Lecture 11: Convex optimization

The topic of this week is convex optimization. The goal is to give a very short introduction. Since there are multiple very good online sources, I will only give very brief descriptions here.

Disclaimer: These lecture notes are informal in nature and are not thoroughly proofread. In case you find a serious error, please send email to the instructor pointing it out.

Convex optimization – overview

The general problem is to minimize a **convex function** over a **convex set**.

Let us define what the two phrases above mean. A convex set S is a subset of n-dimensional space, with the property that for any $u, v \in S$, the segment joining u, v completely lies inside S.

Examples of covex sets include: all of n-dimensional space, the unit sphere (i.e., set of all x s.t. $\|x\| \leq 1$), and the following, which we defined in class:

1. convex hull of a set of points
2. a cone defined by a set of points

Also, the intersection of two convex sets is convex. Now, a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is said to be convex over a domain D, if for all $x, y \in D$, we have

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y).$$

(We discussed several examples of convex functions, e.g., $\|x\|^2$.) For univariate functions ($n = 1$) that are differentiable everywhere, an equivalent characterization is that the derivative f' is monotone non-decreasing over the domain.

Now let us get back to the question we started with: how do we minimize a convex function f over a convex domain D?

A very natural (and popular) algorithm for the problem is what is known as *gradient descent*. This is an iterative 'local search' style algorithm, which goes through points $x^{(0)}, x^{(1)}, \ldots, x^{(t)}$, such that $f(x^{(i)}) < f(x^{(i-1)})$ for all i (i.e., the $f()$ value strictly decreases), and stops when such a *descent* is no longer possible.

How do we achieve this? Suppose we consider some $x^{(j)}$ and consider the gradient ∇f evaluated at the point $x^{(j)}$.[1] Now suppose we set

$$x^{(j+1)} = x^{(j)} - \eta \nabla f.$$

By the definition of the gradient, for any direction $\mathbf{u}$, we have

$$f(x + \eta \mathbf{u}) = f(x) + \eta \langle \nabla f, \mathbf{u} \rangle,$$

for *small* η (to make this formal, we need some conditions on how quickly the gradient can change).

[1] Recall: the gradient of f is the vector whose ith coordinate is the partitial derivative w.r.t. x_i ($\partial f / \partial x_i$).

In the above, we are effectively setting $\mathbf{u} = -\nabla f$. Thus we will have $f(x^{(j)} + \eta \nabla f) = f(x^{(j)}) - \eta \|\nabla f\|^2$. This is strictly less than $f(x^{(j)})$ as long as ∇f is non-zero. Thus, we can continue making updates as long as ∇f is non-zero. Suppose this happens. This means that we're either at a so-called *saddle point*, or we are at a local minimum.

The key thing about convex functions over a convex domain is the following theorem.

Theorem 1. *Let f be a convex function over a convex domain D, and suppose $\nabla f = 0$ at some point $\mathbf{u} \in D$. Then for all $\mathbf{v} \in D$, we have $f(\mathbf{u}) \leq f(\mathbf{v})$.*

Proof. Suppose, by way of contradiction, that there is a point $\mathbf{v}$ with $f(\mathbf{v}) < f(\mathbf{u})$. Then by convexity, if we move slightly along the line joining $\mathbf{u}, \mathbf{v}$, we have

$$f(\mathbf{u} + \eta(\mathbf{v} - \mathbf{u})) = f((1-\eta)\mathbf{u} + \eta\mathbf{v}) \leq (1-\eta) f(\mathbf{u}) + \eta f(\mathbf{v}) < f(\mathbf{u}).$$

The last inequality is because $f(\mathbf{v}) < f(\mathbf{u})$. But this is true no matter how small η is! Thus the gradient of $f()$ along $(\mathbf{v} - \mathbf{u})$ cannot be zero, which contradicts the assumption that $\nabla f = 0$. This completes the proof. □

The above discussion of gradient descent comes with many caveats. For instance, we could be at the boundary of a convex set, in which moving along the gradient will get to a point that is infeasible. There are ways of getting around it (e.g., *projecting back* at every step), but we do not discuss these issues. Also, how to set η is a typical problem in practice (it is called the *learning rate*). One way to go about this is to do a so-called *line-search*, and figure out the best possible value of η. The other is to choose a fixed, yet 'reasonable' value – the tradeoff here is that picking too small an η does not *make enough progress* towards the optimum, while picking a large one might no longer decrease $f()$.

Feasibility and optimization

A simple yet interesting observation is that convex optimization is *equivalent* to feasibility, i.e., minimizing f over a convex set D is equivalent to the problem of checking if a convex set is non-empty. Why is this?

The observation is that the *level set* of a convex function, i.e.,

$$L_\tau(f) := \{x \in D \ : \ f(x) \leq \tau\}$$

is a convex set for every value of τ. Now, the set $L_\tau(f) \cap D$ is convex, since it is the intersection of two convex sets. Further, checking if $L_\tau(f) \cap D$ is non-empty is equivalent to checking if there exists an $x \in D$ s.t. $f(x) \leq \tau$. Thus if we had an algorithm for this *feasibility problem*, we could do a binary search, and find $\min_{x \in D} f(x)$.

The feasibility problem also shows up when we're doing gradient descent – how do we pick the initial point?? Thus solving this feasibility problem for a given convex set is a key step in convex optimization.

Separation oracle

What do we mean by a "given convex set" above? I.e., how do we *specify* a convex set $D \subseteq \mathbb{R}^n$? One natural way is to define it using a set of linear constraints (e.g. as we did in linear programs). This is always

Separation oracle continued on next page...

possible (the constraints are sometimes called the *envelope* defining the convex set), but it might not always be possible using a finite set of constraints (think of writing the unit ball as a set of linear constraints).

From a computational point of view, things might get complicated even if the convex set were defined via an exponential number of linear constraints. Is there still a way to solve the feasibility problem?

It turns out that in many cases, all we require for a convex set D is a so-called *separation oracle*. Formally, this is an efficient algorithm that, given any query point $x \in \mathbb{R}^n$, does the following:

1. If $x \in D$, the algorithm says so.
2. If $x \notin D$, the algorithm outputs a hyperplane that separates x from D, i.e., it outputs an $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that $\langle \mathbf{a}, x \rangle < b$, and $\langle \mathbf{a}, y \rangle \geq b$ for all $y \in D$.

For a convex set D, it is always possible to produce such a hyperplane if $b \notin D$. One way to see this is to consider the closest point to b in the set D (call it c), and set $\mathbf{a} = c - b$.

There are two reasons for which the notion of a separation oracle is very interesting. The first is that efficient oracles could exist even for convex sets that *look complicated*, e.g., they could be defined by exponentially many, or even infinitely many hyperplanes. Second, it turns out that in many cases, an efficient separation oracle leads to an efficient algorithm for the feasibility problem – i.e., one of returning a feasible point x in the convex set.

Separation oracle for minimum spanning tree Given a graph $G = (V, E)$, a spanning tree is a collection of edges that (a) form a tree, and (b) connect all the vertices. Given positive weights on the edges, the minimum spanning tree problem asks to find a spanning tree of the smallest total weight.

We can write a linear programming formulation for this problem as follows. Suppose we have a variable for each edge x_e. Now the objective is to minimize $\sum_e w_e x_e$, where w_e is the weight of edge e. The constraints need to say that the graph formed by the x_e is connected. (Since the w_e are positive, minimizing the total weight will ensure they form a tree.)

Thus for every set S of vertices, we have the constraint that at least one edge in $E(S, \overline{S})$ is selected. I.e.,

$$\forall S \subseteq V, \qquad \sum_{e \in E(S,\overline{S})} x_e \geq 1.$$

We also have the standard constraints $0 \leq x_e \leq 1$. This is an exponential set of constraints (one for each S), and they define some convex set $D \subseteq \mathbb{R}^{|E|}$. Now for any $x \notin D$, we claim that we can efficiently find a violated constraint.

To see this, consider the graph on the same set of vertices V, with edges having weights x_e. Now, we wish to find a set S such the weighted value of the *cut* $(S, \overline{S})$ is as small as possible. If this is < 1, then we can find a violated constraint, and if not, it means that all the constraints are satisfied. But this is precisely the min cut problem, which we know can be solved in polynomial time!

So this is a natural instance in which we have a convex set defined by exponentially many constraints, but there is still an efficient separation oracle.

Separation oracle for the PSD cone In the next class, we will also see an instance of a convex set defined by infinitely many constraints, but there is still an efficient separation oracle. (Another simple example of this is the sphere in $\mathbb{R}^n$.)