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 also show how this gives an online algorithm with guarantees somewhat similar to the multiplicative weights algorithms.[1]

1 Convex Sets and Functions

First, recall the following standard definitions:

Definition 1 *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 2 *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 $K \to \mathbb{R}$ 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.[2] 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 $\mathbf{x}$.

Fact 3 *A differentiable function $f : K \to \mathbb{R}$ is convex iff $\forall \mathbf{x}, \mathbf{y} \in K$,*

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

Geometrically, Fact 3 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.[3]

> **Example:** For 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.

[1] More details on these topics can be found, e.g., in notes by Sebastian Bubeck, Elad Hazan, or Nisheeth Vishnoi.
[2] Many of the ideas here extend to non-differentiable cases too, using the concept of a sub-gradient.
[3] There are other ways of saying that a function is convex, but these will suffice for the moment.

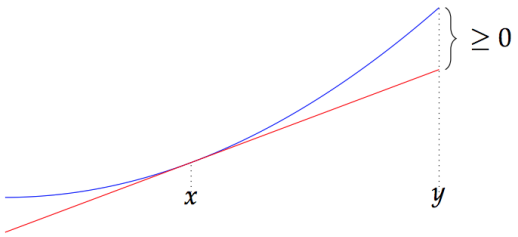

Figure 1: The blue line denotes the function and the red line is the tangent at x. Image from Nisheeth Vishnoi's notes.

1.1 Norms and Inner Products

Finally, recall that the Euclidean norm of $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is given by

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

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,

$$\langle a, b \rangle = \frac{1}{2} \left[\|a\|^2 + \|b\|^2 - \|a - b\|^2 \right] \tag{F1}$$

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

Fact 4 *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:

Fact 5 *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.

Now 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?

The basic gradient descent algorithm 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 algorithm. We need to specify how large the step size η_t is, and how many steps T we need.

3.1 The Analysis

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 $\|\nabla f(\mathbf{x})\| \leq D$ for all $\mathbf{x} \in \mathbb{R}^n$.[4]

The theorem is the following:

[4] 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.

Theorem 6 *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*

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

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

Proof: We first use the convexity of f (via Fact 3) to bound the difference

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

And now using the gradient descent update rule:

$$\begin{aligned}
&= \left\langle \frac{1}{\eta}(\mathbf{x}_t - \mathbf{x}_{t+1}), \mathbf{x}_t - \mathbf{x}^* \right\rangle \\
&= \frac{1}{2\eta}\Big(\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}_t^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \Big) \qquad \text{(Using Fact F1)}
\end{aligned}$$

Now denote the distance $\|\mathbf{x}_t - \mathbf{x}_t^*\|$ by Δ_t, and use the update rule again to get:

$$\begin{aligned}
&= \frac{1}{2\eta}\Big(\|\eta \cdot \nabla f(\mathbf{x}_t)\|^2 + \Delta_t^2 - \Delta_{t+1}^2 \Big) \\
&= \frac{1}{2\eta}\Big(\eta^2 \|\nabla f(\mathbf{x}_t)\|^2 + \Delta_t^2 - \Delta_{t+1}^2 \Big) \leq \frac{1}{2\eta}\Big(\eta^2 G^2 + \Delta_t^2 - \Delta_{t+1}^2 \Big)
\end{aligned}$$

The $\Delta_t^2 - \Delta_{t+1}^2$ at the end cries out to be summed over all t. Indeed, doing that we get:

$$\begin{aligned}
\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}^*)) &\leq \sum_{t=0}^{T-1} \frac{1}{2\eta}\Big(\eta^2 G^2 + \Delta_t^2 - \Delta_{t+1}^2 \Big) \\
&= \frac{\eta}{2} G^2 T + \frac{1}{2\eta}\Big(\Delta_0^2 - \Delta_T^2 \Big).
\end{aligned}$$

Of course, $\Delta_0^2 = D^2$ by definition, and $\Delta_T^2 \geq 0$, so this gives us

$$\leq \frac{\eta}{2} G^2 T + \frac{1}{2\eta} D^2.$$

If we minimize η, we'll get $\eta = \frac{D}{G\sqrt{T}}$. And then setting $T = (\frac{DG}{\varepsilon})^2$ gives

$$\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}^*)) \leq \varepsilon\, T.$$

We're almost there:

$$\begin{aligned}
f(\widehat{\mathbf{x}}) - f(\mathbf{x}^*) &= f\left(\frac{\sum_{t=0}^{T-1} \mathbf{x}_t}{T} \right) - f(\mathbf{x}^*) \\
&\leq \frac{1}{T} \sum_{t=0}^{T-1} \Big(f(\mathbf{x}_t) - f(\mathbf{x}^*) \Big) \qquad \text{By convexity of } f \\
&\leq \frac{1}{T} \varepsilon\, T = \varepsilon.
\end{aligned}$$

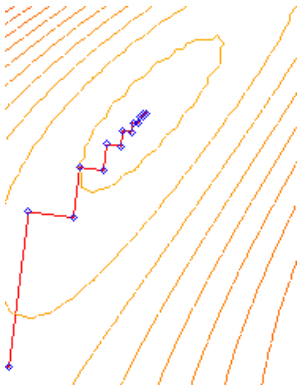

Figure 2: The yellow lines denote the level sets of the function f and the red walk denotes the steps of gradient descent. (Picture from wikipedia.)

Hence, the function value of the output $\widehat{\mathbf{x}}$ is ϵ-close to the optimal value. Just what we wanted. ■

Here's a picture for this algorithm. Note that the steps might not head straight for the minimizer, but it slowly approaches it.

One comment about the convergence rate: ignoring the dependence on G, D, note that 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.) Ideally, we would like 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 $O(1)$.

If we assume nothing else about the function f, we cannot hope for any better guarantee than $f(\widehat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \varepsilon$ after $T = O(\frac{1}{\varepsilon})^2$ steps. But if the functions are "nice" (e.g., strongly convex, or Lipschitz), then you can improve the runtime.

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 theorem remains exactly the same.

Theorem 7 *For any (differentiable) convex function $f : \mathbb{R}^n \to \mathbb{R}$, if we set $T = \left(\frac{GD}{\varepsilon}\right)^2$ and $\eta_t = \eta := \frac{D}{G\sqrt{T}}$, then*

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

Proof: Only the beginning of the proof changes slightly — let's see how.

$$\begin{aligned} f(\mathbf{x}_t) - f(\mathbf{x}^*) &\le \langle \nabla f(\mathbf{x}_t), \mathbf{x}_t - \mathbf{x}^* \rangle && \text{(By convexity of } f\text{)} \\ &= \left\langle \frac{1}{\eta}(\mathbf{x}_t - \mathbf{y}_{t+1}), \mathbf{x}_t - \mathbf{x}^* \right\rangle && \text{(By update rule)} \\ &= \frac{1}{2\eta}\Big(\|\mathbf{x}_t - \mathbf{y}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}_t^*\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}^*\|^2 \Big) && \text{(Using Fact F1)} \end{aligned}$$

But we claim that

$$\|\mathbf{y}_{t+1} - \mathbf{x}^*\| \ge \|\mathbf{x}_{t+1} - \mathbf{x}^*\|,$$

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): Hence we get

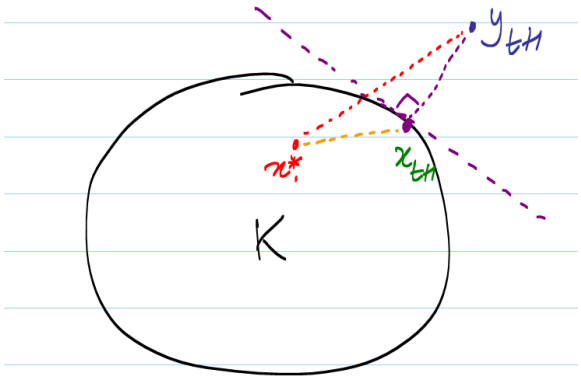

Figure 3: The projection ensures that $\mathbf{x}^*$ lies on the other side of the tangent hyperplane at $\mathbf{x}_{t+1}$, so the angle is obtuse. This means the squared length of the "hypotenuse" is larger than the squared length of either of the sides.

$$\begin{aligned} f(\mathbf{x}_t) - f(\mathbf{x}^*) &\le \frac{1}{2\eta}\Big(\|\mathbf{x}_t - \mathbf{y}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}_t^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \Big) \\ &= \frac{1}{2\eta}\Big(\|\eta \cdot \nabla f(\mathbf{x}_t)\|^2 + \Delta_t^2 - \Delta_{t+1}^2 \Big) \end{aligned}$$

And the rest of the argument is the same as in Theorem 6. ■

5 A Multiplicative Weights-like Algorithm

Consider the following problem: we fix a convex set K. Each day you need to choose a point $\mathbf{x}_t \in K$, and the adversary chooses a convex function $f_t : K \to \mathbb{R}$, and your cost is $f_t(\mathbf{x}_t)$. Just like in Lecture #22, 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 $\mathbf{x}^* \in K$,

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

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

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

The crucial point is that the above analysis, almost until the end, never used that the functions f were the same at all times. So if we replace each occurrence of $f()$ by $f_t()$, we get that

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

Now G is an upper bound on the gradients of all $f_t(\cdot)$.

Suppose we think of D, G as constants for now, and $\eta = \varepsilon$, then this 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 the next section.)

5.1 An(other) Algorithm for Prediction Using Experts

But thinking of D, G as constants is clearly cheating. We now show how to get 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 function is

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

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

Plugging this back into (1) with $\eta = \frac{\varepsilon}{n}$, $D = 2$ and $G = \sqrt{n}$, the guarantee we get is

$$\sum_{t=0}^{T-1} \big(f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)\big) = \sum_{t=0}^{T-1} \big(\langle \mathbf{c}_t, \mathbf{x}_t \rangle - \langle \mathbf{c}_t, \mathbf{x}^* \rangle \big) \leq O(\varepsilon T + \tfrac{N}{\varepsilon}). \tag{2}$$

5.1.1 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 (2), 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 RWM, this was at most $O(\varepsilon T) + O(\frac{\log n}{\varepsilon})$, so our dependence on the number of experts is worse.