Gradient Descent

In this lecture, we will study the gradient descent framework. This is a very general procedure to (approximately) minimize convex functions, and is applicable in many different settings. We use it to give an online algorithm with guarantees like those of the multiplicative weights algorithms.

1 The Basics

1.1 Norms and Inner Products

The inner product between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is written as $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$. Recall that the Euclidean norm of $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is given by

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}.$$

For any $c \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$, we get $\|c\mathbf{x}\| = |c| \cdot \|\mathbf{x}\|$, and also $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$. Moreover,

$$\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2\langle \mathbf{x}, \mathbf{y} \rangle$$

1.2 Convex Sets and Functions

Definition: Convex set

A set $K \subseteq \mathbb{R}^n$ is said to be convex if

$$\big(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}\big) \in K \qquad \forall \mathbf{x}, \mathbf{y} \in K, \forall \lambda \in [0,1]$$

Definition: Convex function

For a convex set $K \subseteq \mathbb{R}^n$, a function $f : K \to \mathbb{R}$ is said to be convex over K iff

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \forall \lambda \in [0,1]$$

Whenever K is not specified, assume $K = \mathbb{R}^n$.

In the context of this lecture, we will always assume that the function f is differentiable. The analog of the derivative in the multivariate case is the *gradient* ∇f, which is itself a function from $K \to \mathbb{R}^n$ defined as follows:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right).$$

We assume that the gradient is well-defined at all points in K.[1] Visually, if you draw the "level sets" of points where the function f takes on the same value as at $\mathbf{x}$, then the gradient $\nabla f(x)$ gives you the tangent plane to this level set at point $\mathbf{x}$.

Theorem 1: Convex function

A differentiable function $f : K \to \mathbb{R}$ is convex iff $\forall \mathbf{x}, \mathbf{y} \in K$,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle.$$

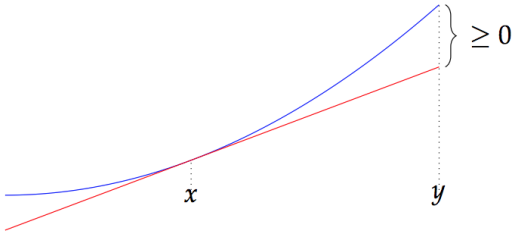

Figure 1: The blue line is function; the red line the tangent at x. Image from Nisheeth Vishnoi's notes.

Geometrically, Theorem 1 states that the function always lies above its tangent (see Fig 1). The right side is a "linear approximation" to the convex function at the point $\mathbf{x}$: this is like taking the constant and linear terms in a Taylor-series expansion of f.[2]

Example 1: F

r the univariate function $f(x) = x^2$, what is ∇f? Plot the curve $f(y)$, and for some fixed point x_0, the line $x_0^2 + \langle \nabla f(x_0), y - x_0 \rangle$. Check that the above inequality holds; i.e., the curve stays above the line for all y.

2 The Problems: Convex Minimization and Gradient Descent

There are two kinds of problems that we will concern ourself with:

1. *Unconstrained Convex Minimization (UCM)*: Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, find

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}).$$

2. *Constrained Convex Minimization (CCM)*: Given a convex set K and a convex function

[1] Many of the ideas here extend to non-differentiable cases too, using the concept of a sub-gradient.
[2] There are other ways of saying that a function is convex, but these will suffice for the moment.

$f : K \to \mathbb{R}$, find

$$\min_{\mathbf{x} \in K} f(\mathbf{x}).$$

Observe that the second problem contains within in the problem of solving a linear program (in that case the function f is linear, and the convex body is a polytope).

When we say "given a convex function f", how is f specified? In this lecture, we will assume we have access to both a "value oracle" that given a point $\mathbf{x} \in \mathbb{R}^n$ will tell us the value of $f(\mathbf{x})$, and also a "gradient oracle" that will tell us the value of $\nabla f(\mathbf{x})$.

2.1 Characterizations of Optimality

A crucial property of convex functions is that f is convex implies that all local minima are also global minima. This gives us a characterization of optimality for the unconstrained problem.

Claim 1: Global minima of convex functions

Given $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x}^* \in \mathbb{R}^n$ is a global minimizer of f exactly when $\nabla f(\mathbf{x}^*) = 0$.

Hence, solving $\nabla f(\mathbf{x}) = 0$ would enable us to compute the global minima exactly. Often, it may not be possible to get a closed form for ∇f. Today we show an algorithm that iteratively gets close to the optimal value.

For constrained minimization, the condition for optimality is slightly different:

Claim 2: Global minima of constrainted convex functions

Given $f : K \to \mathbb{R}$, $\mathbf{x}^* \in K$ is a global minimizer of f exactly when for all $\mathbf{y} \in K$,

$$\langle \nabla f(\mathbf{x}^*), \mathbf{y} - \mathbf{x}^* \rangle \geq 0.$$

Why does this make sense? First convince yourself that this makes sense in the 1-dimensional case, when f is a univariate function. Then use that a multivariate f is convex iff its restriction to each line is convex. The above definition is doing precisely that.

In this constrained case it is not clear how to "solve" for optimality, even if we have a closed form expression for f: this is more complicated than solving for the gradient being zero. Thankfully, we will see how to modify our solution for the unconstrained problem to also solve the constrained problem.

3 Unconstrained Convex Minimization

The idea behind gradient descent is simple: the gradient tells us the direction in which the function increases the most rapidly. So, to minimize f it is natural to move in the direction opposite to the gradient. But how far should we move in this direction?

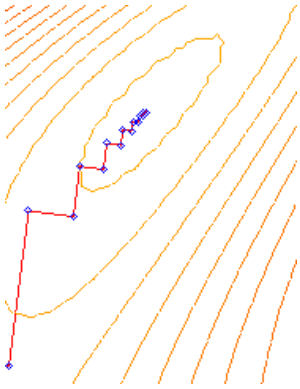

Figure 2: The yellow lines denote the level sets of the function f and the red walk denotes the steps of gradient descent. (Picture from wikipedia.)

The basic gradient descent "framework" says:

> Start with some point $\mathbf{x}_0$. At each step $t = 1, 2, \ldots, T-1$, set
>
> $$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta_t \cdot \nabla f(\mathbf{x}_t).$$
>
> return $\widehat{\mathbf{x}} = \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{x}_t$.

That's the entire "framework". In order to instantiate this framework and get a concrete algorithm, we need to specify how large the step size η_t is, and how many steps T we need.

Figure 2 gives a pictorial view of this algorithm. Note that the steps might not head straight for the minimizer, since each time you move in the direction of the negative gradient which may be pointing elsewhere. But our point slowly approaches the minimizer $\mathbf{x}^*$, at least in this picture. The surprising part is that the convergence holds for all convex functions, we can show tight bounds on how fast it converges. This is what we do next.

3.1 The Convergence Rate for Gradient Descent

The analysis we give works for all convex functions. Its guarantee will depend on two things:

- The distance of the starting point $\mathbf{x}_0$ from the optimal point $\mathbf{x}^*$. Define $D := \|\mathbf{x}_0 - \mathbf{x}^*\|$.
- A bound G on the norm of the gradient at any point $\mathbf{x} \in \mathbb{R}^n$. Specifically, we want that $\|\nabla f(\mathbf{x})\| \leq G$ for all $\mathbf{x} \in \mathbb{R}^n$.[3]

Our main theorem for this lecture is:

Theorem 2: Basic Gradient Descent

For any (differentiable) convex function $f : \mathbb{R}^n \to \mathbb{R}$ and any starting point $\mathbf{x}_0$, if we set $T = \left(\frac{GD}{\varepsilon}\right)^2$ and $\eta_t = \eta := \frac{D}{G\sqrt{T}}$, then

$$f(\widehat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \varepsilon.$$

[3]This is a very strong requirement, but if in the case of constrainted convex minimization, we will require that $\|\nabla f(\mathbf{x})\| \leq G$ only for $\mathbf{x} \in K$, which may be more reasonable.

Remember that G, D depend on both f and $\mathbf{x}_0$.

We'll prove this theorem by the end of this lecture, but some discussion before we do so.

1. If you think of G, D as constants, it takes $O(\frac{1}{\varepsilon^2})$ steps to get within ε of the optimal value. In other words, if we wanted to halve the error (make $\varepsilon \to \frac{\varepsilon}{2}$), the runtime would go up by a factor of 4. (Halving the error is like getting one more bit of precision.)
2. Indeed, if we assume nothing else about the function f, we cannot hope for any better guarantee than $f(\widehat{\mathbf{x}}) \le f(\mathbf{x}^*) + \varepsilon$ after $T = O(\frac{1}{\varepsilon})^2$ steps.
3. But if the functions are "nice" (e.g., strongly convex, or smooth), then you can improve the runtime.

   Indeed, if the functions are "well-conditioned" then we can even get the runtime to depend as $O(\log \frac{1}{\varepsilon})$, in which case getting each additional bit of precision increases the runtime *additively* only by $O(1)$. This is the kind of guarantee that the Ellipsoid algorithm achieves!

3.1.1 Online Convex Optimization

The proof for Theorem 2 actually works in a much stronger *online* setting. Let's state that result, and the give one proof for both theorems!

For the online setting, each day you need to choose a point $\mathbf{x}_t \in \mathbb{R}^n$, and the adversary chooses a convex function $f_t : \mathbb{R}^n \to \mathbb{R}$, and your cost is $f_t(\mathbf{x}_t)$. Just like in Lecture #19, we want an algorithm whose cost is not much more than that of the best fixed choice $\mathbf{x}^*$ in hindsight. I.e., you want that for any $\mathbf{x}^* \in \mathbb{R}^n$,

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^*)) \to 0 \qquad \text{as } T \to \infty$$

Here's how to solve this problem. We can use almost the same update rule as (3), with one slight modification. The update rule is now taken with respect to gradient of the current function f_t.

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta_t \cdot \nabla f_t(\mathbf{x}_t).$$

Theorem 3: Online Gradient Descent

For any (differentiable) convex function $f : \mathbb{R}^n \to \mathbb{R}$ and any starting point $\mathbf{x}_0$, if we set $\eta_t := \eta$, then for any point $\mathbf{x}^* \in \mathbb{R}^n$,

$$\sum_{t=0}^{T-1} f_t(\mathbf{x}_t) \le \sum_{t=0}^{T-1} f_t(\mathbf{x}^*) + \frac{\eta}{2}G^2 T + \frac{1}{2\eta}D^2.$$

where G is an upper bound on $\max_t \|\nabla f_t\|$, and $D := \|\mathbf{x}_0 - \mathbf{x}^*\|$.

Suppose we think of D, G as constants for now, and $\eta = \varepsilon$, then the "regret" (the difference between us and the best fixed solution $\mathbf{x}^*$) is

$$O(\varepsilon T) + O(\tfrac{1}{\varepsilon}).$$

For multiplicative weights, we had this being at most $O(\varepsilon T) + O(\frac{\log n}{\varepsilon})$ (see Slide 13 of the notes), and hence we are in the same ballpark. (A more careful analysis follows in a later section.)

3.1.2 The Proofs

We'll prove two things now. We first prove Theorem 3. Then we show that Theorem 2 follows from Theorem 3 using elementary properties of convexity.

Proof. (of Theorem 3) The proof is a short and sweet potential function argument. Define

$$\Phi_t := \frac{\|\mathbf{x}_t - \mathbf{x}^*\|^2}{2\eta}.$$

Note that $\Phi_0 = \frac{1}{2\eta} D^2$. We will show that

$$\text{amortized cost } f_t(\mathbf{x}_t) + (\Phi_{t+1} - \Phi_t) \leq f_t(\mathbf{x}^*) + \frac{\eta}{2} G^2.$$

Summing this up over all times gives

$$\sum_{t=0}^{T-1} f_t(\mathbf{x}_t) + (\Phi_T - \Phi_0) \leq \sum_{t=0}^{T-1} f_t(\mathbf{x}^*) + \frac{\eta}{2} G^2 T.$$

Now using that $\Phi_T \geq 0$ and $\Phi_0 = D^2/(2\eta)$ completes the proof.

To prove (3.1.2), let's calculate

$$\begin{aligned}
\Phi_{t+1} - \Phi_t = \frac{1}{2\eta}\left(\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 - \|\mathbf{x}_t - \mathbf{x}^*\|^2\right) &\overset{(F1)}{=} \frac{1}{2\eta}\left(\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 + 2\langle \mathbf{x}_{t+1} - \mathbf{x}_t, \mathbf{x}_t - \mathbf{x}^*\rangle\right) \\
&= \frac{1}{2\eta}\left(\eta^2 \|\nabla f_t(\mathbf{x}_t)\|^2 - 2\eta \langle \nabla f_t(\mathbf{x}_t), \mathbf{x}_t - \mathbf{x}^*\rangle\right) \\
&\leq \frac{\eta}{2} G^2 - \langle \nabla f_t(\mathbf{x}_t), \mathbf{x}_t - \mathbf{x}^*\rangle.
\end{aligned}$$

Next we use the convexity of f (via Theorem 1) to bound the difference

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*) \leq \langle \nabla f_t(\mathbf{x}_t), \mathbf{x}_t - \mathbf{x}^*\rangle$$

Summing up (3.1.2) and (3.1.2) means the inner-product term cancels, and gives us the amortized cost bound (3.1.2), and hence proves Theorem 3. □

Now to prove Theorem 2 from Theorem 3, note that if $f_t = f$ for all times t, we can use convexity to argue that

$$f(\widehat{\mathbf{x}}) = f\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{x}_t\right) \leq \frac{1}{T}\left(\sum_{t=0}^{T-1} f(\mathbf{x}_t)\right).$$

Bounding the sum using Theorem 3, we get

$$f(\widehat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \frac{1}{T}\left(\frac{\eta}{2}G^2T + \frac{1}{2\eta}D^2\right) = f(\mathbf{x}^*) + \frac{\eta}{2}G^2 + \frac{1}{2\eta T}D^2.$$

Now setting $T = (\frac{GD}{\varepsilon})^2$ and $\eta = \frac{D}{G\sqrt{T}} = \frac{\varepsilon}{G^2}$ means both the terms on the right equal $\varepsilon/2$. This proves Theorem 2.

4 Constrained Convex Minimization

Having done the analysis for the unconstrained case, we get the constrained case almost for free. The main difference is that the update step may take us outside K. So we just "project back into K". The algorithm is almost the same, let the blue parts highlight the changes.

Start with some point $\mathbf{x}_0$. At each step $t = 1, 2, \ldots, T-1$, set

$$\mathbf{y}_{t+1} \leftarrow \mathbf{x}_t - \eta_t \cdot \nabla f(\mathbf{x}_t).$$

Let $\mathbf{x}_{t+1}$ be the point in K closest to $\mathbf{y}_{t+1}$.

return $\widehat{\mathbf{x}} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{x}_t$.

Now if we satisfy that $\|\nabla f(x)\| \leq G$ for all $x \in K$, the online optimization theorem remains exactly the same.

Theorem 4: Constrained Online Gradient Descent

For any convex body $K \subseteq \mathbb{R}^n$, and sequence of (differentiable) convex functions $f_t : K \to \mathbb{R}$ and any starting point $\mathbf{x}_0$, if we set $\eta_t := \eta$, then for any point $\mathbf{x}^* \in \mathbb{R}^n$,

$$\sum_{t=0}^{T-1} f_t(\mathbf{x}_t) \leq \sum_{t=0}^{T-1} f_t(\mathbf{x}^*) + \frac{\eta}{2}G^2T + \frac{1}{2\eta}D^2.$$

where G is an upper bound on $\max_t \max_{\mathbf{x}\in K} \|\nabla f_t(\mathbf{x})\|$, and $D := \|\mathbf{x}_0 - \mathbf{x}^*\|$.

Proof. The proof changes amazingly little — let's see how. As before, we start to bound the potential change.

$$\Phi_{t+1} - \Phi_t = \frac{1}{2\eta}\left(\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 - \|\mathbf{x}_t - \mathbf{x}^*\|^2\right)$$

But now we claim that

$$\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{y}_{t+1} - \mathbf{x}^*\|^2.$$

Indeed, since $\mathbf{x}_{t+1}$ is the "projection" of $\mathbf{y}_{t+1}$ onto the convex set K. The proof is essentially by picture (see Figure 3):

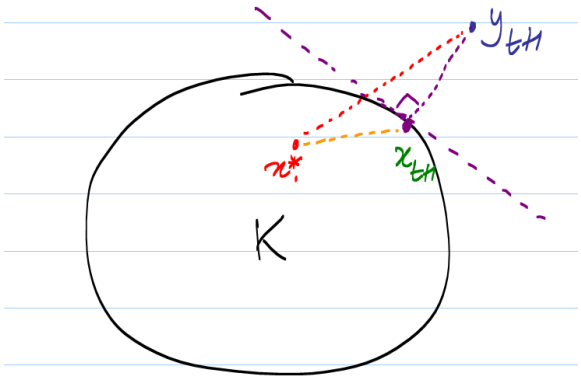

Figure 3: The projection ensures that $\mathbf{x}^*$ lies on the other side of the tangent hyperplane at $\mathbf{x}_{t+1}$, so the angle is obtuse. This means the squared length of the "hypotenuse" is larger than the squared length of either of the sides.

This means the changes die out almost immediately. Indeed,

$$\begin{aligned}\Phi_{t+1}-\Phi_t \le \frac{1}{2\eta}\left(\|\mathbf{y}_{t+1}-\mathbf{x}^*\|^2-\|\mathbf{x}_t-\mathbf{x}^*\|^2\right) &= \frac{1}{2\eta}\left(\|\mathbf{y}_{t+1}-\mathbf{x}_t\|^2+2\langle \mathbf{y}_{t+1}-\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle\right)\\ &= \frac{1}{2\eta}\left(\eta^2\|\nabla f_t(\mathbf{x}_t)\|^2-2\eta\langle\nabla f_t(\mathbf{x}_t),\mathbf{x}_t-\mathbf{x}^*\rangle\right)\\ &\le \frac{\eta}{2}G^2-\langle\nabla f_t(\mathbf{x}_t),\mathbf{x}_t-\mathbf{x}^*\rangle.\end{aligned}$$

And the rest of the argument is the same as in Theorem 2. □

5 An(other) Algorithm for Prediction Using Experts (Optional)

Thinking of D, G as constants is clearly cheating. We now show details of a different randomized algorithm for expert prediction. Suppose the convex body is

$$K = \{\mathbf{x}\in[0,1]^n \mid \sum_i x_i = 1\},$$

the "probability simplex". Suppose at each time the convex function is in fact linear:

$$f_t(\mathbf{x}) = \langle \mathbf{c}_t, \mathbf{x}\rangle$$

for some vector $\mathbf{c}_t \in [0,1]^n$. Let's see what the guarantees we get with these functions:

- For any $\mathbf{x}^*, \mathbf{x}_0 \in K$, the distance $D = \|\mathbf{x}^*-\mathbf{x}_0\| \le \|\mathbf{x}^*\|+\|\mathbf{x}_0\| \le 2$.
- Moreover, for any f_t, the gradient is $\nabla f_t(x) = \mathbf{c}_t$, whose length is at most $\sqrt{n}$. Hence $\|\nabla f_t(x)\| \le G := \sqrt{n}$.

Plugging this back into (3.1.1) with $\eta = \frac{\varepsilon}{n}$, $D = 2$ and $G = \sqrt{n}$, the guarantee we get is

$$\sum_{t=0}^{T-1}\big(f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\big) = \sum_{t=0}^{T-1}\big(\langle \mathbf{c}_t, \mathbf{x}_t\rangle - \langle \mathbf{c}_t, \mathbf{x}^*\rangle\big) \leq O(\varepsilon T + \tfrac{n}{\varepsilon}).$$

Now we use this deterministic procedure to get a different (randomized) algorithm for the experts problem from Lecture #19.

5.1 Algorithm for Expert Prediction

In the expert prediction problem, there are n experts. At each time we pick an expert (perhaps randomly). Then the costs $\mathbf{c}_t = (c_t(1), c_t(2), \ldots, c_t(n))$ incurred by these n experts are revealed, and our cost is the cost of the expert we chose.

Suppose we use the above gradient-descent based algorithm: at each time we get a vector $\mathbf{x}_t \in K$ from the algorithm, and we pick expert i with probability $x_t(i)$. Then

$$\text{our total expected cost} = \sum_{t=0}^{T-1}\sum_{i=1}^{n} c_t(i)x_t(i) = \sum_{t=0}^{T-1}\langle \mathbf{c}_t, \mathbf{x}_t\rangle.$$

If the best expert is $i^\star$, the cost incurred by this best expert is

$$\sum_{t=1}^{T-1} c_t(i^*) = \sum_{t=0}^{T-1}\langle \mathbf{c}_t, \mathbf{e}_{i*}\rangle,$$

where $\mathbf{e}_i$ is the unit vector with 1 in the i^{th} coordinate and zeros elsewhere.

Now combining this with (5), we get

$$\begin{aligned}\text{our total expected cost} - \text{cost of the best expert} &= \sum_{t=0}^{T-1}\langle \mathbf{c}_t, \mathbf{x}_t\rangle - \sum_{t=0}^{T-1}\langle \mathbf{c}_t, \mathbf{e}_{i*}\rangle \\ &\leq O(\varepsilon T + \frac{n}{\varepsilon}).\end{aligned}$$

Now you can relate this to multiplicative weights more precisely. Using Randomized Weighted Majority from the previous lecture, this was at most $O(\varepsilon T) + O(\frac{\log n}{\varepsilon})$, so our dependence on the number of experts is a bit worse. [4]

[4] You can use a more nuanced version of Gradient descent called Mirror Descent to derive the same bounds as Randomized Weighted Majority.