Columbia University in the City of New York
Optimization Methods for Machine Learning
Instructors: Satyen Kale
Authors: Victor Ye Dong
Email: yd2470@columbia.edu

SCRIBE
18

Accelerated Gradient Descent

1 Problem Setup

We remmember from the previous classes that to solve the following problem

$$\begin{aligned} \min \quad & f(x) \\ s.t \quad & x \in R^d \end{aligned} \tag{1.1}$$

and assume $f(x)$ is β-smooth, define $D = ||x_0 - x^*||$ gradient descent can achieve a convergence rate of $\mathcal{O}(\beta D^2/T)$ and the Frank-Wolfe method also achieves the same convergence rate. This convergence rate is not the optimal one for first order methods, however. The optimal rate is achieved by Accelated Gradient Descent, invented by Nesterov in the 80's, which we will describe now.

From the analysis of gradient descent for β-smooth functions with step size $\frac{1}{\beta}$ we have, in any round t:

$$\begin{cases} x_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t) \\ f(x_{t+1}) \leq f(x_t) - \frac{1}{2\beta}||\nabla f(x_t)||^2 \end{cases} \tag{1.2}$$

then if $\nabla f(x_t)$ is large, the reduction will be large.

On the other hand, basic gradient analysis with step size η, gives us

$$\sum_{t=0}^{T-1} f(x_t) - f(x^*) \leq \frac{\eta}{2}\sum_{t=0}^{T-1} ||\nabla f(x_t)||^2 + \frac{D^2}{2\eta} \tag{1.3}$$

then if $\nabla f(x)$ is small, then the RHS will be small.

Of course both the above methods use different step sizes, and it's not clear which step size to use a priori. Thus, we want to come up with some method that blends the two methods. This is Nesterov's Accelerated Gradient Descent method, given below. Instead of one sequence of iterates x_t, it uses three sequences of points. The additional sequences are denoted y_t and z_t. The y_t sequence is updated using the step size $\frac{1}{beta}$, and the z_t sequence is updated using the step size η. x_t is obtained by taking a convex combination of the y_t and z_t sequences with a mixing parameter τ. The values of all of these parameters will be revealed in the analysis.

Algorithm 1: AGD

Start with arbitrary $x_0 = y_0 = z_0$. **for** *t = 0, 1, 2,..., T-1* **do**
  $y_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t)$
  $z_{t+1} = z_t - \eta\nabla f(x_t)$
  $x_{t+1} = \tau z_{t+1} + (1-\tau)y_{t+1}, \quad (\Leftrightarrow z_{t+1} = \frac{1}{\tau}(x_{t+1} - (1-\tau)y_{t+1})$
end
Output X_T

2 Algorithm Analysis

$$\begin{aligned} ||z_{t+1} - x^*||^2 &= ||z_t - \eta\nabla f(x_t) - x^*||^2 \\ &= ||z_t - x^*||^2 + \eta^2||\nabla f(x_t)||^2 - 2\eta\nabla f(x_t)(z_t - x^*) \end{aligned} \tag{2.1}$$

$\Rightarrow$

$$\begin{aligned} 2\eta\nabla f(x_t)(z_t - x^*) &\leq ||z_t - x^*||^2 - ||z_{t+1} - x^*||^2 + \eta^2||\nabla f(x_t)||^2 \\ &\leq ||z_t - x^*||^2 - ||z_{t+1} - x^*||^2 + 2\eta^2\beta(f(x_t) - f(y_{t+1})) \end{aligned} \tag{2.2}$$

$\Rightarrow$

$$\begin{aligned} 2\eta(f(x_t) - f(x_*)) &\leq 2\eta\nabla f(x_t)(x_t - x^*) \\ &\leq ||z_t - x^*||^2 - ||z_{t+1} - x^*||^2 + 2\eta^2\beta(f(x_t) - f(y_{t+1})) + 2\eta\nabla f(x_t)(x_t - z_t) \end{aligned} \tag{2.3}$$

Using the definition of z_t and convexity of f, we get

$$\begin{aligned} \nabla f(x_t)(z_t - x_t) &= \nabla f(x_t)(\frac{1-\tau}{\tau} \cdot (y_t - x_t)) \\ &\leq \frac{1-\tau}{\tau} \cdot (f(y_t) - f(x_t)) \end{aligned} \tag{2.4}$$

$\Rightarrow$

$$2\eta(f(x_t) - f(x^*)) \leq ||z_t - x^*||^2 - ||z_{t+1} - x^*||^2 + 2\eta^2\beta(f(x_t) - f(y_{t+1})) + 2\eta\nabla f(x_t)(\frac{1-\tau}{\tau} \cdot (f(y_t) - f(x_t))) \tag{2.5}$$

If we set $\eta\beta = \frac{1-\tau}{\tau} \Rightarrow \tau = \frac{1}{1+\eta\beta}$ then (2.5) becomes

$$2\eta(f(x_t) - f(x^*)) \leq ||z_t - x^*||^2 - ||z_{t+1} - x^*||^2 + 2\eta^2\beta(f(y_t) - f(y_{t+1})) \tag{2.6}$$

$\Rightarrow$

$$\begin{aligned} 2\eta T(f(\bar{x}) - f(x^*)) &\leq \sum_{t=0}^{T-1} 2\eta(f(x_t) - f(x^*)) \\ &\leq \sum_{t=0}^{T-1} ||z_t - x^*||^2 - \sum_{t=0}^{T-1} ||z_{t+1} - x^*||^2 + \sum_{t=0}^{T-1} 2\eta^2\beta(f(y_t) - f(y_{t+1})) \\ &\leq ||x_0 - x^*||^2 + 2\eta^2\beta(f(x_0) - f(x^*)) \end{aligned} \tag{2.7}$$

$\Rightarrow$

$$f(\bar{x}) - f(x^*) \leq \frac{1}{2\eta T}(||x_0 - x^*||^2) + \frac{\eta\beta}{T}(f(x_0) - f(x^*)) \tag{2.8}$$

Now suppose we know an upper bound on the initial suboptimality gap: $f(x_0) - f(x^*) \leq \triangle$, then

$$f(\bar{x}) - f(x^*) \leq \frac{D^2}{2\eta T} + \frac{\eta\beta\triangle}{T} \tag{2.9}$$

Letting $\eta = D\sqrt{\frac{1}{2\triangle\beta}}$, we have

$$f(\bar{x}) - f(x^*) \leq D\sqrt{\frac{2\beta\triangle}{T}} \tag{2.10}$$

Choose $T \geq \frac{8\beta D^2}{\triangle}$, we can achieve

$$D\sqrt{\frac{2\beta\triangle}{T}} \leq \frac{\triangle}{2} \tag{2.11}$$

So in the suboptimality gap drops to half of what it was initially. Now suppose we restart AGD with x_T as the initial point, and run it for $\frac{8\beta D^2}{\triangle/2}$ steps, then the gap decreases to $\triangle/4$. Repeating this restarting process to get the gap down to ϵ, the number of iterations needed is

$$\sqrt{\frac{8\beta D^2}{\triangle}} + \sqrt{\frac{8\beta D^2}{\triangle/2}} + \sqrt{\frac{8\beta D^2}{\triangle/4}} + \ldots, \sqrt{\frac{8\beta D^2}{\epsilon}} = \mathcal{O}\left(\sqrt{\frac{\beta D^2}{\epsilon}}\right).$$

Thus, the first order complexity of this restarted AGD method to get optimization error at most ϵ is $\mathcal{O}\left(\sqrt{\frac{\beta D^2}{\epsilon}}\right)$, which is significantly faster than either vanilla gradient descent or the Frank-Wolfe method.

If that the function f is α-strongly convex in addition to being β-smooth, then

AGD achieves error $\mathcal{O}(e^{-\sqrt{\frac{\alpha}{\beta}}T})$ after T gradient calls $\Rightarrow$ to get ϵ suboptimality gap we need $O(\sqrt{\frac{\beta}{\alpha}}\log\left(\frac{1}{\epsilon}\right))$ gradient calls (note that this bound ignores some lower order logarithmic dependence on D^2). This analysis is based on bootstrapping the previous analysis and can be found in CMH [https://ee227c.github.io/notes/ee227c-lecture08.pdf section 8.2].

In contrast, *GD* achieves error $\mathcal{O}(e^{-\frac{\alpha}{\beta}T})$ after T gradient calls $\Rightarrow$ to get ϵ suboptimality gap we need $O(\frac{\beta}{\alpha}\log\left(\frac{1}{\epsilon}\right))$ gradient calls.