EE C227C Convex Optimization and Approximation Lecture 4 - 1/27/2022

Subgradient Descent Method

Lecturer: Jiantao Jiao Scribe: Arwa AlAnqary, Philip Canoza, Christian Ikeokwu, Mingrui Zhang

1 Intro and Motivation

Goal Continue exploring first-order methods and improve upon the gradient descent algorithm. Build towards understanding Mirror Descent by first learning about subgradient methods. These methods have deep connections to gradient descent but it took people decades to figure this out.

Last time Analyzed the Gradient Descent (GD) algorithm under two assumptions

1. $f(x)$ is convex
2. Lipschitz gradient (*aka* β-smooth or L-smooth) i.e $\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$ where $\|\cdot\|$ is the Euclidean norm.

Under these two assumptions we showed that the convergence rate of GD is "good". Specifically we showed that

$$f(x_k) - f^* \lesssim \frac{1}{k}$$

though not proved in the previous class this bound is tight for gradient descent [Nemirovski, 1995]. However, the optimal rate for this family of functions is $O(1/k^2)$ [Nesterov, 1983]. For strongly convex functions we showed that GD converges exponentially fast $\left(\frac{L}{2}\left(\frac{Q-1}{Q+1}\right)^{2k}\|x_0 - x^*\|^2\right)$. We also showed that GD is a descent algorithm i.e $f(x_{k+1}) < f(x_k)$.

Motivation Certain applications or regimes might not have Lipschitz gradients (Assumption 2.).

Example 1 (Minimizing $|x|$)**.**
Lets say we wanted to minimize $|x|$ over $\mathbb{R}$ or some finite interval like $[-1,1]$. This function is convex i.e it satisfies

Definition 2 (General (non-smooth) convex function)**.** *f is convex iff it satisfies, for any $x, y \in K, \gamma \in [0,1]$*

$$f((1-\gamma)x + \gamma y) \leq (1-\gamma)f(x) + \gamma f(y) \tag{1}$$

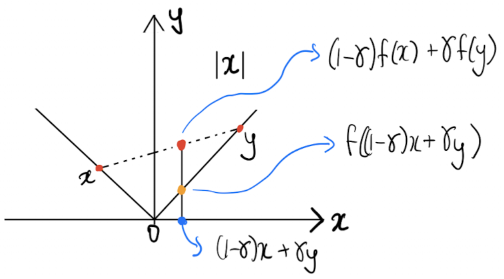

Figure 1: $|x|$ is convex

In addition, when $x > 0$ our function is differentiable with $\frac{df(x)}{dx} = 1$ and when $x < 0$ its also differentiable with $\frac{df(x)}{dx} = -1$. However, this function clearly violates the Lipschitz gradient assumption, because around 0 even when you move infinitesimally small the gradient changes sharply from -1 to 1 or vice versa. Yet, the function itself is still Lipschitz ($|f(x) - f(y)| \leq |x - y|$) and is important with numerous applications like in neural nets with ReLu activation functions.

Subgradients Given the problems highlighted by Example 1, how can we minimize functions that are Lipschitz but not necessarily Lipschitz gradient? When GD style algorithms were first proposed the main intuition behind their performance was the idea of descending at every step using the gradient. Unfortunately, for this class of functions there are results showing that *descent cannot be established* which seems like a big issue. However, because these functions are still convex there exist a useful generalization of the gradient known as the *subgradient*

Definition 3 (Subgradient)**.** *Let $K \subseteq \mathbb{R}^n$, $f : K \mapsto \mathbb{R}$. Then $g \in \mathbb{R}^n$ is a subgradient of f at $x \in K$ if for any $y \in K$ we have*

$$f(x) - f(y) \leq \langle g, x - y\rangle = g^T(x - y) \iff f(y) \geq f(x) + \langle g, y - x\rangle \tag{2}$$

Compare this with our definition of convexity for continuously differentiable (smooth) functions

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle$$

These expressions are very similar except with the gradients swapped out for some function g. Geometrically they capture the same ideas, they are both affine expression for a hyperplane that intersects the function at x and serves as a global lowerbound for the rest of f. In most proofs however, the first inequality from Definition 3 is more useful because the main goal is often to upper bound the sub-optimality of a point. For example, we might take x to be the best output of our method and y to be some reference value, say the global optimal point x^*, so this inequality is useful for upper bounding the sub-optimality of our point. These inequalities are also useful in online learning.

Example 4 (Subgradients of $|x|$)**.**
It is a fact that if a function has subgradients everywhere then the function must be convex and vice versa. However there can be multiple subgradients at a point so we define the subdifferential which is a set.

Definition 5 (Subdifferential)**.** *For functions f with subgradients, the subdifferential of f at x given by $\partial f(x)$ is a set,*

$$\partial f(x) = \{g | g \textit{ is a subgradient at } x \textit{ for } f\} \tag{3}$$

Thus we can define the subgradients of $f(x) = |x|$ as follows.

$$\partial f(x) = \begin{cases} \{1\} & x > 0 \\ [-1, 1] & x = 0 \\ \{-1\} & x < 0 \end{cases}$$

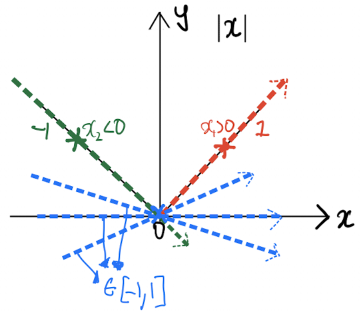

Figure 2: Subgradients of $|x|$

so now our goal is to use the subgradient to optimize over our function using the *Subgradient Descent* algorithm.

2 The Subgradient Descent Algorithm

The optimization problem The goal is to provide an algorithm that can solve the following optimization problem

$$\min_x \quad f(x) \tag{4}$$
$$\text{s.t.} \quad x \in K \tag{5}$$

with the following assumptions on the objective function:

1. $f(x)$ is convex
2. $f(x)$ is not necessarily continuously differentiable
3. $f(x)$ is a Lipschitz function
4. K is a closed bounded convex set

Using the definition of the sub-gradient $g(x_i)$ we can devise an algorithm with the following recursion

$$x_{i+1} = \Pi_K \left(x_i - \gamma_i \frac{g(x_i)}{\|g(x_i)\|_2} \right) \tag{6}$$

where:

1. $K \subset \mathbb{R}^n$ is the feasible set of the optimization problem.
2. $\Pi_K := \text{argmin}\{\|x - y\|_2 | y \in K\}$ is function that projects any point in $\mathbb{R}^n$ onto the nearest point in K.

Termination Criteria The algorithm terminates successfully when we find a point $x^* \in K$ such that $\mathbf{0} \in \partial f(x^*)$.

Algorithm Properties The sub-gradient descent algorithm is not actually a descent algorithm, i.e. we can't guarantee that $f(x_{i+1}) \leq f(x_i)$. One implication of this is that the point in the final iteration might not be the best point. The solutions to this:

1. Track the function value and output the best iteration so far.
2. Output the weighted search point. [1]

3 Analyzing the Subgradient Descent Algorithm

In this part, we aim to analyze the best point up to step N, in the sense of minimum value in f. First, we introduce some notations. Let $\overline{x}_i$ denote the best point up to step i, satisfying $\overline{x}_i \in \{x_1, ..., x_i\}$ and

$$f(\overline{x}_i) \leq f(x_j), \qquad \forall 1 \leq j \leq i.$$

Let $r_i = \|x_i - x^*\|_2$ and $\epsilon_i = f(\overline{x}_i) - f^*$. Suppose that f is an L-Lipschitz function with respect to the Euclidean norm, i.e. $|f(x) - f(y)| \leq L\|x - y\|_2$ for all x and y in K. We claim

$$\frac{2\gamma_i \epsilon_i}{L} \leq r_i^2 - r_{i+1}^2 + \gamma_i^2. \tag{7}$$

By definition, it is immediately followed that ϵ_i is non-increasing, thus the summation of (7) over $i = 1, ..., N$ implies

$$\epsilon_N \leq \frac{L}{2}\left(\frac{\|x_1 - x^*\|_2^2 + \sum_{i=1}^N \gamma_i^2}{\sum_{i=1}^N \gamma_i}\right) \tag{8}$$

To derive (7), we review some facts in mathematics.

Lemma 6 (Lemma 8.1.1 in the Georgia Tech lecture notes)**.** *Let $x \in \mathbb{R}^n$, and let K be a closed convex subset in $\mathbb{R}^n$. Under projection onto K, x becomes closer to any point u of K, namely, the squared distance from x to u decreases at least by the squared distance from x to K:*

$$\|\Pi_K(x) - u\|_2^2 \leq \|x - u\|_2^2 - \|x - \Pi_K(x)\|_2^2 \leq \|x - u\|_2^2$$

Lemma 7. *Let $f : K \to \mathbb{R}$ be an L-Lipschitz function with respect to the Euclidean norm. Then, for all $x \in K$ and its sub-gradient $g(x)$, we have $\|g(x)\|_2 \leq L$.*

Based on the two lemmas, (7) is from the following inequalities

$$\begin{aligned}
r_{i+1}^2 &= \|x_{i+1} - x^*\|_2^2 \\
&= \left\|\Pi_K\left(x_i - \gamma_i \frac{g(x_i)}{\|g(x_i)\|_2}\right) - x^*\right\|_2^2 \\
&\leq \left\|x_i - \gamma_i \frac{g(x_i)}{\|g(x_i)\|_2} - x^*\right\|_2^2 \\
&= \|x_i - x^*\|_2^2 - 2\gamma_i \frac{\langle g(x_i), x_i - x^*\rangle}{\|g(x_i)\|_2} + \gamma_i^2 \\
&\leq r_i^2 - \frac{2\gamma_i \epsilon_i}{\|g(x_i)\|_2} + \gamma_i^2 \\
&\leq r_i^2 - \frac{2\gamma_i \epsilon_i}{L} + \gamma_i^2
\end{aligned}$$

where the first inequality is by Lemma 6, the second inequality is by the definition of a sub-gradient, and the third inequality is by Lemma 7.

[1] Definition of the weighted search point will follow in future lectures

4 Choosing Step Sizes: γ_i

Using results from real analysis, we can recognize that if

$$\sum_{i=1}^{\infty} \gamma_i := \infty, \gamma_i \to 0, i \to \infty$$

then our bounding term always goes to zero.
With this in mind, we can choose a step-size by optimizing for γ_i by fixing N and minimizing the RHS, we get

$$\gamma_i = \frac{\|x_1 - x^*\|}{\sqrt{N}}$$
$$\Rightarrow \epsilon_N \leq L \frac{\|x_1 - x^*\|}{\sqrt{N}}$$

This has a nice interpretation. Notice how the accuracy ϵ_N is proportional to L, as well as the initial guess's distance from optimum $\|x_1 - x^*\|$. It is also inversely proportional to $N^{1/2}$, and so converges slower than that of GD, which was on order k. We note that this is hard to use in practice since we don't know x^* or the horizon N. So, we propose alternative forms based on different assumptions.

- Assume $\|x_1 - x^*\| \leq D$, i.e. we assume the initial guess is some distance from the optimal set. Then we have $\gamma_i = \frac{D}{\sqrt{N}}$, and

$$\Rightarrow \epsilon_N \leq \frac{LD}{\sqrt{N}}$$

- Instead, we could have $\gamma_i = \frac{D}{\sqrt{i}}$, which is popular in empirical practice.

$$\Rightarrow \epsilon_N \lesssim L \frac{D \log N}{\sqrt{N}}$$

  Aside: Note that the $\sqrt{N}$ comes from the fact $\sum_{i=1}^{N} \gamma_i^2 = \sum_{i=1}^{N} \frac{1}{\sqrt{i}} = \Theta\left(\sqrt{N}\right)$. We also note that the $\log N$ term is purely an analysis artifact. This comes from $\gamma_i = \frac{D}{\sqrt{i}} \Rightarrow \gamma_i^2 = \frac{D^2}{i}$, and so we have $\sum_{i=1}^{N} \gamma_i^2 = \Theta(\log N)$. Actually if we use a different way to analyze we would not have this additional logarithmic factor. We start from step M:

$$\begin{aligned} 2\epsilon_N \sum_{i=M}^{N} \gamma_i &\leq L \left(r_m^2 + \sum_{i=M}^{N} \gamma_i^2 \right) \\ &\leq L \left(D^2 + \sum_{i=M}^{N} \gamma_i^2 \right) \\ \epsilon_N &\leq \frac{L}{2} \left(\frac{D^2 + \sum_{i=M}^{N} \gamma_i^2}{\sum_{i=M}^{N} \gamma_i} \right) \end{aligned}$$

  where $M \leq N$. We can choose $M = \lfloor \frac{N}{2} \rfloor$ and $\gamma_i = \frac{D}{\sqrt{i}}$. Now we have

$$\sum_{i=\lfloor \frac{N}{2} \rfloor}^{N} \gamma_i^2 = \Theta(1)$$
$$\Rightarrow \epsilon_N \lesssim \frac{LD}{\sqrt{N}}$$

  This rate is actually optimal for this family of convex and Lipschitz functions.

In reality, there is an even better step size choice.

4.1 Boris Polyak Step Size

Recall our analysis from Section 3. We used the inequality

$$r_{i+1}^2 \leq r_i^2 - 2\gamma_i \frac{f(x_i) - f^*}{\|g(x_i)\|} + \gamma_i^2$$

where we upper bounded the numerator of the linear term by ϵ_i, and lower bounded the denominator by L. Instead, we can recognize that this is just a quadratic function of γ, so we can solve

$$\begin{aligned}\arg\min_{\gamma_i} &\left(r_i^2 - 2\gamma_i \frac{f(x_i) - f^*}{\|g(x_i)\|} + \gamma_i^2\right)\\ &= \frac{f(x_i) - f^*}{\|g(x_i)\|}\end{aligned}$$

This is the **Polyak learning rate**.
In practice, we may have to guess f^*. Estimation can be highly empirical, but one special case can be the following: Find x s.t.

$$\begin{aligned} f_i(x) &\leq 0, \qquad 1 \leq i \leq m \\ \Rightarrow \min_x & \max_{1 \leq i \leq m} f_i(m) \\ f^* &= 0 \end{aligned}$$

Irregardless, if you have this info, you can use this step size and get a convergence rate of

$$\Rightarrow \epsilon_N \leq \frac{L\|x_1 - x^*\|}{\sqrt{N}}$$

Remark: We can see that it is proportional to the distance of the guess from the optimal set x^*. So if your initial guess is close, you converge faster.

5 Concluding thoughts

A more thorough treatment can be found in Lecture 8 of https://www2.isye.gatech.edu/~nemirovs/Lect_EMCO.pdf This subgradient method analysis raised several questions. Why can you guarantee some point is good? Why is a small gradient size good here, but large gradient size is preferred in Gradient Descent?. To answer this, we will dive into deeper theory in the following lectures by analyzing Mirror Descent.

References

Arkadi Nemirovski. Information-based complexity of convex programming. *Lecture Notes*, 834, 1995.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.