ECE 490: Introduction to Optimization **Spring 2022**

Supplementary Material for Note 9

Large-Scale Learning: Stochastic Gradient Descent, and Subgradient

Lecturer: Bin Hu, Date:02/24/2022

In this note, we will cover a few more technical issues for optimization in large-scale machine learning.

9.1 Stochastic Gradient Descent (SGD)

In machine learning, many problems are in the form of finite-sum minimization

$$\min_{x\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{9.1}$$

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, 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

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

For big data applications, n is typically very large. The per iteration cost of the gradient method is high. This motivates the use of the stochastic gradient descent (SGD) method. 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.

9.2 Subgradient Method

Sometimes the activation function is not differentiable at certain points. Then we need the concept of subgradient. 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 machine learning tasks with non-smooth activation functions, we can replace σ' with its subgradient.