

Newton's Method

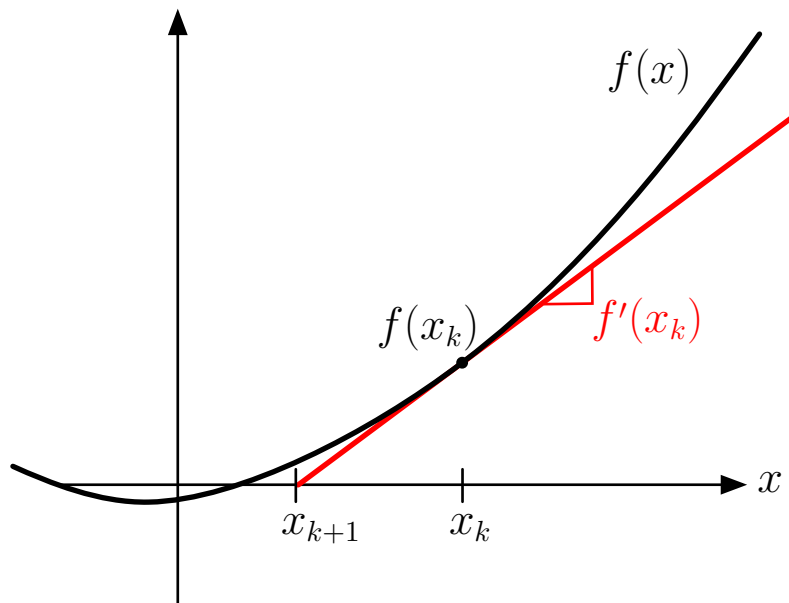
Newton's method is a classical technique for finding the root of a general differentiable function $f(x) : \mathbb{R} \rightarrow \mathbb{R}$. That is, we want to find an $x \in \mathbb{R}$ such that

$$f(x) = 0.$$

As you probably learned in high school, one technique for doing this is to start at some guess x_0 , and then follow the iteration

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

This update results from taking a simple linear approximation at each step:



Of course, there can be many roots, and which one we converge to will depend on what we choose for x_0 . It is also very much possible that the iterations do not converge for certain (or even almost all) initial values x_0 .

However there is a classical convergence theory that says that once we are close enough to a particular root x_0 , we will have

$$\underbrace{|x_0 - x_{k+1}|}_{\epsilon_{k+1}} \leq C \cdot \underbrace{(x_0 - x_k)^2}_{\epsilon_k^2},$$

where the constant C depends on the ratio between the first and second derivatives in an interval¹ around the root x_0 . The take-away here is that close to the solution, Newton's method exhibits *quadratic convergence*: the error at the next iteration is proportional to the square of the error at the last iteration. Since we are concerned with ϵ_k small, $\epsilon_k \ll 1$, this means that under the right conditions, the error goes down in dramatic fashion from iteration to iteration.

Notice that applying the technique requires that f is differentiable, but the convergence guarantee depends on f be twice (continuously) differentiable.

When $f(x)$ is convex, twice differentiable, and has a minimizer, we can find a minimizer by applying Newton's method to the derivative. We start at some initial guess x_0 , and then take

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \quad (1)$$

In this case if f is three-times continuously differentiable, we converge to the global minimizer quadratically. Again, applying the method

¹There are various technical conditions that f must obey for this result to hold, including the second derivative being continuous and the first derivative not being equal to zero. Also, the condition "close enough" is characterized by looking at ratios of derivatives at the root and an interval around it. The Wikipedia article on this is not bad: https://en.wikipedia.org/wiki/Newton's_method.

relies on us being able to compute first and second derivatives of f , and the analysis relies on f being three-times differentiable.

We can interpret the iteration (1) above in the following way:

1. At x_k , approximate $f(x)$ using the Taylor expansion

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2.$$

2. Find the exact minimizer of this quadratic approximation. Taking the derivative of the expansion above and setting it equal to zero yields the following optimality condition for \hat{x} to be a minimizer:

$$(\hat{x} - x_k)f''(x_k) = -f'(x_k).$$

This is just a re-arrangement of the iteration (1).

3. Take $x_{k+1} = \hat{x}$.

This last interpretation extends naturally to the case where $f(\mathbf{x})$ is a function of many variables, $f: \mathbb{R}^N \rightarrow \mathbb{R}$. We know that if f is convex and twice differentiable, we have a minimizer \mathbf{x}^* when $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Newton's method to find such a minimizer proceeds as above. We start with an initial guess \mathbf{x}_0 , and use the following iteration:

1. Take a Taylor approximation around $f(\mathbf{x}_k)$:

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \langle \mathbf{x} - \mathbf{x}_k, \mathbf{g} \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \mathbf{H}(\mathbf{x} - \mathbf{x}_k)$$

where

$$\begin{aligned} \mathbf{g} &= \nabla f(\mathbf{x}_k) = N \times 1 \text{ gradient vector at } \mathbf{x}_k \\ \mathbf{H} &= \nabla^2 f(\mathbf{x}_k) = N \times N \text{ Hessian matrix at } \mathbf{x}_k. \end{aligned}$$

2. Find the exact minimizer $\hat{\mathbf{x}}$ to this approximation. This gives us the problem

$$\underset{\mathbf{x} \in \mathbb{R}^N}{\text{minimize}} \quad \mathbf{g}^\top(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \mathbf{H}(\mathbf{x} - \mathbf{x}_k).$$

Since \mathbf{H} is positive semidefinite (since we are assuming f is convex), we know that the conditions for $\hat{\mathbf{x}}$ being a minimizer² are

$$\mathbf{H}(\mathbf{x} - \mathbf{x}_k) = -\mathbf{g}.$$

If \mathbf{H} is invertible (i.e., \mathbf{H} is positive definite), then we have a unique minimizer and

$$\hat{\mathbf{x}} = \mathbf{x}_k - \mathbf{H}^{-1}\mathbf{g}.$$

3. Take $\mathbf{x}_{k+1} = \hat{\mathbf{x}}$.

This procedure is often referred to as a *pure Newton step*, as it does not involve the selection of a step size. In practice, however, it is often beneficial to choose the step direction as

$$\mathbf{d}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k),$$

and then choose a step size α_k using a backtracking line search, and then take

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

as before.

²Take the gradient of this new expression and set it equal to $\mathbf{0}$.

Convergence of Newton's Method

Suppose that $f(\mathbf{x})$ is twice-differentiable, M -smooth, and strongly convex, i.e., the eigenvalues of the Hessian $\nabla^2 f(\mathbf{x})$ all lie within the interval $[m, M]$. Moreover, assume that the Hessian is Lipschitz:

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|_2.$$

(The norm on the left-hand side above is the standard operator norm.) We will show that the Newton algorithm coupled with an exact line search³ provides an estimate \mathbf{x}_k that is within ϵ of the global minimizer \mathbf{x}^* , i.e.,

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \epsilon,$$

provided that the number of iterations satisfies

$$k \geq C_1 (f(\mathbf{x}_0) - f(\mathbf{x}^*)) + \log_2 \log_2(\epsilon_0/\epsilon),$$

where we can take the constants above to be $C_1 = 2M^2L^2/m^5$ and $\epsilon_0 = 2m^3/L^2$. Qualitatively, this says that Newton's method takes a constant number of iterations to converge to any reasonable precision — we can bound $\log_2 \log_2(\epsilon_0/\epsilon) \leq 6$ (say) for ridiculously small values of ϵ .

To establish this result, we break the analysis into two stages. In the first, the *damped Newton stage*, we are far from the solution (as measured by $\|\nabla f(\mathbf{x}_k)\|_2$), but we make constant progress towards the answer. Specifically, we will show that in this stage,

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq 1/C_1.$$

³These results are easily extended to backtracking line searches; we are just using an exact line search to make the exposition easier. See [BV04, Sec. 9.5.3] for the analysis with backtracking.

This implies that when we are far from the solution, we reduce the gap $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ by at least $1/C_1$ at each iteration. It should be clear, then, that the number of damped Newton steps is no greater than $C_1 (f(\mathbf{x}_0) - f(\mathbf{x}^*))$.

We will then show that when $\|\nabla f(\mathbf{x}_k)\|_2$ is small enough, the gap closes dramatically at every iteration. We call this the *quadratic convergence stage*, as we will be able to show that once the algorithm enters this stage at iteration ℓ , for all $k > \ell$,

$$\|\nabla f(\mathbf{x}_k)\|_2 \leq C_2 \cdot 2^{-2^{k-\ell}},$$

where $C_2 = L/(2m^2)$ is another constant.

Damped phase

We are in this stage when

$$\|\nabla f(\mathbf{x}_k)\|_2 \geq m^2/L.$$

We take $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_{\text{exact}} \mathbf{d}_{k+1}$, where

$$\mathbf{d}_{k+1} = -\nabla^2 f(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k),$$

and α_{exact} is the result of an exact line search⁴:

$$\alpha_{\text{exact}} = \arg \min_{0 \leq \alpha \leq 1} f(\mathbf{x}_k + \alpha \mathbf{d}_{k+1}).$$

We define the current *Newton decrement* as

$$\lambda_k = \sqrt{\nabla f(\mathbf{x}_k)^T (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)},$$

⁴For convenience, we are not letting α be larger than 1, just as in a back-tracking method.

and note that $\lambda_k^2 = -\nabla f(\mathbf{x}_k)^\top \mathbf{d}_{k+1}$. Moreover, strong convexity implies that the eigenvalues of $(\nabla^2 f(\mathbf{x}_k))^{-1}$ are at least $1/M$ and at most $1/m$, yielding the bounds

$$\|\mathbf{d}_{k+1}\|_2^2 \leq \frac{1}{m} \lambda_k^2 \quad \text{and} \quad \frac{1}{M} \|\nabla f(\mathbf{x}_k)\|_2^2 \leq \lambda_k^2,$$

which we will use below. From the definition of strong convexity, we know that for any t we have

$$\begin{aligned} f(\mathbf{x}_k + t\mathbf{d}_{k+1}) &\leq f(\mathbf{x}_k) + \langle t\mathbf{d}_{k+1}, \nabla f(\mathbf{x}_k) \rangle + \frac{M}{2} \|t\mathbf{d}_{k+1}\|_2^2 \\ &= f(\mathbf{x}_k) - t\lambda_k^2 + \frac{Mt^2}{2} \|\mathbf{d}_{k+1}\|_2^2 \\ &\leq f(\mathbf{x}_k) - t\lambda_k^2 + \frac{Mt^2}{2m} \lambda_k^2. \end{aligned}$$

Plugging in $t = m/M$ above yields

$$\begin{aligned} f(\mathbf{x}_k + \alpha_{\text{exact}} \mathbf{d}_{k+1}) - f(\mathbf{x}_k) &\leq f\left(\mathbf{x}_k + \frac{m}{M} \mathbf{d}_{k+1}\right) - f(\mathbf{x}_k) \\ &\leq -\frac{m}{2M} \lambda_k^2 \\ &\leq -\frac{m}{2M^2} \|\nabla f(\mathbf{x}_k)\|_2^2 \\ &\leq -\frac{m^5}{2L^2 M^2}. \end{aligned}$$

Quadratic convergence

When

$$\|\nabla f(\mathbf{x}_k)\|_2 < m^2/L,$$

we start to settle things very quickly. We will assume that in this stage, we choose the step size to be $\alpha_k = 1$. In fact, you can show that under very mild assumptions on the backtracking parameter ($c < 1/3$, to be specific), backtracking will indeed not backtrack at all and return $\alpha_k = 1$ (see [BV04, p. 490]).

We start by pointing out that by construction,

$$\nabla^2 f(\mathbf{x}_k) \mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_k),$$

and so by the fundamental theorem of calculus,

$$\begin{aligned} \nabla f(\mathbf{x}_{k+1}) &= \nabla f(\mathbf{x}_k + \mathbf{d}_{k+1}) - \nabla f(\mathbf{x}_k) - \nabla^2 f(\mathbf{x}_k) \mathbf{d}_{k+1} \\ &= \int_0^1 \nabla^2 f(\mathbf{x}_k + t\mathbf{d}_{k+1}) \mathbf{d}_{k+1} dt - \nabla^2 f(\mathbf{x}_k) \mathbf{d}_{k+1} \\ &= \int_0^1 [\nabla^2 f(\mathbf{x}_k + t\mathbf{d}_{k+1}) - \nabla^2 f(\mathbf{x}_k)] \mathbf{d}_{k+1} dt. \end{aligned}$$

Thus, we obtain

$$\begin{aligned} \|\nabla f(\mathbf{x}_{k+1})\|_2 &\leq \int_0^1 \|\nabla^2 f(\mathbf{x}_k + t\mathbf{d}_{k+1}) - \nabla^2 f(\mathbf{x}_k)\| \cdot \|\mathbf{d}_{k+1}\|_2 dt \\ &\leq \int_0^1 tL \|\mathbf{d}_{k+1}\|_2^2 dt \\ &= \frac{L}{2} \|[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)\|_2^2 \\ &\leq \frac{L}{2m^2} \|\nabla f(\mathbf{x}_k)\|_2^2, \end{aligned}$$

where the second inequality follows from the Lipschitz assumption on the Hessian and the last inequality follows from the fact that the maximum eigenvalue of $(\nabla^2 f(\mathbf{x}_k))^{-2}$ is $1/m^2$. Thus we have

$$\frac{L}{2m^2} \|\nabla f(\mathbf{x}_{k+1})\|_2 \leq \left(\frac{L}{2m^2} \|\nabla f(\mathbf{x}_k)\|_2 \right)^2 \leq \left(\frac{1}{2} \right)^2,$$

where the last inequality follows since $\|\nabla f(\mathbf{x}_k)\|_2 \leq m^2/L$. That is, at every iteration, we are **squaring** the error (which is less than $1/2$). If we entered this stage at iteration ℓ , this means

$$\frac{L}{2m^2} \|\nabla f(\mathbf{x}_k)\|_2 \leq \left(\frac{L}{2m^2} \|\nabla f(\mathbf{x}_\ell)\|_2 \right)^{2^{k-\ell}} \leq \left(\frac{1}{2} \right)^{2^{k-\ell}}.$$

Then using the strong convexity of f ,

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{1}{2m} \|\nabla f(\mathbf{x}_k)\|_2^2 \leq \frac{2m^3}{L^2} \left(\frac{1}{2} \right)^{2^{k-\ell+1}}.$$

The right hand side above is less than ϵ when

$$k - \ell + 1 \geq \log_2 \log_2(\epsilon_0/\epsilon), \quad \epsilon_0 = 2m^3/L^2,$$

so we spend no more than $\log_2 \log_2(\epsilon_0/\epsilon)$ iterations in this phase.

Note that

$$\epsilon = 10^{-20} \epsilon_0 \quad \Rightarrow \quad \log_2 \log_2(\epsilon_0/\epsilon) = 6.0539.$$

Quasi-Newton Methods

A great resource for the material in this section is [NW06, Chapter 6].

Newton's method is great in that it converges to tremendous accuracy in a very surprisingly small number of iterations, especially for smooth functions. It is not so great in that each iteration is extremely expensive. To compute the step direction,

$$\mathbf{d}_k = (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k),$$

we have to

1. compute the gradient (an $N \times 1$ vector),
2. compute the Hessian (an $N \times N$ matrix),
3. invert the Hessian and apply the inverse to the gradient.

Typically, computing the gradient is reasonable (maybe $O(N^2)$ or $O(N)$ computations and storage). Computing and inverting the Hessian might be harder; in general, these operations take $O(N^3)$ computations, and this is something we will have to repeat at every iteration. If N is very large, this can be completely impractical.

At the end of the day, the quadratic model is exactly that – a model. A natural question to ask is if there are alternative quadratic models that might be cheaper while retaining the essential efficacy of Newton. There are, and they are called **quasi-Newton methods**.

Instead of calculating (and inverting) the Hessian at every point, we try to form a simple estimate of the (inverse of the) Hessian. We do this by collecting information about the curvature of the functional

from the point we visit (and their gradients) as we iterate – basically, we are approximating the Hessian (the second derivative) by measuring how the gradients (the first derivative) change from point to point. What is great is that these Hessian estimates (and their inverses) can be quickly updated from one iteration to the next, thus avoiding the expensive matrix inversion.

The cost of these methods is comparable to gradient descent – along with the gradient computation, we will have to do a few matrix-vector multiplies at each iteration, the cost of which is again typically comparable to calculating $\nabla f(\mathbf{x}_k)$. Theoretically, their convergence properties are better than gradient descent, but not as good as Newton. In practice, they typically significantly outperform gradient descent and they are practical for problem sizes where we dare not even dream about computing the Hessian and inverting it.

Approximating the Hessian

Newton’s method works by forming a quadratic model around the current iterate \mathbf{x}_k :

$$\tilde{f}_k(\mathbf{x}) = f(\mathbf{x}_k) + \langle \mathbf{x} - \mathbf{x}_k, \mathbf{g}_k \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T \mathbf{H}_k (\mathbf{x} - \mathbf{x}_k).$$

The particular choices of $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ and $\mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)$ are motivated by Taylor’s theorem. We minimize the surrogate functional above to compute the step direction

$$\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k,$$

choosing a step size α_k , then moving

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k.$$

We then repeat with a new quadratic model,

$$\tilde{f}_{k+1}(\mathbf{x}) = f(\mathbf{x}_{k+1}) + \langle \mathbf{x} - \mathbf{x}_{k+1}, \mathbf{g}_{k+1} \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}_{k+1})^T \mathbf{H}_{k+1}(\mathbf{x} - \mathbf{x}_{k+1}).$$

Quasi-Newton methods operate in this same general framework, and keep the same linear term $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$. Rather than using the Hessian, we ask only that our quadratic model yield gradients that are consistent with the true gradient at both the current point \mathbf{x}_{k+1} and the previous point \mathbf{x}_k . That is, we want

$$\nabla \tilde{f}_{k+1}(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_{k+1}) \quad (2)$$

and

$$\nabla \tilde{f}_{k+1}(\mathbf{x}_k) = \nabla f(\mathbf{x}_k). \quad (3)$$

Note that

$$\nabla \tilde{f}_{k+1}(\mathbf{x}) = \mathbf{g}_{k+1} + \mathbf{H}_{k+1}(\mathbf{x} - \mathbf{x}_{k+1}).$$

By setting $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$, (2) will hold automatically no matter what we choose for \mathbf{H}_{k+1} . Thus, we would like to choose \mathbf{H}_{k+1} so that (3) also holds. This will occur provided that

$$\mathbf{g}_{k+1} + \mathbf{H}_{k+1}(\mathbf{x}_k - \mathbf{x}_{k+1}) = \mathbf{g}_k,$$

or more compactly

$$\mathbf{H}_{k+1} \mathbf{s}_k = \mathbf{y}_k, \quad (4)$$

where

$$\mathbf{s}_k := \mathbf{x}_{k+1} - \mathbf{x}_k$$

$$\mathbf{y}_k := \mathbf{g}_{k+1} - \mathbf{g}_k.$$

There are many choices for \mathbf{H}_{k+1} that satisfy (4), even if we add the constraint that it be symmetric and positive definite (which we need to ensure that \mathbf{H}_{k+1} is invertible, allowing us to compute \mathbf{d}_{k+1}). In general, quasi-Newton methods choose \mathbf{H}_{k+1} so that it can be easily computed from \mathbf{H}_k – different update rules lead to different quasi-Newton methods.

BFGS

Perhaps the most widely used quasi-Newton methods, and what is viewed to be the most effective, is called the BFGS⁵ algorithm. BFGS is similar to many other quasi-Newton methods in that it chooses \mathbf{H}_{k+1} to be “close” to the previous \mathbf{H}_k in a certain sense that turns out to have computational advantages. In particular, the BFGS update can be derived as the solution to the optimization problem

$$\underset{\mathbf{H}}{\text{minimize}} \quad \|\mathbf{H} - \mathbf{H}_k\|_{\mathbf{W}} \quad \text{subject to} \quad \mathbf{H}^T = \mathbf{H}, \quad \mathbf{H}\mathbf{s}_k = \mathbf{y}_k,$$

where $\|\cdot\|_{\mathbf{W}}$ is a particular weighted Frobenius norm (see [NW06] for details.)

It turns out that this optimization problem has a closed form solution, giving the BFGS update rule for constructing \mathbf{H}_{k+1} from \mathbf{H}_k :

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} - \frac{\mathbf{H}_k\mathbf{s}_k(\mathbf{H}_k\mathbf{s}_k)^T}{\mathbf{s}_k^T\mathbf{H}_k\mathbf{s}_k}. \quad (5)$$

At each iteration, we update \mathbf{H}_k by adding two rank-1 matrices to \mathbf{H}_k . This is a critical since in the end we need to be able to invert \mathbf{H}_{k+1} to be able to compute the next update. In general this would

⁵Named after Broyden, Fletcher, Goldfarb, and Shanno.

remain a computational challenge, but in this case since we already know \mathbf{H}_k^{-1} (which would have been required at the previous step) and \mathbf{H}_{k+1} is a low-rank update to \mathbf{H}_k , there will be an efficient solution to computing \mathbf{H}_{k+1}^{-1} . As we will see below, the cost of computing \mathbf{H}_{k+1} will be the same order as a vector-matrix multiply (i.e., $O(N^2)$ instead of $O(N^3)$).

However, before we discuss the mechanics of computing \mathbf{H}_{k+1}^{-1} , let us look a bit closer at the BFGS update in (5) and verify that it makes sense. It is easy to check that $\mathbf{H}_{k+1}\mathbf{s}_k = \mathbf{y}_k$ is always satisfied:

$$\begin{aligned}\mathbf{H}_{k+1}\mathbf{s}_k &= \mathbf{H}_k\mathbf{s}_k + \frac{\mathbf{y}_k\mathbf{y}_k^\top\mathbf{s}_k}{\mathbf{y}_k^\top\mathbf{s}_k} - \frac{\mathbf{H}_k\mathbf{s}_k\mathbf{s}_k^\top\mathbf{H}_k^\top\mathbf{s}_k}{\mathbf{s}_k^\top\mathbf{H}_k\mathbf{s}_k} \\ &= \mathbf{H}_k\mathbf{s}_k + \mathbf{y}_k - \mathbf{H}_k\mathbf{s}_k \\ &= \mathbf{y}_k,\end{aligned}$$

where above we exploit the fact that \mathbf{H}_k is symmetric.

It is also the case that if \mathbf{H}_k is positive definite then \mathbf{H}_{k+1} will also be positive definite, provided that f is *strictly* convex.⁶ This follows from the monotonic gradient property of convex functions:

$$\langle \mathbf{x} - \mathbf{y}, \nabla f(\mathbf{x}) - \nabla f(\mathbf{y}) \rangle > 0.$$

Setting $\mathbf{x} = \mathbf{x}_{k+1}$ and $\mathbf{y} = \mathbf{x}_k$, this tells us that $\mathbf{y}_k^\top\mathbf{s}_k > 0$. (This is reassuring, since if $\mathbf{y}_k^\top\mathbf{s}_k = 0$ then this update rule would be somewhat problematic.)

⁶Newton and quasi-Newton algorithms are typically motivated in the context of twice differentiable functions so that the Hessian matrix always exists. Strict convexity ensures that the Hessian is always invertible, which we clearly need. If f is not strictly convex, we can actually still use the BFGS algorithm, but we need to be a bit more careful to ensure that $\mathbf{y}_k^\top\mathbf{s}_k > 0$ and the Hessian remains invertible. We will see later that this can instead be guaranteed as a part of the line search that selects the step size α_k .

Moreover, the fact that $\mathbf{y}_k^T \mathbf{s}_k > 0$ ensures that $\mathbf{y}_k \mathbf{y}_k^T / \mathbf{y}_k^T \mathbf{s}_k$ is positive semidefinite. We show below that

$$\mathbf{H}_k - \frac{\mathbf{H}_k \mathbf{s}_k (\mathbf{H}_k \mathbf{s}_k)^T}{\mathbf{s}_k^T \mathbf{H}_k \mathbf{s}_k} \quad (6)$$

is positive semidefinite. Thus \mathbf{H}_{k+1} is the sum of two positive semidefinite matrices, and hence positive semidefinite as well. In fact, we will be able to show that \mathbf{H}_{k+1} is strictly positive definite by looking closely at the vectors that live in the nullspace of (6).

To see that (6) is positive semidefinite, recall that a symmetric matrix \mathbf{M} is positive semidefinite if $\mathbf{x}^T \mathbf{M} \mathbf{x} \geq 0$ for all $\mathbf{x} \neq \mathbf{0}$. Thus, we would like to show that

$$\mathbf{x}^T \mathbf{H}_k \mathbf{x} \geq \frac{\mathbf{x}^T \mathbf{H}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{H}_k \mathbf{x}}{\mathbf{s}_k^T \mathbf{H}_k \mathbf{s}_k}.$$

Notice that the numerator in the fraction above can be written as $(\mathbf{x}^T \mathbf{H}_k \mathbf{s}_k)^2$. A fact that you can easily verify on your own is that for any symmetric positive definite matrix \mathbf{M} , $\mathbf{x}^T \mathbf{M} \mathbf{y}$ defines a valid inner product. Applying the Cauchy-Schwarz inequality with this inner product yields

$$(\mathbf{x}^T \mathbf{H}_k \mathbf{s}_k)^2 \leq (\mathbf{x}^T \mathbf{H}_k \mathbf{x})(\mathbf{s}_k^T \mathbf{H}_k \mathbf{s}_k)$$

and thus (6) is positive semidefinite, as desired.

From the above we have that \mathbf{H}_{k+1} must be positive semidefinite. We can say more by looking at the eigenvalues of (6) that are zero. From the argument above it is clear that we actually have a strict inequality *unless* \mathbf{x} is proportional to \mathbf{s}_k (since Cauchy-Schwarz is strict unless the vectors are colinear). Said differently, the eigenvalues of (6) are all

strictly positive except for one, which corresponds to the eigenvector \mathbf{s}_k . Thus, the only way that \mathbf{H}_{k+1} could have an eigenvector of zero would be if \mathbf{s}_k also lived in the nullspace of $\mathbf{y}_k \mathbf{y}_k^\top / \mathbf{y}_k^\top \mathbf{s}_k$, but this is explicitly ruled out by the fact that $\mathbf{y}_k^\top \mathbf{s}_k > 0$. Thus, \mathbf{H}_{k+1} must actually be positive definite.

Now, we return to the issue of calculating \mathbf{H}_{k+1}^{-1} . Noting that \mathbf{H}_{k+1} can be expressed as the sum of \mathbf{H}_k plus two additional terms, we can apply the Woodbury matrix identity

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{VA}^{-1} \mathbf{U})^{-1} \mathbf{VA}^{-1}$$

to “simplify” this inverse. After some tedious calculations we arrive at the formula:

$$\mathbf{H}_{k+1}^{-1} = \mathbf{H}_k^{-1} + \frac{(\mathbf{s}_k^\top \mathbf{y}_k + \mathbf{y}_k^\top \mathbf{H}_k^{-1} \mathbf{y}_k)(\mathbf{s}_k \mathbf{s}_k^\top)}{(\mathbf{s}_k^\top \mathbf{y}_k)^2} - \frac{\mathbf{H}_k^{-1} \mathbf{y}_k \mathbf{s}_k^\top + \mathbf{s}_k \mathbf{y}_k^\top \mathbf{H}_k^{-1}}{\mathbf{s}_k^\top \mathbf{y}_k}.$$

Note that the formula above requires computing a matrix-vector product ($\mathbf{H}_k^{-1} \mathbf{y}_k$) and computing two rank-1 matrices (scaled versions of $\mathbf{s}_k \mathbf{s}_k^\top$ and $\mathbf{s}_k \mathbf{y}_k^\top$), but all of these computations are $O(N^2)$ as opposed to $O(N^3)$.

Above, we have spoken only about updates to the quadratic model. The BFGS algorithm requires not only an initial guess \mathbf{x}_0 , but also an initial matrix \mathbf{H}_0 . The most common choice here is take $\mathbf{H}_0 = \mathbf{I}$.

This gives us the following algorithm:

BFGS

Input: $\mathbf{x}_0, \mathbf{H}_0^{-1}$

Initialize: $k = 0, \mathbf{g}_0 = \nabla f(\mathbf{x}_0)$

while not converged **do**

$$\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$$

Select α_k using a line search

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

$$\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$$

$$\mathbf{s} = \mathbf{x}_{k+1} - \mathbf{x}_k, \mathbf{y} = \mathbf{g}_{k+1} - \mathbf{g}_k, \mathbf{a} = \mathbf{H}_k^{-1} \mathbf{y}, \gamma = \mathbf{s}^\top \mathbf{y}$$

$$\mathbf{H}_{k+1}^{-1} = \mathbf{H}_k^{-1} + \frac{\gamma + \mathbf{y}^\top \mathbf{a}}{\gamma^2} \mathbf{s} \mathbf{s}^\top - \frac{1}{\gamma} \mathbf{a} \mathbf{s}^\top - \frac{1}{\gamma} \mathbf{s} \mathbf{a}^\top$$

$$k = k + 1$$

end while

Convergence of BFGS

There are two main convergence results for BFGS with a step size chosen using an appropriate line search.

Global convergence: If f is strongly convex, then BFGS with backtracking converges to \mathbf{x}^* from any starting point \mathbf{x}_0 and initial quadratic model determined by a positive definite matrix \mathbf{H}_0 .

Superlinear local convergence: If f is strongly convex and the *gradient* of f is M -smooth (i.e., the Hessian is Lipschitz), then when we are close to the solution

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2 \leq c_k \|\mathbf{x}_k - \mathbf{x}^*\|_2$$

where $c_k \rightarrow 0$.

This is not quite the quadratic convergence of the Newton method, but it can still be much, much faster than the linear rate given by gradient descent. In practice, there is often times very little difference between the convergence of BFGS and Newton's method.

Line search for BFGS

We can use similar line search methods for BFGS as we have seen before in the context of gradient descent and Newton's method, with two important caveats.

First, in initializing a backtracking search it is important to set the initial step size $\bar{\alpha} = 1$. This ensures that when we get close to a solution we will be taking sufficiently large steps to ensure superlinear convergence.

Second, in addition to the Armijo condition that

$$f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \geq c_1 \alpha_k \langle \mathbf{d}_k, \nabla f(\mathbf{x}_k) \rangle \quad (7)$$

for some $0 < c_1 < 1$, it can be important to also impose an additional “sufficient curvature” condition on the step size:

$$\langle \mathbf{d}_k, \nabla f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \rangle \geq c_2 \langle \mathbf{d}_k, \nabla f(\mathbf{x}_k) \rangle \quad (8)$$

for some $c_1 < c_2 < 1$. In the context of gradient descent, we often ignore (8) and focus only on (7). However, in the context of BFGS (8) also has an important role to play (especially if the objective function being minimized is not strictly convex).

Specifically, (8) guarantees that $\mathbf{y}_k^T \mathbf{s}_k > 0$ at iteration k , which as discussed above ensures that the BFGS update for \mathbf{H}_{k+1} is well-defined and guarantees that \mathbf{H}_{k+1} remains positive definite. To see this, note that for α_k satisfying (8) we have

$$\langle \mathbf{d}_k, \mathbf{g}_{k+1} \rangle \geq c_2 \langle \mathbf{d}_k, \mathbf{g}_k \rangle,$$

which implies that

$$\langle \mathbf{d}_k, \mathbf{g}_{k+1} - \mathbf{g}_k \rangle \geq (c_2 - 1) \langle \mathbf{d}_k, \mathbf{g}_k \rangle.$$

Note $c_2 < 1$, so that $(c_2 - 1) < 0$. Moreover, since \mathbf{d}_k is a descent direction, we have that $\langle \mathbf{d}_k, \mathbf{g}_k \rangle < 0$, and thus the right-hand side above is strictly positive. Thus

$$\langle \mathbf{d}_k, \mathbf{g}_{k+1} - \mathbf{g}_k \rangle = \langle \mathbf{d}_k, \mathbf{y}_k \rangle > 0.$$

Since $\mathbf{s}_k = \alpha_k \mathbf{d}_k$, this also shows that $\mathbf{y}_k^T \mathbf{s}_k > 0$, as desired.

References

- [BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.