Chapter 7

Quasi-Newton Methods

Contents

The main computational bottleneck in Newton's method (6.6) is the computation and inversion of the Hessian matrix in each step. This matrix has size $d \times d$, so it will take up to $\mathcal{O}(d^3)$ time to invert it (or to solve the system $\nabla^2 f(\mathbf{x}_t)\Delta\mathbf{x} = -\nabla f(\mathbf{x}_t)$ that gives us the next Newton step $\Delta\mathbf{x}$). Already in the 1950s, attempts were made to circumvent this constly step, the first one going back to Davidon [Dav59]. To motivate them, let us go back to the 1-dimensional case.

7.1 The secant method

Like Newton's method (6.1), the secant method is an iterative method for finding a zero of a univariate function. Unlike Newton's method, it does not use derivatives and hence does not require the function under consideration to be differentiable. In fact, it is (therefore) much older than Newton's method. Reversing history and starting from the Newton step

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}, \quad t \geq 0,$$

we can derive the secant method by replacing the derivative $f'(x_t)$ with its finite difference approximation

$$\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}}.$$

As we (in the differentiable case) have

$$f'(x_t) = \lim_{x \to x_t} \frac{f(x_t) - f(x)}{x_t - x},$$

we get

$$\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}} \approx f'(x_t)$$

for $|x_t - x_{t-1}|$ small. As the method proceeds, we expect consecutive iterates x_{t-1}, x_t to become closer and closer, so that the *secant step*

$$x_{t+1} := x_t - f(x_t)\frac{x_t - x_{t-1}}{f(x_t) - f(x_{t-1})}, \quad t \geq 1 \tag{7.1}$$

approximates the Newton step (*two* starting values x_0, x_1 need to be chosen here). Figure 7.1 shows what the method does: it constructs the line through the two points $(x_{t-1}, f(x_{t-1}))$ and $(x_t, f(x_t))$ on the graph of f; the next iterate x_{t+1} is where this line intersects the x-axis. Exercise 34 asks you to formally prove this.

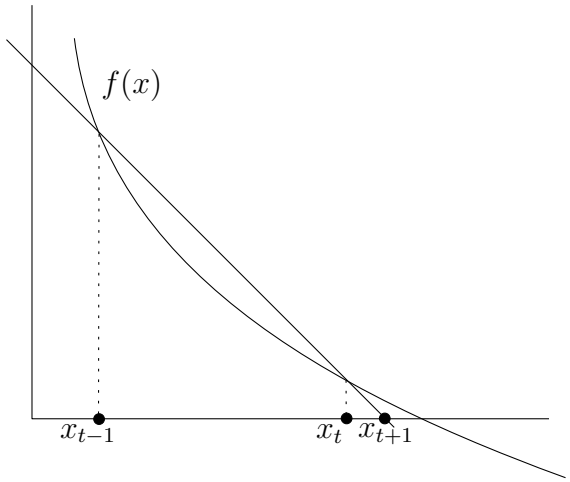

Figure 7.1: One step of the secant method

Convergence of the secant method can be analyzed, but we don't do this here. The main point for us is that we now have a *derivative-free* version of Newton's method.

When the task is to optimize a differentiable univariate function, we can apply the secant method to its derivative to obtain the secant method for optimization:

$$x_{t+1} := x_t - f'(x_t)\frac{x_t - x_{t-1}}{f'(x_t) - f'(x_{t-1})}, \quad t \geq 1. \tag{7.2}$$

This is a *second-derivative-free* version of Newton's method (6.5) for optimization. The plan is now to generalize this to higher dimensions to obtain a *Hessian-free* version of Newton's method (6.6) for optimization over $\mathbb{R}^d$.

7.2 The secant condition

Applying finite difference approximation to the second derivative of f (we're still in the 1-dimensional case), we get

$$H_t := \frac{f'(x_t) - f'(x_{t-1})}{x_t - x_{t-1}} \approx f''(x_t),$$

which we can write as

$$f'(x_t) - f'(x_{t-1}) = H_t(x_t - x_{t-1}) \approx f''(x_t)(x_t - x_{t-1}). \tag{7.3}$$

Now, while Newton's method for optimization uses the update step

$$x_{t+1} = x_t - f''(x_t)^{-1} f'(x_t), \quad t \geq 0,$$

the secant method works with the approximation $H_t \approx f''(x_t)$:

$$x_{t+1} = x_t - H_t^{-1} f'(x_t), \quad t \geq 1. \tag{7.4}$$

The fact that H_t approximates $f''(x_t)$ in the twice differentiable case was our motivation for the secant method, but in the method itself, there is no reference to f'' (which is exactly the point). All that is needed is the *secant condition* from (7.3) that defines H_t:

$$f'(x_t) - f'(x_{t-1}) = H_t(x_t - x_{t-1}). \tag{7.5}$$

This view can be generalized to higher dimensions. If $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable, (7.4) becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t), \quad t \geq 1, \tag{7.6}$$

where $H_t \in \mathbb{R}^{d \times d}$ is now supposed to be a symmetric matrix satisfying the d-dimensional secant condition

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}). \tag{7.7}$$

7.3 Quasi-Newton methods

If f is twice differentiable, the secant condition (7.7) along with the first-order Taylor approximation of $\nabla f(\mathbf{x})$ yields the d-dimensional analog of (7.3):

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}) \approx \nabla^2 f(\mathbf{x}_t)(\mathbf{x}_t - \mathbf{x}_{t-1}),$$

This tells us that (7.6) is meant to approximates Newton's method. Therefore, whenever we use (7.6) with a symmetric matrix satisfying the secant condition (7.7), we say that we have a *Quasi-Newton method*.

In the 1-dimensional case, there is only one Quasi-Newton method—the secant method (7.1). Indeed, equation (7.5) uniquely defines the number H_t in each step.

But in the d-dimensional case, the matrix H_t in the secant condition is underdetermined, starting from $d = 2$: Taking the symmetry requirement into account, (7.7) is a system of d equations in $(d+1)/2$ unknowns, so if it is satisfiable at all, there are many solutions H_t. This raises the question of which one to choose, and how to do so efficiently; after all, we want to get some savings over Newton's method.

Newton's method is a Quasi-Newton method if and only if f is a non-degenerate quadratic function (Exercise 35). Hence, Quasi-Newton methods do not generalize Newton's method but form a family of related algorithms.

The first Quasi-Newton method was developed by William C. Davidon in 1956; he desperately needed iterations that were faster than those of Newton's method in order obtain results in the short time spans between expected failures of the room-sized computer that he used to run his computations on.

But the paper he wrote about his new method got rejected for lacking a convergence analysis, and for allegedly dubious notation. It became a very influential Technical Report in 1959 [Dav59] and was finally officially published in 1991, with a foreword giving the historical context [Dav91]. Ironically, Quasi-Newton methods are today the methods of choice in a number of relevant machine learning applications.

7.4 Greenstadt's approach *(Optional Material)*

Suppose that in a Quasi Newton method, we have the iterates $\mathbf{x}_{t-1}, \mathbf{x}_t$ as well as the matrix H_{t-1}^{-1}; now we want to compute a matrix H_t^{-1} to perform the next Quasi-Newton step (7.6). Greenstadt's approach from 1970 [Gre70] is to update H_{t-1}^{-1} by an "error matrix" E_t to obtain

$$H_t^{-1} = H_{t-1}^{-1} + E_t.$$

Moreover, the errors should be as small as possible, subject to the constraints that H_t^{-1} is symmetric and satisfies the secant condition (7.7). A natural and easy measure of error introduced by an update matrix E is its squared *Frobenius norm*

$$\|E\|_F^2 := \sum_{i=1}^{d} \sum_{j=1}^{d} E_{ij}^2.$$

The Frobenius norm itself is simply the 2-norm of the long vector with d^2 entries that we get by writing all matrix entries next to each other. Since Greenstadt considered the resulting Quasi-Newton method as "too specialized", he searched for a compromise between variability in the method and simplicity of the resulting formulas. This led him to minimize the error term

$$\|AEA^\top\|_F^2,$$

where $A \in \mathbb{R}^{d\times d}$ is some fixed invertible transformation matrix. If $A = I$, we recover the squared Frobenius norm of E.

Let us now fix t and simplify notation by setting

$$\begin{aligned} H &:= H_{t-1}^{-1}, \\ H' &:= H_t^{-1}, \\ E &:= E_t, \\ \boldsymbol{\sigma} &:= \mathbf{x}_t - \mathbf{x}_{t-1}, \\ \mathbf{y} &= \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}), \\ \mathbf{r} &= \boldsymbol{\sigma} - H\mathbf{y}. \end{aligned}$$

The update formula then is

$$H' = H + E, \tag{7.8}$$

and the secant condition $\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1})$ becomes

$$H'\mathbf{y} = \boldsymbol{\sigma} \quad (\Leftrightarrow E\mathbf{y} = \mathbf{r}). \tag{7.9}$$

Greenstadt's approach can now be distilled into the following convex constrained minimization problem in the d^2 variables E_{ij}:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|AEA^\top\|_F^2 \\ \text{subject to} & E\mathbf{y} = \mathbf{r} \\ & E^\top - E = 0 \end{array} \tag{7.10}$$

7.4.1 The method of Lagrange multipliers

Minimization subject to equality constraints can be done via the method of *Lagrange multipliers*. Here we need it only for the case of *linear* equality constraints in which case the method assumes a very simple form.

Theorem 7.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, $C \in \mathbb{R}^{n\times d}$ for some $n \in \mathbb{N}$, $\mathbf{e} \in \mathbb{R}^n$, $\mathbf{x}^\star \in \mathbb{R}^d$ such that $C\mathbf{x}^\star = \mathbf{e}$. Then the following two statements are equivalent.*

(i) $\mathbf{x}^\star = \operatorname{argmin}\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^d, C\mathbf{x} = \mathbf{e}\}$

(ii) There exists a vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ such that

$$\nabla f(\mathbf{x}^\star)^\top = \boldsymbol{\lambda}^\top C.$$

The entries of $\boldsymbol{\lambda}$ are known as the Lagrange multipliers.

Proof. The easy direction is (ii)⇒(i): if $\boldsymbol{\lambda}$ as specified exists and $\mathbf{x} \in \mathbb{R}^n$ satisfies $C\mathbf{x} = \mathbf{e}$, we get

$$\nabla f(\mathbf{x}^\star)^\top(\mathbf{x} - \mathbf{x}^\star) = \boldsymbol{\lambda}^\top C(\mathbf{x} - \mathbf{x}^\star) = \boldsymbol{\lambda}^\top(\mathbf{e} - \mathbf{e}) = 0.$$

Hence, $\mathbf{x}^\star$ is a minimizer of f over $\{\mathbf{x} \in \mathbb{R}^d : C\mathbf{x} = \mathbf{e}\}$ by the optimality condition of Lemma 1.17.

The other direction is Exercise 36. □

In order to apply this method to (7.10), we need to compute the gradient of $f(E) = \frac{1}{2}\|AEA^\top\|_F^2$. Formally, this is a d^2-dimensional vector, but it is customary and more practical to write it as a matrix again,

$$\nabla f(E) = \left(\frac{\partial f(E)}{\partial E_{ij}}\right)_{1\le i,j\le d}.$$

Fact 7.2 (Exercise 37)**.** *Let $A, B \in \mathbb{R}^{d\times d}$ two matrices. With $f : \mathbb{R}^{d\times d} \to \mathbb{R}$, $f(E) := \frac{1}{2}\|AEB\|_F^2$, we have*

$$\nabla f(E) = A^\top AEBB^\top.$$

The second step is to write the system of equations $E\mathbf{y} = \mathbf{r}, E^\top - E = 0$ in Greenstadt's convex program (7.10) in matrix form $C\mathbf{x} = \mathbf{e}$ so that we can apply the method of Lagrange multipliers according to Theorem 7.1.

As there are $d + d^2$ equations in d^2 variables, it is best to think of the rows of C as being indexed with elements $i \in [d] := \{1, \dots, d\}$ for the first d equations $E\mathbf{y} = \mathbf{r}$, and pairs $(i, j) \in [d] \times [d]$ for the last d^2 symmetry constraints (more than half of which are redundant but we don't care). Columns of C are indexed with pairs (i, j) as well.

Let us denote by $\boldsymbol{\lambda} \in \mathbb{R}^d$ the Lagrange multipliers for the first d equations and $\Gamma \in \mathbb{R}^{d\times d}$ the ones for the last d^2 ones.

In column (i, j) of C corresponding to variable E_{ij}, we have entry y_j in row i as well as entries 1 (row (j, i)) and -1 (row (i, j)). Taking the inner product with the Lagrange multipliers, this column therefore yields

$$\lambda_i y_j + \Gamma_{ji} - \Gamma_{ij}.$$

After aggregating these entries into a $d \times d$ matrix, Theorem 7.1 tells us that we should aim for equality with $\nabla f(E)$ as derived in Fact 7.2. We have proved the following intermediate result.

Lemma 7.3. *An update matrix $E^\star$ satisfying the constraints $E\mathbf{y} = \mathbf{r}$ (secant condition in the next step) and $E^\top - E = 0$ (symmetry) is a minimizer of the error function $f(E) := \frac{1}{2}\|AEA^\top\|_F^2$ subject to the aforementioned constraints if and only if there exists a vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ and a matrix $\Gamma \in \mathbb{R}^{d\times d}$ such that*

$$WE^\star W = \boldsymbol{\lambda}\mathbf{y}^\top + \Gamma^\top - \Gamma, \tag{7.11}$$

where $W := A^\top A$ (a symmetric and posititive definite matrix).

Note that $\boldsymbol{\lambda}\mathbf{y}^\top$ is the *outer product* of a column and a row vector and hence a matrix. As we assume A to be invertible, the quadratic function $f(E)$ is easily seen to be strongly convex and as a consequence has a unique minimizer $E^\star$ subject to the set of linear equations in (7.10) (see Lemma 2.9 which also applies if we minimize over a closed set). Hence, we know that the miminmizer $E^\star$ and corresponding Lagrange multipiers $\boldsymbol{\lambda}, \Gamma$ exist.

7.4.2 The Greenstadt family

We need to solve the system of equations

$$\begin{aligned} E\mathbf{y} &= \mathbf{r}, && (7.12)\\ E^\top - E &= 0, && (7.13)\\ WEW &= \boldsymbol{\lambda}\mathbf{y}^\top + \Gamma^\top - \Gamma. && (7.14) \end{aligned}$$

This system is linear in $E, \boldsymbol{\lambda}, \Gamma$, hence easy to solve computationally. However, we want a formula for the unique solution $E^\star$ in terms of the parameters $W, \mathbf{y}, \boldsymbol{\sigma} = \mathbf{r} + H\mathbf{y}$. In the following derivation, we closely follow Greenstadt [Gre70, pages 4–5].

With $M := W^{-1}$ (which exists since $W = A^\top A$ is positive definite), (7.14) can be rewritten as

$$E = M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \Gamma^\top - \Gamma\right) M. \tag{7.15}$$

Transposing this system (using that M is symmetric) yields

$$E^\top = M\left(\mathbf{y}\boldsymbol{\lambda}^\top + \Gamma - \Gamma^\top\right) M.$$

By symmetry (7.13), we can subtract the latter two equations to obtain

$$M\left(\boldsymbol{\lambda}\mathbf{y}^\top - \mathbf{y}\boldsymbol{\lambda}^\top + 2\Gamma^\top - 2\Gamma\right) M = 0.$$

As M is invertible, this is equivalent to

$$\Gamma^\top - \Gamma = \frac{1}{2}\left(\mathbf{y}\boldsymbol{\lambda}^\top - \boldsymbol{\lambda}\mathbf{y}^\top\right),$$

so we can eliminate Γ by substituting back into (7.15):

$$E = M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \frac{1}{2}\left(\mathbf{y}\boldsymbol{\lambda}^\top - \boldsymbol{\lambda}\mathbf{y}^\top\right)\right) M = \frac{1}{2} M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right) M. \tag{7.16}$$

To also eliminate $\boldsymbol{\lambda}$, we now use (7.12)—the secant condition in the next step—to get

$$E\mathbf{y} = \frac{1}{2} M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right) M\mathbf{y} = \mathbf{r}.$$

Premultiplying with $2M^{-1}$ gives

$$2M^{-1}\mathbf{r} = \left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right) M\mathbf{y} = \boldsymbol{\lambda}\mathbf{y}^\top M\mathbf{y} + \mathbf{y}\boldsymbol{\lambda}^\top M\mathbf{y}.$$

Hence,

$$\boldsymbol{\lambda} = \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2M^{-1}\mathbf{r} - \mathbf{y}\boldsymbol{\lambda}^\top M\mathbf{y}\right). \tag{7.17}$$

To get rid of $\boldsymbol{\lambda}$ on the right hand side, we premultiply this with $\mathbf{y}^\top M$ to obtain

$$\underbrace{\mathbf{y}^\top M\boldsymbol{\lambda}}_{z} = \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2\mathbf{y}^\top\mathbf{r} - (\mathbf{y}^\top M\mathbf{y})(\underbrace{\boldsymbol{\lambda}^\top M\mathbf{y}}_{z})\right) = \frac{2\mathbf{y}^\top\mathbf{r}}{\mathbf{y}^\top M\mathbf{y}} - \underbrace{\boldsymbol{\lambda}^\top M\mathbf{y}}_{z}$$

It follows that

$$z = \boldsymbol{\lambda}^\top M\mathbf{y} = \frac{\mathbf{y}^\top\mathbf{r}}{\mathbf{y}^\top M\mathbf{y}}.$$

This in turn can be substituted into the right-hand side of (7.17) to remove $\boldsymbol{\lambda}$ there, and we get

$$\boldsymbol{\lambda} = \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2M^{-1}\mathbf{r} - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}\mathbf{y}\right).$$

Consequently,

$$\begin{aligned}
\boldsymbol{\lambda}\mathbf{y}^\top &= \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2M^{-1}\mathbf{r}\mathbf{y}^\top - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}\mathbf{y}\mathbf{y}^\top\right), \\
\mathbf{y}\boldsymbol{\lambda}^\top &= \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2\mathbf{y}\mathbf{r}^\top M^{-1} - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}\mathbf{y}\mathbf{y}^\top\right).
\end{aligned}$$

This gives us an explicit formula for E, by substituting the previous expressions back into (7.16). For this, we compute

$$\begin{aligned}
M\boldsymbol{\lambda}\mathbf{y}^\top M &= \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2\mathbf{r}\mathbf{y}^\top M - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}M\mathbf{y}\mathbf{y}^\top M\right), \\
M\mathbf{y}\boldsymbol{\lambda}^\top M &= \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(2M\mathbf{y}\mathbf{r}^\top - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}M\mathbf{y}\mathbf{y}^\top M\right),
\end{aligned}$$

and consequently,

$$E = \frac{1}{2}M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right)M = \frac{1}{\mathbf{y}^\top M\mathbf{y}} \left(\mathbf{r}\mathbf{y}^\top M + M\mathbf{y}\mathbf{r}^\top - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}M\mathbf{y}\mathbf{y}^\top M\right). \tag{7.18}$$

Finally, we use $\mathbf{r} = \boldsymbol{\sigma} - H\mathbf{y}$ to obtain the update matrix $E^\star$ in terms of the original parameters $H = H_{t-1}^{-1}$ (previous approximation of the inverse Hessian that we now want to update to $H_t^{-1} = H' = H + E^\star$), $\boldsymbol{\sigma} = \mathbf{x}_t - \mathbf{x}_{t-1}$ (previous Quasi-Newton step) and $\mathbf{y} = \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})$ (previous change in gradients). This gives us the Greenstadt family of Quasi-Newton methods.

Definition 7.4. *Let $M \in \mathbb{R}^{d\times d}$ be a symmetric and invertible matrix. Consider the Quasi-Newton method*

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H_t^{-1}\nabla f(\mathbf{x}_t), \quad t \geq 1,$$

where $H_0 = I$ (or some other positive definite matrix), and $H_t^{-1} = H_{t-1}^{-1} + E_t$ is chosen for all $t \geq 1$ in such a way that H_t^{-1} is symmetric and satisfies the secant condition

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}).$$

For any fixed t, set

$$\begin{aligned} H &:= H_{t-1}^{-1}, \\ H' &:= H_t^{-1}, \\ \boldsymbol{\sigma} &:= \mathbf{x}_t - \mathbf{x}_{t-1}, \\ \mathbf{y} &= \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}), \end{aligned}$$

and define

$$\begin{aligned} E^\star = \frac{1}{\mathbf{y}^\top M\mathbf{y}}\Big(\boldsymbol{\sigma}\mathbf{y}^\top M &+ M\mathbf{y}\boldsymbol{\sigma}^\top - H\mathbf{y}\mathbf{y}^\top M - M\mathbf{y}\mathbf{y}^\top H \\ &- \frac{1}{\mathbf{y}^\top M\mathbf{y}}(\mathbf{y}^\top\boldsymbol{\sigma} - \mathbf{y}^\top H\mathbf{y})M\mathbf{y}\mathbf{y}^\top M\Big). \end{aligned} \quad (7.19)$$

If the update matrix $E_t = E^\star$ is used, the method is called the Greenstadt method *with parameter M.*

7.4.3 The BFGS method

In his paper, Greenstadt suggested two obvious choices for the matrix M In Definition 7.4, namely $M = H$ (the previous approximation of the inverse Hessian) and $M = I$. In the next paper of the same issue of the same

journal, Goldfarb suggested to use the matrix $M = H'$, the *next* approximation of the inverse Hessian. Even though we don't yet have it, we can use it in the formula (7.19) since we know that H' will by design satisfy the secant condition $H'\mathbf{y} = \boldsymbol{\sigma}$. And as M always appears next to $\mathbf{y}$ in (7.19), $M\mathbf{y} = H'\mathbf{y} = \boldsymbol{\sigma}$, so H' disappears from the formula!

Definition 7.5. *The* BFGS method *is the Greenstadt method with parameter $H' = H_t^{-1}$ in step t, in which case the update matrix $E^\star$ assumes the form*

$$\begin{aligned} E^\star &= \frac{1}{\mathbf{y}^\top\boldsymbol{\sigma}}\Big(2\boldsymbol{\sigma}\boldsymbol{\sigma}^\top - H\mathbf{y}\boldsymbol{\sigma}^\top - \boldsymbol{\sigma}\mathbf{y}^\top H - \frac{1}{\boldsymbol{\sigma}^\top\mathbf{y}}(\mathbf{y}^\top\boldsymbol{\sigma} - \mathbf{y}^\top H\mathbf{y})\boldsymbol{\sigma}\boldsymbol{\sigma}^\top\Big) \\ &= \frac{1}{\mathbf{y}^\top\boldsymbol{\sigma}}\Big(-H\mathbf{y}\boldsymbol{\sigma}^\top - \boldsymbol{\sigma}\mathbf{y}^\top H + \Big(1 + \frac{\mathbf{y}^\top H\mathbf{y}}{\mathbf{y}^\top\boldsymbol{\sigma}}\Big)\boldsymbol{\sigma}\boldsymbol{\sigma}^\top\Big). \end{aligned} \tag{7.20}$$

The method is named after Broyden, Fletcher, Goldfarb and Shanno who all came up with it independently around 1970. Greenstadt's name is mostly forgotten.

Let's take a step back and see what we have achieved. Recall that our starting point was that Newton's method needs to compute and invert Hessian matrices in each iteration and therefore has in practice a cost of $O(d^3)$ per iteration. Did we improve over this?

First of all, any method in Greenstadt's family avoids the computation of Hessian matrices altogether. Only gradients are needed. In the BFGS method in particular, the cost per iteration drops to $O(d^2)$. Indeed, the computation of the update matrix $E^\star$ in Definition 7.5 reduces to matrix-vector multiplications and outer-product computations, all of which can be done in $O(d^2)$ time.

Newton and Quasi-Newton methods are often performed with *scaled steps*. This means that the iteration becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t H_t^{-1}\nabla f(\mathbf{x}_t), \quad t \geq 1, \tag{7.21}$$

for some $\alpha_t \in \mathbb{R}_+$. This parameter can for example be chosen such that $f(\mathbf{x}_{t+1})$ is minimized (line search). Another approach is *backtracking line search* where we start with $\alpha_t = 1$, and as long as this does not lead to sufficient progress, we halve α_t. Line search ensures that the matrices H_t^{-1} in the BFGS method remain positive definite [Gol70].

As the Greenstadt update method just depends on the step $\boldsymbol{\sigma} = \mathbf{x}_t - \mathbf{x}_{t-1}$ but not on how it was obtained, the update works in exactly the same way as before even if scaled steps are being used.

7.4.4 The L-BFGS method

In high dimensions d, even an iteration cost of $O(d^2)$ as in the BFGS method may be prohibitive. In fact, already at the end of the 1970s, the first *limited memory* (and limited time) variants of the method have been proposed. Here we essentially follow Nocedal [Noc80]. The idea is to use only information from the previous m iterations, for some small value of m, and "forget" anything older. In order to describe the resulting L-BFGS method, we first rewrite the BFGS update formula in product form.

Observation 7.6. *With $E^\star$ as in Definition 7.5 and $H' = H + E^\star$, we have*

$$H' = \left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) H \left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) + \frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}. \tag{7.22}$$

To verify this, simply expand the product in the right-hand side and compare with (7.20).

We further observe that we do not need the actual matrix $H' = H_t^{-1}$ to perform the next Quasi-Newton step (7.6), but only the vector $H'\nabla f(\mathbf{x}_t)$. Here is the crucial insight.

Lemma 7.7. *Let H, H' as in Observation 7.6, and let $\mathbf{g}' \in \mathbb{R}^d$. Suppose that we have an oracle to compute $\mathbf{s} = H\mathbf{g}$ for any vector $\mathbf{g}$. Then $\mathbf{s}' = H'\mathbf{g}'$ can be computed with one oracle call and $O(d)$ additional arithmetic operations.*

Proof. From (7.22), we conclude that

$$H'\mathbf{g}' = \underbrace{\underbrace{\left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) H \underbrace{\underbrace{\left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right)\mathbf{g}'}_{\mathbf{g}}}_{\mathbf{s}}}_{\mathbf{w}} + \underbrace{\frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\mathbf{g}'}_{\mathbf{h}}}_{\mathbf{z}}.$$

We compute the vectors $\mathbf{h}, \mathbf{g}, \mathbf{s}, \mathbf{w}, \mathbf{z}$ in turn. We have

$$\mathbf{h} = \frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\mathbf{g}' = \boldsymbol{\sigma}\frac{\boldsymbol{\sigma}^\top\mathbf{g}'}{\mathbf{y}^\top\boldsymbol{\sigma}},$$

so $\mathbf{h}$ can be computed with two inner products, a real division, and a multiplication of $\boldsymbol{\sigma}$ with a scalar. For $\mathbf{g}$, we obtain

$$\mathbf{g} = \left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right)\mathbf{g}' = \mathbf{g}' - \mathbf{y}\frac{\boldsymbol{\sigma}^\top\mathbf{g}'}{\mathbf{y}^\top\boldsymbol{\sigma}}.$$

which is a multiplication of $\mathbf{y}$ with a scalar that we already know, followed by a vector addition. To get $\mathbf{s} = H\mathbf{g}$, we call the oracle. For $\mathbf{w}$, we similarly have

$$\mathbf{w} = \left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right)\mathbf{s} = \mathbf{s} - \boldsymbol{\sigma}\frac{\mathbf{y}^\top\mathbf{s}}{\mathbf{y}^\top\boldsymbol{\sigma}},$$

which is one inner product (the other one we already know), a real divison, a multiplication of $\boldsymbol{\sigma}$ with a scalar, and a vector addition. Finally,

$$H'\mathbf{g}' = \mathbf{z} = \mathbf{w} + \mathbf{h}$$

is a vector addition. In total, we needed three inner product computations, three scalar multiplications, three vector additions, two real divisions, and one oracle call. □

How do we implement the oracle? We simply apply the previous Lemma recursively. Let

$$\begin{aligned}\boldsymbol{\sigma}_k &= \mathbf{x}_k - \mathbf{x}_{k-1},\\ \mathbf{y}_k &= \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1})\end{aligned}$$

be the values of $\boldsymbol{\sigma}$ and $\mathbf{y}$ in iteration k. When we perform the Quasi-Newton step $\mathbf{x}_{t+1} = \mathbf{x}_t - H_t^{-1}\nabla f(\mathbf{x}_t)$ in iteration $t \geq 1$, we have already computed these vectors for $k = 1, \ldots, t$. Using Lemma 7.7, we could therefore call the recursive procedure in Figure 7.2 with $k = t, \mathbf{g}' = \nabla f(\mathbf{x}_t)$ to compute the required vector $H_t^{-1}\nabla f(\mathbf{x}_t)$ in iteration t. To maintain the immediate connection to Lemma 7.7, we refrain from introducing extra variables for values that occur several times; but in an actual implementation, this would be done, of course.

By Lemma 7.7, the runtime of BFGS-STEP$(t, \nabla f(\mathbf{x}_t))$ is $O(td)$. For $t > d$, this is slower (and needs more memory) than the standard BFGS step according to Definition 7.5 which always takes $O(d^2)$ time.

The benefit of the recursive variant is that it can easily be adapted to a step that is *faster* (and needs *less* memory) than the standard BFGS step.

function BFGS-STEP($k, \mathbf{g}'$) ▷ returns $H_k^{-1}\mathbf{g}'$
  if $k = 0$ **then**
    return $H_0^{-1}\mathbf{g}'$
  else ▷ apply Lemma 7.7
    $\mathbf{h} = \boldsymbol{\sigma}\dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{g} = \mathbf{g}' - \mathbf{y}\dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{s} =$ BFGS-STEP$(k-1, \mathbf{g})$
    $\mathbf{w} = \mathbf{s} - \boldsymbol{\sigma}_k\dfrac{\mathbf{y}_k^\top \mathbf{s}}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{z} = \mathbf{w} + \mathbf{h}$
    return $\mathbf{z}$
  end if
end function

Figure 7.2: Recursive view of the BFGS method. To compute $H_t^{-1}\nabla f(\mathbf{x}_t)$, call the function with arguments $(t, \nabla f(\mathbf{x}_t))$; values $\boldsymbol{\sigma}_k, \mathbf{y}_k$ from iterations $1, \ldots, t$ are assumed to be available.

The idea is to let the recursion bottom out after a fixed number m of recursive calls (in practice, values of $m \leq 10$ are not uncommon). The step then has runtime $O(md)$ which is a substantial saving over the standard step if m is much smaller than d.

The only remaining question is what we return when the recursion bottoms out prematurely at $k = t - m$. As we don't know the matrix H_{t-m}^{-1}, we cannot return $H_{t-m}^{-1}\mathbf{g}'$ (which would be the correct output in this case). Instead, we pretend that we have started the whole method just now and use our initial matrix H_0 instead of H_{t-m}.[1] The resulting algorithm is depicted in Figure 7.3.

Note that the L-BFGS method is still a Quasi-Newton method as long as $m \geq 1$: if we go through at least one update step of the form $H' = H + E$, the matrix H' will satisfy the secant condition by design.

[1] In practice, we can do better: as we already have some information from previous steps, we can use this information to construct a more tuned H_0. We don't go into this here.

function L-BFGS-STEP($k, \ell, \mathbf{g}'$) ▷ $\ell \le k$; returns $\mathbf{s}' \approx H_k^{-1}\mathbf{g}'$
  if $\ell = 0$ **then**
    return $H_0^{-1}\mathbf{g}'$
  else ▷ apply Lemma 7.7
    $\mathbf{h} = \boldsymbol{\sigma}\dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{g} = \mathbf{g}' - \mathbf{y}\dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{s} =$ L-BFGS-STEP$(k-1, \ell-1, \mathbf{g})$
    $\mathbf{w} = \mathbf{s} - \boldsymbol{\sigma}_k\dfrac{\mathbf{y}_k^\top \mathbf{s}}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{z} = \mathbf{w} + \mathbf{h}$
    return $\mathbf{z}$
  end if
end function

Figure 7.3: The L-BFGS method. To compute $H_t^{-1}\nabla f(\mathbf{x}_t)$ based on the previous m iterations, call the function with arguments $(t, m, \nabla f(\mathbf{x}_t))$; values $\boldsymbol{\sigma}_k, \mathbf{y}_k$ from iterations $t-m+1, \dots, t$ are assumed to be available.

7.5 Exercises

Exercise 34. *Consider a step of the secant method:*

$$x_{t+1} = x_t - f(x_t)\frac{x_t - x_{t-1}}{f(x_t) - f(x_{t-1})}, \quad t \ge 1.$$

Assuming that $x_t \ne x_{t-1}$ and $f(x_t) \ne f(x_{t-1})$, prove that the line through the two points $(x_{t-1}, f(x_{t-1}))$ and $(x_t, f(x_t))$ intersects the x-axis at the point $x = x_{t+1}$.

Exercise 35. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a twice differentiable function with nonzero Hessians everywhere. Prove that the following two statements are equivalent.*

(i) f is a nondegenerate quadratic function, meaning that

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top M\mathbf{x} - \mathbf{q}^\top \mathbf{x} + c,$$

where $M \in \mathbb{R}^{d\times d}$ is an invertible symmetric matrix, $\mathbf{q} \in \mathbb{R}^d, c \in R$ (see also Lemma 6.1).

(ii) Applied to f, Newton's update step

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla^2 f(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t), \quad t \geq 1$$

defines a Quasi-Newton method for all $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^d$.

Exercise 36. *Prove the direction (i)⇒(ii) of Theorem 7.1! You may want to do proceed in the following steps.*

1. *Prove the* Poor Man's Farkas Lemma*: a system of linear equations $A\mathbf{x} = \mathbf{b}$ in d variables has a solution if and only for all $\boldsymbol{\lambda} \in \mathbb{R}^d$, $\boldsymbol{\lambda}^\top A = \mathbf{0}^\top$ implies $\boldsymbol{\lambda}^\top \mathbf{b} = 0$. (You may use the fact that the row rank of a matrix equals its column rank.)*

2. *Argue that $\mathbf{x}^\star = \operatorname{argmin}\{\nabla f(\mathbf{x}^\star)^\top \mathbf{x} : \mathbf{x} \in \mathbb{R}^d, C\mathbf{x} = \mathbf{e}\}$.*

3. *Apply the Poor Man's Farkas Lemma.*

Exercise 37. *Prove Fact 7.2!*

Bibliography

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. https://web.stanford.edu/~boyd/cvxbook/.

[Dav59] William C. Davidon. Variable metric method for minimization. Technical Report ANL-5990, AEC Research and Development, 1959.

[Dav91] William C. Davidon. Variable metric method for minimization. *SIAM J. Optimization*, 1(1):1–17, 1991.

[DSSSC08] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the 1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279, 07 2008.

[Gol70] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

[Gre70] J. Greenstadt. Variations on variable-metric methods. *Mathematics of Computation*, 24(109):1–22, 1970.

[Noc80] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[NP06] Yurii Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, Aug 2006.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *J. R. Statist. Soc. B*, 58(1):267–288, 1996.

[Vis14] Nisheeth Vishnoi. Lecture notes on fundamentals of convex optimization, 2014. https://tcs.epfl.ch/files/content/sites/tcs/files/Lec3-Fall14-Web.pdf.

[Zim16] Judith Zimmermann. *Information Processing for Effective and Stable Admission*. PhD thesis, ETH Zurich, 2016. .