Lecture 2

Lecturer: Jonathan Kelner *Scribe: Jonathan Weed*

1 Overview

Last class, we talked about gradient descent as a general philosophy for solving optimization problems. The motivation for this approach comes from the simplest possible case: finding $\min_{x \in \mathbb{R}^n} f(x)$ when we assume that f is differentiable and convex. The assumption that f is differentiable allows us to speak sensibly about the gradient ∇f at any point, and the assumption that f is convex allows us to conclude that $\nabla f = 0$ is a sufficient condition for global optimality.

A reasonable algorithm for this simple case is the following: initialize x_1 to some arbitrary point in $\mathbb{R}^n$, and iteratively set $x_{x+1} = x_i - \eta \nabla g(x_i)$ for some $\eta > 0$. This algorithm is morally sound: the vector $-\nabla g(x_i)$ points in the direction of steepest descent, so we hope to make nontrivial progress towards a global optimum at every step.

Of course, this "algorithm" leaves a lot of questions unanswered. (How should we choose η? Can we guarantee convergence to an optimal solution?) More troublingly, this algorithm can't be applied to the maximum flow problem out of the box. For one, our objective function in the maximum flow case is not differentiable, so it's not clear what object we should be considering.

Today, we will generalize this sketch of a gradient descent algorithm in two important ways: we will consider non-differentiable objective functions and will allow our solution space to be constrained. (It turns out we cannot relax the convexity assumption, since that is what makes our problem tractable at all.) The resulting algorithm, called Projected Subgradient Descent, will be our first working algorithm for maximum flow.

2 Review of the General Framework

As always, our goal is to try to solve the maximum flow problem on an undirected, unit-capacity graph. However, we choose to view this problem as a general convex optimization problem instead. We'll use this perspective to develop algorithms that match the performance of the best-known combinatorial approaches without having to take into account any particular graph-theoretic structure of the maximum flow problem. Later, when we incorporate graph-theoretic insights, we'll be able to beat combinatorial algorithms.

By rescaling by the value of the flow, we can recast the maximum flow problem as the following minimum congestion problem:

$$\begin{aligned} &\min \|f\|_\infty \\ &\text{such that } f \in \mathcal{F}_{s,t}, \end{aligned}$$

where $\mathcal{F}_{s,t}$ is the (affine) subspace of $\mathbb{R}^n$ corresponding to the set of valid s-t flows of value 1.

As noted above, we can't apply gradient descent to this problem for two reasons. First, $\|f\|_\infty$ is not differentiable, so it is not clear how to define a quantity that plays the role of the gradient. Second, even if $\|f\|_\infty$ did have a gradient, following that gradient would likely lead us outside of $\mathcal{F}_{s,t}$, the space of feasible solutions.

We can solve both of these problems in a natural way. To replace the gradient, we appeal to the *subgradient*, a generalization of the gradient that exists for all convex functions. To respect the constraint, we will push our candidate solution back into $\mathcal{F}_{s,t}$ at each step.

Are there other ways to solve these two problems? Certainly. We might imagine replacing the function $\|f\|_\infty$ by some smooth approximation. We might choose to rewrite the constraint as a penalty

in the objective function instead. We could even try to choose a basis for the affine subspace $\mathcal{F}_{s,t}$ and solve an unconstrained optimization problem inside this subspace. In fact, all of these ideas are good ones, and the best algorithms we'll see in this course will use all three.

3 Projected Subgradient Descent

We begin by recalling the notion of a subgradient.

Definition 1 *For $f : \mathbb{R}^n \to \mathbb{R}$, a vector $s \in \mathbb{R}^n$ is a* subgradient *at x (notation: $s \in \partial f(x)$) if*

$$f(x) - f(y) \leq s^\top (x - y) \tag{1}$$

for all $y \in \mathbb{R}^n$.

As noted last time, convex functions have subgradients everywhere. When f is differentiable at x, its gradient $\nabla f(x)$ is a subgradient (indeed, the only subgradient) and is a good linear approximation to f; when f is not differentiable, a subgradient still provides a linear lower bound.

We will also need a projection operator to return our steps to the constraint set.

Definition 2 *Let $\mathcal{K} \subseteq \mathbb{R}^n$ be convex and compact. Then the* projection $\Pi_{\mathcal{K}} : \mathbb{R}^n \to \mathcal{K}$ *is given by the rule*

$$\Pi_{\mathcal{K}}(x) = \operatorname{argmin}_{y \in \mathcal{K}} \|y - x\|,$$

where $\|\cdot\|$ is the ℓ^2 norm.

Projection is a well-behaved operation when $\mathcal{K}$ is a convex set. We will depend on two properties, one obvious and the other not: that $\Pi_{\mathcal{K}}$ is the identity on $\mathcal{K}$, and that $\Pi_{\mathcal{K}}$ is a contraction. Formally, we prove the following lemma.

Lemma 3 *Let $x \in \mathcal{K}$, $y \in \mathbb{R}^n$. Then the following hold:*

1. $\Pi_{\mathcal{K}}(x) = x$
2. $(\Pi_{\mathcal{K}}(y) - x)^\top (\Pi_{\mathcal{K}}(y) - y) \leq 0$.
3. $\|\Pi_{\mathcal{K}}(y) - x\|^2 + \|y - \Pi_{\mathcal{K}}(y)\|^2 \leq \|y - x\|^2$.

The first claim is obvious, so we content ourselves with proving the second two.

Proof [Proof of Claims 2 and 3] We provide a "proof by picture." (See Figure 1.) The line joining the point y to $\Pi_{\mathcal{K}}(y)$ is perpendicular to a line tangent to $\mathcal{K}$, and any point $x \in \mathcal{K}$ lies on the only side of the tangent. Therefore the angle between the vectors $(\Pi_{\mathcal{K}}(y) - x)$ and $(\Pi_{\mathcal{K}}(y) - y)$ is obtuse. Claim 2 and Claim 3 follow. ■

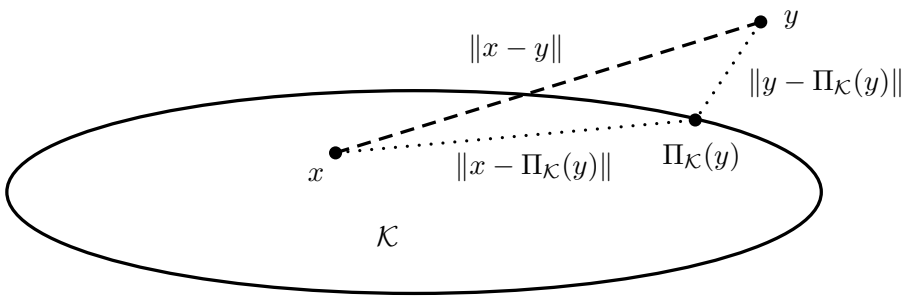

Figure 1: Proof of Lemma 3.

Finally, to provide provable bounds on the efficacy of our algorithm, we make the following two assumptions, which capture the "scale" of our problem instance:

- The set $\mathcal{K}$ is contained in $\mathrm{B}(x, R)$, the ball of radius R around some point $x \in \mathcal{K}$
- There is a constant G such that for all $x \in \mathcal{K}, g \in \partial f(x)$, the bound $\|g\| \leq G$ holds.

Two remarks are in order about the second assumption. First, the constant G is in fact a Lipschitz constant for the function f. Indeed, the bound $\|g\| \leq G$ implies that $\|f(x) - f(y)\| \leq G\|x - y\|$ for all y. If g were a gradient, we could deduce this by integrating, and a similar but more complicated argument suffices when g is a subgradient. Second, note that such a G always exists since $\mathcal{K}$ is compact by assumption; however, projected subgradient descent does not work well when G is extremely large.

The full Projected Subgradient Descent algorithm appears as Algorithm 1.

Algorithm 1 Projected Subgradient Descent

$x_1 \leftarrow x$ for some $x \in \mathcal{K}$
for $s = 1 \dots T - 1$ **do**
  $y_{s+1} \leftarrow x_s - \eta g_s$ for some $g_s \in \partial f(x_s)$
  $x_{s+1} \leftarrow \Pi_{\mathcal{K}}(y_{s+1})$
end for
return $\bar{x}_T \leftarrow \frac{1}{T} \sum_{s=1}^{T} x_s$

It remains to specify η, and to show that this algorithm actually finds an optimum. Theorem 4 achieves both goals.

Theorem 4 *When $\eta = \frac{R}{G\sqrt{T}}$, then the output of Algorithm 1 satisfies*

$$f(\bar{x}_T) - f(x^*) \leq \frac{RG}{\sqrt{T}}, \tag{2}$$

where $x^* = \operatorname{argmin}_{x \in \mathcal{K}} f(x)$ is the true optimum, and $\bar{x}_T = \frac{1}{T} \sum_{s=1}^{T} x_s$.

Note that the bound of Equation 2 appears to be dimension-free. However, in practice the constant G will depend on the dimension, since it is a maximum of Euclidean norms. We will prove Theorem 4 by appealing to a basic identity and using the definition of the subgradient. Lemma 3 guarantees that the projection step does not ultimately affect our progress towards the goal.

Proof Using the identity $2a^\top b = \|a\|^2 + \|b\|^2 - \|a - b\|^2$, we obtain:

$$\begin{aligned} f(x_s) - f(x^*) &\leq g_s^\top (x_s - x^*) \\ &= \frac{1}{\eta}(x_s - y_{s+1})^\top (x_s - x^*) \\ &= \frac{1}{2\eta}\left(\|x_s - x^*\|^2 + \|x_s - y_{s+1}\|^2 - \|y_{s+1} - x^*\|^2\right) \\ &= \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2\right) + \frac{\eta}{2}\|g_s\|^2. \end{aligned}$$

Note that $\|y_{s+1} - x^*\|^2 \geq \|x_{s+1} - x^*\|^2$, by Lemma 3, and $\|g_s\| \leq G$, by assumption.

The resulting bound is

$$f(x_s) - f(x^*) \leq \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2\right) + \frac{\eta}{2}G^2.$$

Summing the result on s, we get a telescoping sum:

$$\sum \left(f(x_s) - f(x^*)\right) \leq \frac{R^2}{2\eta} + \frac{\eta}{2}G^2 T,$$

where we have used the fact that $\|x_1 - x^*\| \leq R$.

The bound follows upon dividing by T, optimizing η, and noting that $f(\bar{x}_T) = f\left(\frac{1}{T}\sum_{s=1}^{T} x_2\right) \leq \frac{1}{T}\sum_{s=1}^{T} f(x_s)$, since f is convex.■

Without additional assumptions on the properties of the function f, Algorithm 1 is optimal. In the context of maximum flow, it is not hard to see (exercise!) that $R = \sqrt{n}$ and $G = \sqrt{m}$. The projection step appears to require slow linear algebra, but it turns out that this particular problem can be solved elegantly using electrical flows. The biggest problem is the factor of $1/\sqrt{T}$, which is a terrible dependence on the desired level of precision.

4 Gradient Descent for Smooth Functions

Algorthm 1 cannot be improved without better control over the function f. Our primary problem is knowing how much to "trust" the information of a gradient—if it represents a good linear approximation to the function, then we can follow it farther than if it does not. It turns out that controlling how fast the gradient changes gives Algorithm 1 significantly better performance.

Definition 5 *A function f is* L-smooth *if*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

for all x and y.

We have used a gradient symbol in the definition of L-smooth since such functions are always differentiable. In particular, the sup norm $\|\cdot\|_\infty$ is *not* L-smooth.

Suppose that we use Algorithm 1 to optimize a L-smooth function f over all of $\mathbb{R}^n$, using $\eta = \frac{1}{L}$.

Claim 6 *With the above setup, the output of Algorithm 1 satisfies the bound*

$$f(x_t) - f(x^*) \leq \frac{2L\|x_1 - x^*\|^2}{t - 1}.$$

We will not prove this claim now, but note the dependence on t—this is a large improvement over the performance of the generic Projected Subgradient Descent algorithm.

The intuition for this algorithm is the Taylor expansion of f: We know $f(x+\delta) = f(x)+\nabla\cdot\delta+O(\delta^2)$. Algorithm 1 essentially ignores the quadratic term, but the smoothness condition allows us to control it better. In particular, the scale $1/L$ is precisely the scale at which we can guarantee that the higher-order terms do not swamp the first two terms. In other words, it allows us to choose the most aggressive step we can while ensuring that we continue to move downhill.

Our analysis will have two pieces: a lower bound (from the gradient) and a progress guarantee (from the smoothness). The trade-off between these two terms has a nice interpretation: when you don't make much progress, it's because the gradient is very flat, so you're already close to optimal. We will make this formal next time.

But there's a problem with all this: it's not true for max flow! As noted above, the sup norm is not L-smooth for any L, so it is not clear that we will be able to take advantage of this improved rate. The solution will be to replace the sup norm with the so-called *soft max*, a smooth function that approximates the maximum.