ECE 901: Large-scale Machine Learning and Optimization **Fall 2016**

Lecture 10 — 06/10

Lecturer: Dimitris Papailiopoulos *Scribe: Lingjiao Chen*

Note: These lecture notes are still rough, and have only have been mildly proofread.

10.1 Recap of Projection-based Constraint Optimization

In the previous lecture, we discussed how to solve the convex constraint optimization problem

$$\begin{aligned} &\min f(x) \\ &s.t. x \in C \end{aligned} \tag{10.1}$$

where C is a convex closed set. A widely used method is projected gradient descent (PGD), the key idea of which is to apply projection after gradient descent. More precisely, PGD uses

$$X_{k+1} = P_C(X_k - \gamma_k g_k) \tag{10.2}$$

at iteration k, where γ_k is the step size at iteration k, g_k is the (approximate) gradient at iteration k, and $P_C(x) = \min_{y \in C} ||x - y||_2^2$ is the projection operator. As we have seen from the last lecture, PGD has the advantage that it achieves similar convergence rate as the traditional gradient descent method for unconstrained optimization problem.

However, the computational cost of projection can be prohibitively large. For example, consider the feasible set $C_{PSD(s)} = \{X \in R^n \times n | X \text{ is } psd, Tr(X) \leq s\}$. The computational cost of projection on $C_{PSD(s)}$ in general is $O(n^3)$, which is unacceptable when n is very large. In such scenario, one would seek to have faster algorithms.

10.2 Projection-free Gradient Descent

Note that the key bottleneck of PGD is the projection operator. Thus, a natural idea is to accelerate the algorithm by avoiding projection. Frank-Walfe method utilizes exactly this idea by, roughly speaking, replacing the quadratic term by a linear one in the objective function. The precise algorithm is shown in Algorithm 1.

10.2.1 An demonstrative example

Considering that the key component in Frank-Walfe Algorithm is $g_k = \arg\min_{y \in C} < y, g_k^{FULL} >$, we are interested in obtaining an intuitive understanding of it. Consider $C = \{X | ||X||_2 \leq L\}$. Then we have $g_k = \arg\min_{y \in C} < y, g_k^{FULL} >= -L \frac{\nabla f(x)}{||\nabla f(x)||_2}$. In other words, Frank-Walfe reduces to the classic gradient descent method.

Algorithm 1 Frank-Walfe Algorithm

```
$X_0 \leftarrow$ initial value
for $k = 0 : T$ do
    $g_k^{FULL} = \nabla f(X_k)$
    $g_k = \arg\min_{y \in C} < y, g_k^{FULL} >$
    $X_{k+1} = (1-\gamma_k)X_k + \gamma g_k$
end for
```

10.2.2 Convergence Analysis

The next question is if/how Frank-Walfe Algorithm converges. The following theorem gives the answer.

Theorem 10.1. *If f is $\beta-$smooth, $\gamma_k = \frac{1}{K+1}$, then*

$$f(X_T) - \min_{x \in C} f(x) \leq O(1)\frac{\beta R^2}{T+1}, \tag{10.3}$$

where $R = \sup_{x,y \in C} ||x-y||_2$.

Proof: For simplicity, let f^* and X^* denote the optimal function value and the optimal solution. From $\beta-$smooth, we have

$$f(X_{k+1}) - f(X_k) \leq < \nabla f(X_k), X_{k+1} - X_k > + \frac{\beta}{2}||X_{k+1} - X_k||^2. \tag{10.4}$$

Noting that $X_{k+1} = (1-\gamma_k)X_k + \gamma_k g_k$, we have

$$\begin{aligned} f(X_{k+1}) - f(X_k) &\leq \gamma_k < \nabla f(X_k), g_k - X_k > + \frac{\beta\gamma_k^2}{2}||g_k - X_k||^2 \\ &\leq \gamma_k < \nabla f(X_k), X^* - X_k > + \frac{\beta\gamma_k^2 R^2}{2} \\ &\leq \gamma_k(f^* - f(X_k)) + \frac{\beta\gamma_k^2 R^2}{2}, \end{aligned} \tag{10.5}$$

which implies

$$f(X_{k+1}) - f^* \leq (1-\gamma_k)f(X_k - f^*) + \frac{\beta}{2}\gamma_k^2 R^2. \tag{10.6}$$

Let $\gamma_k = \frac{1}{k+1}$. By induction, one can easily see that

$$f(X_{k+1}) - f^* \leq O(1)\frac{\beta R^2}{k+1}. \tag{10.7}$$

□

10.2.3 Revisiting an Example

How much computational cost does Frank-Wolfe Algorithm save compared to PGD? Although it is hard to answer in general, we give an example to have a taste. Still consider the convex set $C_{PSD(s)}$. Note that X is symmetric psd, so $\exists V$ such that $X = V^T V$. Thus,

$$\begin{aligned} &\min_{X \in C_{PSD(s)}} < X, Y > \\ =& \min_{\|V\| < \sqrt{s}} Tr < V^T Y V > \\ =& \min_{\|V\| < \sqrt{s}} Tr < V^T Y V > \\ =& \min_{\|V\| < \sqrt{s}} \sum_{i=1}^{n} V_i^T Y V_i. \end{aligned} \tag{10.8}$$

The problem is equivalent to find the maximum eigenvalue of Y, which requires only $O(n)$ computations. In other words, for $C_{PSD(s)}$, Frank-Wolfe Algorithm can save $O(n)$ computational cost compared to PGD.

10.2.4 Open Questions

One major disadvantages of Frank-Wolfe Algorithm is it requires computing the full gradient at each iteration. It is not acceptable in machine learning application when there are a large amount of data and it is expensive to compute the full gradient. One may take it as granted to develop a stochastic Frank-Wolfe Algorithm (SFWA) by replacing the full gradient with a sampled gradient, but it turns out to be difficult. So far, there is no known SFWA with one sample gradient computation per iteration and fast convergence guarantee. A related work is *Projection-free Online Learning* coauthored by Elad Hazan and Satyen Kale, where they developed SFWA for online learning. We refer interested readers to this paper and its follow-ups for more information.