10-725: Convex Optimization **Spring 2023**

Lecture 5: January 31

Lecturer: Siva Balakrishnan

In the last lecture we discussed subgradients, and today we will finally discuss the subgradient method. It is common to use the term subgradient method (instead of subgradient descent) since often the method is not a descent method (i.e. in most cases where we apply the method, and for reasonable choices of the step-size, function values can go up between iterations).

5.1 Subgradient Method

Exactly like gradient descent, but we replace gradients by subgradients, i.e. we initialize at x^0, and iterate:

$$x^{t+1} = x^t - \eta_t g_{x^t},$$

where $g_{x^t} \in \partial f(x^t)$, is any subgradient of f at x^t.

Since it often will not be a descent method, we'll usually keep track of the best iterate found so far, and output:

$$x^{\text{best}} = \arg \min_{t \in \{0,\ldots,k\}} f(x^t).$$

It's a bit beyond the scope of this course so I'll just mention this in passing:

1. Subgradient directions need not be descent directions, i.e. they can be directions such that for any step-size $\eta > 0$, the function value increases. We already know the example of $f(x) = |x|$ where at the point 0, -1 (for instance) is a valid subgradient, but moving in that direction will only increase the objective function.

   A slightly more interesting example is to take the function $f(x) = |x_1| + 10|x_2|$, then at the point $(1, 0)$ (which is not the minimizer), the vector $(1, 10)$ is a valid subgradient, but for any $\eta > 0$ our next iterate would be at $(1 - \eta, -10\eta)$ which would be a strictly worse point (i.e. the negative subgradient direction is a direction of increase for the function).

2. It will turn out to be the case that if you were very clever in your choice of subgradient (and particularly, picked a direction in $\partial f(x^t)$ which has minimum ℓ_2 norm) then these special subgradient directions are descent directions, and one could hope to design a subgradient descent method (i.e. by carefully choosing the step-sizes).

3. Finding the full sub-differential and then finding the minimum norm element is hard in a lot of examples. So our goal will instead be to show that a naive algorithm that simply follows an arbitrary subgradient at each iteration will in fact converge to a globally optimal solution, under some assumptions, provided we are careful about step-size choice.

5.2 Step Sizes

One of the basic differences between GD and the subgradient method is that we typically use different methods for choosing the step-size.

Since, as we discussed above subgradient directions need not be descent directions, in practice people don't usually use adaptive step-sizes (i.e. backtracking line searches) since those are based on finding step-sizes which ensure "sufficient descent".

Instead people typically use:

1. **Fixed Step-Sizes:** If we're going to run k iterations of the subgradient method, one can use a fixed step-size η for all iterations. As will be more clear when we look at some convergence proofs a typical choice will be $\eta \sim 1/\sqrt{k}$, i.e. if we're going to do many steps we'll take a smaller step-size.
2. **Step-Size Schedules:** In practice, it is more common to use decaying step sizes. For reasons that will be clear in a little while, we usually choose the step-sizes $\eta_t \to 0$ so that two conditions are satisfied:

$$\sum_{t=0}^{\infty} \eta_t = \infty \tag{5.1}$$

$$\sum_{t=0}^{\infty} \eta_t^2 < \infty. \tag{5.2}$$

5.3 Lipschitz Functions

To analyze the subgradient method we'll assume that our objective function is G-Lipschitz, i.e. for any x, y

$$|f(x) - f(y)| \leq G\|x - y\|_2.$$

A more useful (for us) equivalent characterization for convex functions f will be that for any $g \in \partial f(x)$,

$$\|g\|_2 \leq G.$$

We are not assuming that the function is smooth (or even differentiable), rather we are assuming that the subgradients are bounded. Notice that all the non-smooth functions we've encountered so far (things like the absolute value function) satisfy this assumption.

We can check one direction of this equivalence: by convexity, we know that for any y,

$$g_x^T(x-y) \leq f(x) - f(y),$$

so choosing $y = x - g_x$, we see that,

$$\|g_x\|_2^2 \leq f(x) - f(x-g_x) \leq |f(x) - f(x-g_x)| \leq G\|g_x\|_2.$$

5.4 Convergence of Subgradient Method

Our goal will now to be to prove the following theorem:

Theorem 5.1 *Suppose f is convex, and G-Lipschitz. Denote by x^{best} the best iterate of the subgradient method amongst $x^0, \ldots, x^k$. Then:*

1. *If the step-size is chosen to be a constant $\eta = R/(G\sqrt{k})$*

$$f(x^{best}) - f(x^*) \leq \frac{GR}{\sqrt{k}}.$$

2. *For any sequence of step-sizes, which satisfy the conditions in* (5.1), (5.2) *we have that,*

$$f(x^{best}) - f(x^*) \to 0, \quad \text{as } k \to \infty.$$

Proof: The proof is similar to our earlier proofs so hopefully you're getting the hang of the manipulations.

$$\begin{aligned}\|x^{t+1} - x^*\|_2^2 &= \|x^t - x^*\|_2^2 + \eta_t^2\|g_{x^t}\|_2^2 - 2\eta g_{x^t}^T(x^t - x^*) \\ &\leq \|x^t - x^*\|_2^2 + \eta_t^2 G^2 - 2\eta_t g_{x^t}^T(x^t - x^*).\end{aligned}$$

By convexity we know that,

$$f(x^*) \leq f(x^t) + g_{x^t}^T(x^* - x^t),$$

and as a consequence we obtain,

$$\|x^{t+1} - x^*\|_2^2 \leq \|x^t - x^*\|_2^2 + \eta_t^2 G^2 + 2\eta_t(f(x^*) - f(x^t)).$$

Summing from $t = \{0, \ldots, k-1\}$ and dropping a negative term we obtain that,

$$2\sum_{t=0}^{k-1} \eta_t (f(x^t) - f(x^*)) \leq \|x^0 - x^*\|_2^2 + G^2 \sum_{t=0}^{k-1} \eta_t^2.$$

Recalling our definition of the best iterate seen we see that,

$$f(x^{\text{best}}) - f(x^*) \leq \frac{\|x^0 - x^*\|_2^2 + G^2 \sum_{t=0}^{k-1} \eta_t^2}{2\sum_{t=0}^{k-1} \eta_t}.$$

From this expression we can directly see all of our conclusions. Suppose that $\|x^0 - x^*\|_2 \leq R$.

1. If we choose a constant step-size, $\eta_t = R/(G\sqrt{k})$ then we see that,

$$f(x^{\text{best}}) - f(x^*) \leq \frac{GR}{\sqrt{k}}.$$

2. More generally, for any sequence of step-sizes which satisfy (5.1), (5.2) we have that $f(x^{\text{best}}) - f(x^*) \to 0$ (which is some type of "convergence" of the method).

3. Finally, if we don't know the number of steps in advance (and we don't know the parameters R and G) we might choose, $\eta_t = 1/\sqrt{t}$, to obtain a similar guarantee,

$$f(x^{\text{best}}) - f(x^*) \lesssim \frac{G^2 \log(k) + R^2}{\sqrt{k}}.$$

   This is worse than the fixed choice (by a log factor) but allows one to iterate beyond an a-priori fixed number of iterations.

■

5.5 Lower Bounds

We will not spend too much time on this but will briefly mention lower bounds – at least in part because these lower bounds motivate several algorithmic ideas (including the proximal method, and accelerated methods).

At this point we have seen several different convergence rates under different assumptions (for both gradient descent and the subgradient method), and a natural question is whether these rates are in any sense the best we can hope for.

5.5.1 Oracle Model

Formalizing what precisely we mean by lower bounds could be done in many ways. One is based on the so-called oracle model of computation – here we are only allowed to access the function we are trying to optimize via an oracle. Now, one could imagine several possible oracles:

1. A zeroth-order oracle simply returns function values, i.e. the optimizer submits a query point x, and the oracle returns the value $f(x)$.
2. A first-order oracle returns both function and subgradient values, i.e. for a query point x, the oracle returns a pair (x, g_x) where $g_x \in \partial f(x)$.
3. A second-order oracle returns function, gradient, and Hessian values, and so on.

Since, we've focused so far on first-order methods, we'll suppose we have access to a first-order oracle. Now the basic question is can we *lower bound* the number of calls to the oracle we need in the *worst-case* to guarantee suboptimality ϵ, i.e. $f(\widehat{x}) - f(x^*) \leq \epsilon$?

A black-box optimization algorithm starts from an arbitrary initial point x^0, and maps the history $\{x^0, g_0, x_1, g_1, \ldots, x_{t-1}, g_{t-1}\}$ (where each $g_i \in \partial f(x_i)$) to a new query point x_t.

We will further restrict the class of algorithms to be of the following form: they begin at $x^0 = 0$, and at each time t, select a new query point which is in the span of subgradients $x_t \in \{g_0, g_1, \ldots, g_{t-1}\}$. The algorithms we've looked at (things like GD and the subgradient method) all follow this general template. (The results are more generally true.)

A proof of the following lower bounds can be found in Bubeck's textbook (they follow simply by constructing a particular bad function which is hard to optimize).

Theorem 5.2 *Suppose that $t \leq (d-1)/2$, then:*

1. *There is a convex function f, which is L-Lipschitz, for which:*

$$\min_{1 \leq s \leq k} f(x^s) - \min_{\|x\|_2 \leq R} f(x) \gtrsim \frac{RL}{\sqrt{k}}.$$

2. *There is a β-smooth convex function f, for which:*

$$\min_{1 \leq s \leq k} f(x^s) - f(x^*) \gtrsim \frac{\beta \|x^0 - x^*\|_2^2}{k^2}.$$

3. *There is a β-smooth, α-strongly convex function f for which:*

$$\min_{1 \leq s \leq k} f(x^s) - f(x^*) \gtrsim \alpha \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{2(k-1)} \|x^0 - x^*\|_2^2.$$

1. Notice the requirement that $t \leq (d-1)/2$ (i.e. the number of iterations has to be small relative to the dimension). This is fine provided we're interested in understanding the fundamental limits of "dimension-free" convex optimization algorithms. (If you're willing to pay a dimension dependent penalty, then rates of convergence under almost all of our settings will be geometric, i.e. you can often get rates of convergence which are of the form $d \log(1/\epsilon)$ using methods like the ellipsoid algorithm which are not dimension-free.)

2. The main takeaways from the above result are that the subgradient method is slow, but optimal for the class of convex, Lipschitz functions. However, GD is (potentially) sub-optimal for both smooth functions (it gets $1/k$ instead of $1/k^2$ rates), and for smooth and strongly convex functions where the dependence on κ is better in the lower bound (which has dependence on $\sqrt{\kappa}$ instead of κ).

   It will turn out that both the latter two gaps can be closed using a method known as *accelerated* gradient descent which we might cover later in the course.

   On the other hand we'll focus in the next lecture on understanding special subclasses of non-smooth problems where we can improve on the subgradient method. The lower bound says that we shouldn't expect to do this without some additional structure, and we'll explore some interesting problem structure (and discuss *proximal* algorithms which can exploit this structure in the next class).