are a variety of ways to do this. The simplest is to apply BFGS to f_0 , but in that case only linear approximations are used for the constraints f_i. Alternatively, BFGS can be applied to some form of Lagrangian or "augmented" Lagrangian (= Lagrangian + penalties for violated constraints) [9].

References

[1] C. Broyden, "The convergence of a class of double-rank minimization algorithms," *J. Inst. Math. Appl.* **6**, pp. 76–90 (1970).

[2] R. Fletcher, "A new approach to variable-metric algorithms," *Computer J.* **13**, pp. 317–322 (1970).

[3] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Math. Comp.* **24**, pp. 23–26 (1970).

[4] D. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comp.* **24**, pp. 647–656 (1970).

[5] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Comput. J.* **6**, pp. 163–168 (1963).

[6] C. G. Broyden, "Quasi-Newton methods and their application to function minimisation," *Math. Comp.* **21**, pp. 368–381 (1967).

[7] J. Greenstadt, "Variations on variable metric methods," *Math. Comp.* **24**, pp. 1–22 (1970).

[8] R. H. Byrd, J. Nocedal, R. B. Schnabel, "Representations of quasi-Newton matrices and their use in limited memory methods," *Math. Prog.* **63**, pp. 129–156 (1994).

[9] R. H. Byrd, R. A. Tapia, Y. Zhang, "An SQP augmented Lagrangian BFGS algorithm for constrained optimization," *SIAM J. Optim.* **2**, pp. 210–241 (1992).

18.335J Introduction to Numerical Methods
Spring 2019