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

Lecture 8

Unconstrained Optimization of a Sum of Smooth Strongly-Convex Functions

Lecturer: Bin Hu, Date:09/25/2018

In the last lecture, we talked about how Nesterov's method can lead to an improved iteration complexity $O(\sqrt{\frac{L}{m}}\log\frac{1}{\varepsilon})$ when the objective function is L-smooth and m-strongly convex. Is this the end of the story for unconstrained optimization of smooth strongly-convex functions? The answer is no. Sometimes the problem may have some additional structure which can be used to design new algorithms. In this lecture, we look at the finite-sum minimization

$$\min_{x\in\mathbb{R}^p}\frac{1}{n}\sum_{i=1}^{n}f_i(x) \tag{8.1}$$

The objective function has a finite sum structure, i.e. $f=\frac{1}{n}\sum_{i=1}^{n}f_i$. This type of objective functions arise naturally from machine learning. The finite-sum minimization is also called empirical risk minimization (ERM). In ERM, the number n is the number of the data points in the training set. Typically one has $f_i(x)=l_i(x)+\Omega(x)$ where l_i is the loss function preventing underfit and $\Omega(x)$ is the regularizer preventing overfit. So $l_i(x)$ measure how x fits the i-th data point. The smaller $l_i(x)$ is, the better x fits the i-th data point. $\Omega(x)$ measures the complexity of x and can prevents overfitting. More motivations for ERM will be taught in a machine learning course.

If we apply the gradient method or Nesterov's method, we need to evaluate $\nabla f=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i$ for each iteration. In other words, we need to evaluate the gradient on all the data points. The computation cost for each iteration scales with $O(n)$. So the total computation cost to achieve ε-accuracy is $T\times O(n)$ where T is the iteration complexity. For example, the total computation cost for the gradient method scales with

$$O\left(\kappa\log\frac{1}{\varepsilon}\right)\times O(n)=O\left(n\kappa\log\frac{1}{\varepsilon}\right)$$

Similarly, the total computation cost for Nesterov's accelerated method scales with

$$O\left(\sqrt{\kappa}\log\frac{1}{\varepsilon}\right)\times O(n)=O\left(n\sqrt{\kappa}\log\frac{1}{\varepsilon}\right)$$

For big data applications, n is typically very large. The per iteration cost of the gradient method and Nesterov's method is high. This motivates the use of the stochastic gradient descent (SGD) method. In this lecture, we will talk about the performance of SGD. This lecture is completely for exposure purpose and is not going to be tested in homework or exam since some background on probability and statistics may be required. This is the only lecture that we talk about stochastic finite-sum optimization.

8.1 SGD

SGD iterates as

$$x_{k+1} = x_k - \alpha \nabla f_{i_k}(x_k)$$

where i_k is uniformly sampled from $\{1, 2, \ldots, n\}$ in an I.I.D manner. In other words, one data point is sampled at every iteration and the gradient is only evaluated on that data point. By doing this, the computation cost for each iteration does not depend on n and scales with $O(1)$. The hope is that there will be a lot of redundancy between data points and SGD will work well in some average sense in the long run. In the extreme case where $f_i = f_j$ for all (i, j) (all the data points are the same), SGD requires the same iteration number as the gradient method but the per iteration cost is n times smaller. But of course, that is not going to happen for real problems. In general, SGD is very efficient in obtaining a "rough" solution which is sufficiently useful for many learning problems (since one cares more about testing errors for these applications).

Another related algorithm is the incremental gradient method where i_k is based on a prescribed permutation of $\{1, 2, \ldots, n\}$. SGD has some nice statistical properties and currently is the most popular optimization method for machine learning applications. Both the incremental gradient method and SGD cannot give an accurate solution for x^*. To see this, just notice typically we have $\nabla f_i(x^*) \neq 0$ even when $\nabla f(x^*) = 0$. Even if we set $x_0 = x^*$, we will have $x_1 = x_0 - \alpha \nabla f(x^*) \neq x^*$ if α is a constant. So for a constant stepsize α, even when SGD is initialized at the optimal point x^*, it is not going to stay there.

Now we make the following two assumptions:

1. f is m-strongly convex.

2. f_i is L-smooth and convex for all i.

Under these two assumptions, we can show that SGD satisfies a bound in the following form:

$$\mathbb{E}\|x_k - x^*\|^2 \leq \rho^{2k} \|x_0 - x^*\|^2 + H \tag{8.2}$$

where $\rho^2 = 1 - 2m\alpha + O(\alpha^2)$ and $H = O(\alpha)$. Here ρ^2 quantifies the convergence speed and H quantifies the accuracy. Therefore, for SGD, there is a fundamental trade-off between the convergence speed and the accuracy. If one wants a very accurate solution, one has to decrease α so that H is decreased. However, ρ^2 increases as α decreases and the convergence speed becomes slower.

In the deterministic case, we wants to make ρ^2 small. For SGD, we want to balance ρ^2 and H. If only a rough solution is required (i.e. it is OK to allow some inaccuracy at the level of $O(\alpha)$), the rate $\rho^2 = 1 - 2m\alpha + O(\alpha^2)$ matches the order of the convergence rate of the gradient method (recall that in HW1 we have shown $\rho^2 = 1 - 2m\alpha + m^2\alpha^2$ for the gradient method with $\alpha \leq \frac{2}{m+L}$).

8.2 Analysis of SGD

To prove (8.2), we will slightly modify the dissipation inequality approach we have used in the previous lectures. The main difference is that we need to take expectations of the dissipation inequality. The supply rate condition is also different. Instead of $S \leq 0$, we need to use $\mathbb{E}S \leq M$ for some constant M. This difference leads to the trade-off between convergence speed ρ^2 and accuracy H. In addition, we need to look at the general system in the following form:

$$\begin{aligned} \xi_{k+1} &= A\xi_k + Bu_k \\ v_k &= C\xi_k \\ u_k &= \nabla f_{i_k}(v_k) \end{aligned}$$

So instead of $u_k = \nabla f(v_k)$, now we have $u_k = \nabla f_{i_k}(v_k)$. SGD can be written in the above model with $A = I$, $B = -\alpha I$, $C = I$, and $\xi_k = x_k$.

We follow the three-step routine again:

1. We replace $u_k = \nabla f_{i_k}(v_k)$ with some quadratic inequality

$$\mathbb{E}\begin{bmatrix} x_k - x^* \\ u_k \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 & -LI \\ -LI & I \end{bmatrix} \begin{bmatrix} x_k - x^* \\ u_k \end{bmatrix} \leq \frac{2}{n}\sum_{i=1}^{n} \|\nabla f_i(x^*)\|^2 = M \tag{8.3}$$

$$\mathbb{E}\begin{bmatrix} x_k - x^* \\ u_k \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mI & -I \\ -I & 0 \end{bmatrix} \begin{bmatrix} x_k - x^* \\ u_k \end{bmatrix} \leq 0 \tag{8.4}$$

   The above two inequalities not only depend on the function properties (the values of (m, L), etc), but also depend on the sampling strategies of i_k. Hence eventually the above inequalities only hold in the average sense and we have to take expectation of the inequalities. The proofs of these two inequalities require probabilistic arguments and hence are omitted here. Send me an email if you want to know more about the proofs of these two inequalities.

   Now we just set $X_1 = \begin{bmatrix} 0 & -LI \\ -LI & I \end{bmatrix}$ and $X_2 = \begin{bmatrix} 2mI & -I \\ -I & 0 \end{bmatrix}$.

2. Now we test if there exists $P \geq 0$ and non-negative scalers (λ_1, λ_2) such that

$$\begin{bmatrix} A^{\mathsf{T}}PA - \rho^2 P & A^{\mathsf{T}}PB \\ B^{\mathsf{T}}PA & B^{\mathsf{T}}PB \end{bmatrix} - \lambda_1 X_1 - \lambda_2 X_2 \leq 0. \tag{8.5}$$

   If so, we have

$$\begin{aligned} &\mathbb{E}(\xi_{k+1} - \xi^*)^{\mathsf{T}}P(\xi_{k+1} - \xi^*) - \rho^2\mathbb{E}(\xi_k - \xi^*)^{\mathsf{T}}P(\xi_k - \xi^*) \\ \leq &\lambda_1 \mathbb{E}\begin{bmatrix} \xi_k - \xi^* \\ u_k \end{bmatrix}^{\mathsf{T}} X_1 \begin{bmatrix} \xi_k - \xi^* \\ u_k \end{bmatrix} + \lambda_2 \mathbb{E}\begin{bmatrix} \xi_k - \xi^* \\ u_k \end{bmatrix}^{\mathsf{T}} X_2 \begin{bmatrix} \xi_k - \xi^* \\ u_k \end{bmatrix} \\ \leq &\lambda_1 M \end{aligned}$$

For simplicity, we can choose $P = I$. Recall for SGD we have $A = I$ and $B = -\alpha I$. Hence (8.5) is equivalent to

$$\begin{bmatrix} 1-\rho^2 & -\alpha \\ -\alpha & \alpha^2 \end{bmatrix} - \lambda_1 \begin{bmatrix} 0 & -L \\ -L & 1 \end{bmatrix} - \lambda_2 \begin{bmatrix} 2m & -1 \\ -1 & 0 \end{bmatrix} \leq 0 \tag{8.6}$$

Now we set the left side to be a zero matrix. We have $\lambda_1 = \alpha^2$, $\lambda_2 = \alpha - \lambda_1 L$, and $\rho^2 = 1 - 2m\lambda_2 = 1 - 2m\alpha + 2mL\alpha^2$.

3. Now the dissipation inequality leads to

$$\mathbb{E}\|x_{k+1} - x^*\|^2 \leq \rho^2 \mathbb{E}\|x_k - x^*\|^2 + \lambda_1 M$$

Iterating the above bound leads to

$$\begin{aligned} \mathbb{E}\|x_k - x^*\|^2 \leq & \rho^2 \mathbb{E}\|x_{k-1} - x^*\|^2 + \lambda_1 M \\ \leq & \rho^4 \mathbb{E}\|x_{k-1} - x^*\|^2 + (\rho^2 + 1)\lambda_1 M \\ \leq & \rho^{2k} \mathbb{E}\|x_0 - x^*\|^2 + \left(\sum_{t=0}^{\infty} \rho^{2t} \right) \lambda_1 M \\ = & \rho^{2k} \mathbb{E}\|x_0 - x^*\|^2 + \frac{\lambda_1 M}{1-\rho^2} \end{aligned}$$

From Step 2, we have $\rho^2 = 1 - 2m\alpha + 2mL\alpha^2 = 1 - 2m\alpha + O(\alpha^2)$, and $H = \frac{\lambda_1 M}{1-\rho^2} = O(\alpha)$. This leads to the desired conclusion (8.2).

8.3 More discussions

We have shown that for minimization of a sum of smooth strongly-convex functions, SGD achieves a rate similar to the gradient method when one only seeks a "rough" solution. One issue for SGD is that it cannot quickly converge to "accurate" solutions for ERM. If one uses a time varying stepsize $\alpha_k = O(\frac{1}{k})$, SGD can converge to the solution x^* but at a sublinear rate. There are many methods that aim to take advantages of both SGD and the gradient method. Specifically, it is desired to have an algorithm that converges linearly to the optimal solution x^* but the per iteration cost is $O(1)$. The main technique to achieve this is the so-called variance reduction technique. In the past five years, various researchers have proposed many variance reduction methods, e.g. stochastic average gradient (SAG), SVRG, SAGA, Finito, SDCA, SPDC, Katyusha, etc. If you are interested in this topic, just send me an email for discussion.