Indian Institute of Technology Kanpur
CS774 Optimization Techniques

Scribe: Deepanshu Gupta
Instructor: Purushottam Kar
Date: October 17, 2012

LECTURE
18

Stochastic Optimization - II

1 Introduction

Continuing with the discussion on Stochastic Optimization from the last lecture, current lecture will focus on one particular optimization technique: Stochastic Gradient Descent(`SGD`). The lecture shall cover problem and technique of Stochastic Optimization, its analysis, application and also a brief review of Mini-batch `SGD` and Online - `SGD`.

`SGD` in particular is a very popular optimization technique applicable to various Machine Learning algorithms, for example: training Neural Networks and large scale Machine Learning.

Consider the following problem of optimizing an error function $f(\cdot)$ with respect to some parameter $x \in \mathcal{C}$ that performs well in expectation over some data distribution $\mathcal{D}$

$$\min_{x \in \mathcal{C}} f(x) = \mathop{\mathbb{E}}_{\theta \sim \mathcal{D}} F(x, \theta)$$

For example in Classification we hope to achieve lowest possible test error. Similarly in Least Squares Regression we optimize the euclidean error between predictions and observed quantities. To solve these optimization problems 2 approaches are taken:

- `Empirical Average Minimization/Finite Sample Approximation`
  Many Machine Learning Algorithms take this approach. We begin by sampling several data points(Independent and Identically Distributed) and construct a function

  $$\theta^1, \theta^2, \ldots, \theta^n \sim \mathcal{D}$$

  $$\hat{f_n} = \frac{1}{n} \sum_i F(x, \theta^i)$$

  if function F & $\mathcal{C}$ are "well behaved" then we can show that with high probability

  $$\min_x f(x) - \min_x \hat{f_n}(x) \leq \mathcal{O}(\frac{1}{\sqrt{n}})$$

- `Stochastic Optimization`(refers to the problem as well as solution technique)
  We shall cover this in the next section of the scribe.

Algorithm 1: Stochastic Gradient Descent

1: $x^0 \leftarrow \mathbf{0}$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad g^t \leftarrow \texttt{SFO}(f, x^t)$
4: $\quad x^t \leftarrow \Pi_{\mathcal{C}}(x^t - \eta_t g^t)$ //Projection step
5: **end for**
6: **return** x^T

2 Stochastic Optimization

2.1 Algorithm

Definition 18.1. Define a Stochastic First-Order Oracle(SFO) as:

$$g^t \leftarrow \texttt{SFO}(f, x^t) \quad \text{such that}$$
$$\mathbb{E}(g^t|x) = \nabla f(x^t)$$

Remark 18.1. In general x will be a random variable. At every time 't' we need only '1' point which makes the

2.2 Analysis

Assume $x^* \in \arg\min_{x \in \mathcal{C}} f(x)$ and and$f^* = f(x^*)$ where f is a convex function. Also, let $||g^t|| \leq G$.

$$\begin{aligned} ||x^{t+1} - x^*||_2^2 &\leq ||x^t - \eta_t g^t - x^*||_2^2 && \text{projection step \& property} \\ &= ||x^t - x^*||_2^2 + \eta_t^2 ||g^t||_2^2 - 2\eta_t \langle g^t, x^t - x^* \rangle \end{aligned}$$

s

Here, the potential function has to be carefully defined since defining $||x^t - x^*||_2^2$ is a random variable in itself hence we cannot try reducing a random quantity. Hence, we define ϕ_t as our potential function where,

$$\phi_t = \mathbb{E}(||x^t - x^*)||_2^2$$

We could have uses variance instead of expectation as well, however expectation works in our case. Let, $\tilde{\phi_t} = \mathbb{E}[||x^t - x^*||_2^2 | x^{t-1}]$ and denote $\mathbb{E}_t(Y) = \mathbb{E}(Y|X^t)$. By law of iterated expectations, $\mathbb{E}(Y) = \mathbb{E}_X(\mathbb{E}_{Y|X}(Y|X))$.

$$\begin{aligned} \mathbb{E}_t ||x^{t+1} - x^t||_2^2 &\leq ||x^t - x^*||_2^2 + \eta_t^2 G^2 - 2\eta_t \langle \nabla f(x^t), x^t - x^* \rangle \\ &\leq ||x^t - x^*||_2^2 + \eta_t^2 G^2 - 2\eta_t f(x^t) - f^* && \text{By Convexity} \\ \phi_{t+1} &\leq \phi_t + \eta_t^2 G^2 - 2\eta_t (\mathbb{E} f(x^t) - f^*) && \text{Taking expectation} \\ \mathbb{E} f(x^t) - f^* &\leq \frac{\phi_t - \phi_{t+1}}{2\eta_t} + \frac{\eta_t G^2}{2} \end{aligned}$$

Assume $\eta_t = \eta$ and sum last equation over all time periods.

$$\sum_{t=1}^{T}(\mathbb{E}f(x^t) - f^*) \leq \frac{\phi_0}{2\eta} + \frac{T\eta G^2}{2}$$
$$\leq \frac{\phi_0\sqrt{T}}{2} + \frac{G^2\sqrt{T}}{2} \qquad \text{putting } \eta = \frac{1}{\sqrt{T}}$$

We have proved sublinear rate of convergence for Regret for `SGD`, however we need only 1 x. So, to get the optimal x we shall make use of Jensen's inequality.[1]

Hence, consider $x = \frac{\sum_{t=1}^{T} x^t}{T}$ and using Jensen's inequality over $\mathbb{E}f(\frac{\sum_t x^t}{T})$

$$\mathbb{E}f(\frac{\sum_{t=1}^{T} x^t}{T}) - f^* \leq \sum_t \frac{(\mathbb{E}f(x^t))}{T} - f^*$$
$$\mathbb{E}f(\frac{\sum_{t=1}^{T} x^t}{T}) - f^* \leq \frac{\phi_0}{2\sqrt{T}} + \frac{G^2}{2\sqrt{T}}$$

Remark 18.2. One can show that with probability $1 - \delta$

$$f(\frac{1}{T}\sum_{t=1}^{T} x^t) \leq f^* + c\sqrt{\frac{\log(\frac{1}{\delta})}{T}}$$

Proving this result requires theory of martingales which is beyond the scope of this course.

In practice we can use `SGD` to solve Empirical Average Minimization problem.
For example:

2.2.1 `Support vector machines(SVM)`

For (data, label) $= (x^t, y^t)_{t=1}^{T}$ and weight vector given by w. The objective function is

$$\min_w \frac{1}{2}||w||_2^2 + \frac{C}{n}\sum_{t=1}^{n}\max(0, 1 - y^i w^T x^i)$$
$$\min_w \frac{1}{n}\sum_{t=1}^{n}(\frac{1}{2}||w||_2^2 + \frac{C}{n}\max(0, 1 - y^i w^T x^i))$$

Put $y^t x^t = \theta^i$ and $x = w$ then

$$F(x, \theta) = \frac{1}{2}||w||_2^2 + C\max(0, 1 - x^T\theta)$$
$$g = x + c\begin{cases} 0, & x^T\theta > 1 \\ 1, & x^T\theta \leq 1 \end{cases}$$

where, $g \in \partial F(x, \theta)$

[1] **Jensen's Inequality** For any "well-behaved" convex function $f(\cdot)$ over a space $\mathbb{V}$

$$f : \mathbb{V} \to \mathbb{R}$$

Let x be a random variable in space $\mathbb{V}$, then:

$$\mathbb{E}f(x) \geq f(\mathbb{E}(x))$$

Algorithm 2: Mini-batch Stochastic Gradient Descent

$x^0 \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  $h^1, \ldots, h^b \leftarrow \texttt{SFO}(f, x^t)$
  $g^t \leftarrow \frac{1}{b} \sum_{t=1}^{T} h^t$
  $x^t \leftarrow \Pi_{\mathcal{C}}(x^t - \eta_t g^t)$
end for
return x^T

2.2.2 `Mini-batch SGD`

Mini-batch `SGD` is quite popular in Neural Network community for training purposes. We shall consider another situation in which getting gradients of a function are cheap but noisy. Note the Mini-batch `SGD` algorithm: Selecting the size of batch is tricky. Normally, the batch size is taken in between 50 and 100. However, if too large then the advantage of `SGD` is lost and if size taken too large then the procedure becomes slow.

Remark 18.3. There is another version of `SGD` where we sample data point i.i.d called Online Gradient Descent(`OGD`)

`Online Optimization` At each time point propose a x^t and play it. Adversary presents with incurred loss $f_t(x^t)$. Then update your x^{t+1} appropriately.
Online optimization finds huge application in Recommendation Systems. In Online optimization we do not use the concept of reducing error, instead we optimize "Regret" which is the error by which our algorithm is worse then the best performing algorithm had we seen data altogether. Formally Regret(R_T):

$$R_T = \sum_t f_t(x^t) - \min_x \sum_t f_t(x)$$

For more information on Online Optimization refer Hazan et al. (2016)

References

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.