ECE 490: Introduction to Optimization **Fall 2018**

Lecture 14

BFGS Method and Subgradient Method

Lecturer: Bin Hu, Date:10/18/2018

So far we have only talked about optimization of differentiable functions. What if the objective function is not differentiable? Today we will talk about this case and introduce the subgradient method. Before that, we will talk about one more method for optimization of differentiable functions. Specifically, we will talk about the Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method, which is the most popular Quasi-Newton method.

14.1 Quasi-Newton Methods

Quasi-Newton Methods are a family of methods that follow the idea of Newton's Method but estimate the Hessian $\nabla^2 f(x_k)$ with some simpler matrix H_k. Specifically, Quasi-Newton methods have the iteration form:

$$x_{k+1} = x_k - \alpha_k H_k^{-1} \nabla f(x_k)$$

where H_k is some estimated version of $\nabla^2 f(x_k)$, and the stepsize α_k is typically determined by Armijo rule.

Recall that the idea of Newton's method is based on approximating the objective function $f(x)$ as a quadratic function via Taylor expansion:

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

What if we estimate $\nabla^2 f(x_k)$ with some simpler matrix H_k? Specifically, we define

$$g(x) = f(x_k) + \nabla f(x_k)^\intercal (x - x_k) + \frac{1}{2}(x - x_k)^\intercal H_k (x - x_k)$$

Then we hope $g(x) \approx f(x)$ and optimize g for this step. What properties should H_k have such that g is a good estimate for f? It is reasonable to just enforce $\nabla f(x_k) = \nabla g(x_k)$ and $\nabla f(x_{k-1}) = \nabla g(x_{k-1})$. The condition $\nabla f(x_k) = \nabla g(x_k)$ is automatically satisfied. What about $\nabla f(x_{k-1}) = \nabla g(x_{k-1})$? This is equivalent to

$$H_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1}) \tag{14.1}$$

The above condition is called the secant equation. Therefore, we should choose H_k based on this condition. There are infinitely many H_k satisfying this condition. Various choices of H_k lead to different Quasi-Newton methods. We will talk about the most popular one, i.e. the BFGS method.

14.2 BFGS Method

We need H_k to be constructed in a way that it can be efficiently computed. It will be nice if H_k can be computed by some iterative formula $H_k = H_{k-1} + M_{k-1}$. Another nice property we want H_k to have is the positive definiteness. If H_k is positive definite, we can at least guarantee that the BFGS method is a decent method, i.e. $f(x_{k+1}) \leq f(x_k)$. Suppose we choose $H_0 > 0$ and then guarantee $M_k \geq 0$. Then by induction we have the positive definiteness of H_k. So one reasonable thing to do is to set up $\{H_k\}$ using the following iterative formula:

$$H_{k+1} = H_k + a_k v_k v_k^\mathsf{T} + b_k u_k u_k^\mathsf{T} \tag{14.2}$$

where $v_k \in \mathbb{R}^p$ and $u_k \in \mathbb{R}^p$ are some vectors. If $H_0 > 0$, the above iterative formula just guarantees H_k to be positive definite. How can we choose v_k and u_k to guarantee the secant equation $H_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$? Let's denote $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. The secant equation becomes $H_{k+1}s_k = y_k$. Substituting this into (14.2) leads to

$$y_k = H_{k+1}s_k = H_k s_k + a_k v_k v_k^\mathsf{T} s_k + b_k u_k u_k^\mathsf{T} s_k$$

Since $v_k v_k^\mathsf{T} s_k = v_k(v_k^\mathsf{T} s_k) = (v_k^\mathsf{T} s_k)v_k$ and $u_k u_k^\mathsf{T} s_k = u_k(u_k^\mathsf{T} s_k) = (u_k^\mathsf{T} s_k)u_k$, the above equation is just equivalent to

$$y_k - H_k s_k = a_k(v_k^\mathsf{T} s_k)v_k + b_k(u_k^\mathsf{T} s_k)u_k$$

How can we choose v_k, u_k, a_k, and b_k such that the above equation is satisfied? We can choose $v_k = y_k$ and $a_k(v_k^\mathsf{T} s_k) = 1$ so that the first terms on the left and right sides exactly match. Similarly, we can choose $u_k = H_k s_k$ and $b_k(u_k^\mathsf{T} s_k) = -1$ so that the second terms on the left and right sides exactly match. Therefore, we have $v_k = y_k$, $u_k = H_k s_k$, $a_k = \frac{1}{y_k^\mathsf{T} s_k}$, and $b_k = -\frac{1}{s_k^\mathsf{T} H_k s_k}$. The iteration formula (14.2) becomes

$$H_{k+1} = H_k + \frac{y_k y_k^\mathsf{T}}{y_k^\mathsf{T} s_k} - \frac{H_k s_k s_k^\mathsf{T} H_k}{s_k^\mathsf{T} H_k s_k} \tag{14.3}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. This is exactly the BFGS method. When implementing the BFGS method $x_{k+1} = x_k - \alpha_k H_k^{-1} \nabla f(x_k)$, it will be better to directly update H_k^{-1} other than first obtaining H_k and then solving $H_k^{-1}\nabla f(x_k)$. Based on (14.3), one can use the matrix inversion lemma to show

$$H_{k+1}^{-1} = \left(I - \frac{s_k y_k^\mathsf{T}}{y_k^\mathsf{T} s_k}\right) H_k^{-1} \left(I - \frac{y_k s_k^\mathsf{T}}{y_k^\mathsf{T} s_k}\right) + \frac{s_k s_k^\mathsf{T}}{y_k^\mathsf{T} s_k} \tag{14.4}$$

You will be asked to show the above formula in Homework 3. Therefore, for the BFGS method, the computation cost is mainly required for updating (14.4). At each iteration,

the main computation is doing the matrix multiplication twice and the cost scales with $O(p^2)$ if $x \in \mathbb{R}^p$. In contrast, Newton's method requires computing the Hessian $\nabla^2 f(x_k)$ and then solving the linear equation $\nabla^2 f(x_k) d_k = \nabla f(x_k)$. The cost for solving the linear equation $\nabla^2 f(x_k) d_k = \nabla f(x_k)$ scales with $O(p^3)$ in general [1]. Therefore, the per iteration computation cost for Newton's method is the cost for computing Hessian plus some value scaling with $O(p^3)$. This is much higher than the per iteration cost for the BFGS method which roughly scales with $O(p^2)$.

Locally, the BFGS method also achieves superlinear convergence. This is similar to Newton's method. One interpretation for the BFGS update (14.4) is that H_{k+1}^{-1} is chosen to be as close to H_k^{-1} as possible for some appropriate metric quantifying the distance between two matrices. We skip the details of these interpretations.

It is worth mentioning that the BFGS method requires storing H_k^{-1} in memory. When p is large, this could be an issue. Therefore, the limited-memory BFGS (L-BFGS) method is developed. We will not talk about L-BFGS in details in this course.

Compared with the gradient method, Newton's method typically requires much less iterations but the per iteration cost is significantly higher. The BFGS method can be viewed as an interpolation of the gradient method and Newton's method. There is another method called the conjugate gradient method which can also be viewed as some interpolation of the gradient method and Newton's method. Due to the time constraint, we will not cover this method in the class.

14.3 Subgradient Method

We have only talked about the optimization of differential functions. Now let's look at the case where the objective function is not differentiable. Recall for a convex differentiable f, we have

$$f(y) \geq f(x) + \nabla f(x)^\mathsf{T}(y - x) \ \forall y \in \mathbb{R}^p$$

If f is convex and not differentiable at x, we can still find vector g such that $f(y) \geq f(x) + g^\mathsf{T}(y - x) \ \forall y \in \mathbb{R}^p$. This vector g is called the subgradient. Consider a one-dimensional example $f(x) = |x|$. This function is not differentiable at $x = 0$. But there are many subgradients at $x = 0$. Actually any $-1 \leq g \leq 1$ is a subgradient in this case. As long as f is convex, it has some subgradient at every point. If f is differentiable, then at each point it has a unique subgradient which is its gradient. The set of all subgradients at x is called the subdifferential at x, and is denote as $\partial f(x)$. For a convex (possibly non-differentiable) f, if $0 \in \partial f(x^*)$, then x^* is a global min (Verify this by yourself!). Therefore, we are interested in finding points whose subdifferential includes 0.

A straightforward way to generalize the gradient method for convex non-differentiable f is to replace the gradient with the subgradient. This leads to the subgradient method which

[1] When there is some sparsity in the Hessian matrix, one can solve this equation much faster. But in general, $O(p^3)$ is the required cost.

iterates as

$$x_{k+1} = x_k - \alpha g_k$$

where $g_k \in \partial f(x_k)$. If we assume $\|g_k\| \leq G$, then we will be able to use the dissipation inequality approach to show some convergence properties of the subgradient method. Just realize that $\|g_k\| \leq G$ and $f(x_k) - f(x^*) \leq g_k^{\mathsf{T}}(x_k - x^*)$ can both be written as quadratic supply rate conditions. I will let you think about how to use these two supply rate conditions for now. We will talk a little bit more about this in the next lecture.