Course Notes for EE227C (Spring 2018): Convex Optimization and Approximation

Instructor: Moritz Hardt
Email: hardt+ee227c@berkeley.edu

Graduate Instructor: Max Simchowitz
Email: msimchow+ee227c@berkeley.edu

March 4, 2018

Abstract

This course explores some theory and algorithms for nonlinear optimization. We will focus on problems that arise in machine learning and modern data analysis, paying attention to concerns about complexity, robustness, and implementation in these domains. We will also see how tools from convex optimization can help tackle non-convex optimization problems common in practice.

Code examples are available at:

https://ee227c.github.io/.

Below are the course notes for EE227C (Spring 2018): Convex Optimization and Approximation, taught at UC Berkeley.

Contents

1 Lecture 1: Convexity

This lecture provides the most important facts about convex sets and convex functions that we'll heavily make use of. These are often simple consequences of Taylor's theorem.

1.1 Convex sets

Definition 1.1 (Convex set)**.** A set $K \subseteq \mathbb{R}^n$ is *convex* if it the line segment between any two points in K is also contained in K. Formally, for all $x, y \in K$ and all scalars $\gamma \in [0,1]$ we have $\gamma x + (1-\gamma)y \in K$.

Theorem 1.2 (Separation Theorem)**.** *Let $C, K \subseteq \mathbb{R}^n$ be convex sets with empty intersection $C \cap K = \emptyset$. Then there exists a point $a \in \mathbb{R}^n$ and a number $b \in \mathbb{R}$ such that*

1. *for all $x \in C$, we have $\langle a, x\rangle \geqslant b$.*
2. *for all $x \in K$, we have $\langle a, x\rangle \leqslant b$.*

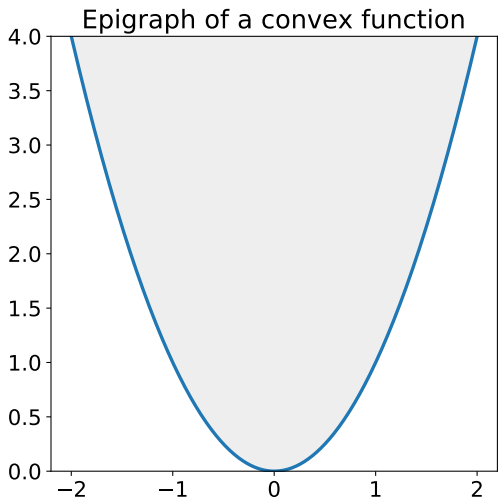
If C and K are closed and at least one of them is bounded, then we can replace the inequalities by strict inequalities.

The case we're most concerned with is when both sets are compact (i.e., closed and bounded). We highlight its proof here.

Proof of Theorem 1.2 for compact sets. In this case, the Cartesian product $C \times K$ is also compact. Therefore, the distance function $\|x - y\|$ attains its minimum over $C \times K$. Taking p, q to be two points that achieve the minimum. A separating hyperplane is given by the hyperplane perpendicular to $q - p$ that passes through the midpoint between p and q. That is, $a = q - p$ and $b = (\langle a, q\rangle - \langle a, p\rangle)/2$. For the sake of contradiction, suppose there is a point r on this hyperplane contained in one of the two sets, say, C. Then the line segment from p to r is also contained in C by convexity. We can then find a point along the line segment that is close to q than p is, thus contradicting our assumption. ■

1.1.1 Notable convex sets

- Linear spaces $\{x \in \mathbb{R}^n \mid Ax = 0\}$ and halfspaces $\{x \in \mathbb{R}^n \mid \langle a, x\rangle \geqslant 0\}$
- Affine transformations of convex sets. If $K \subseteq \mathbb{R}^n$ is convex, so is $\{Ax + b \mid x \in K\}$ for any $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. In particular, affine subspaces and affine halfspaces are convex.
- Intersections of convex sets. In fact, every convex set is equivalent to the intersection of all affine halfspaces that contain it (a consequence of the separating hyperplane theorem).



- The cone of positive semidefinite matrices, denotes, $S_+^n = \{A \in \mathbb{R}^{n\times n} \mid A \succeq 0\}$. Here we write $A \succeq 0$ to indicate that $x^\top Ax \geqslant 0$ for all $x \in \mathbb{R}^n$. The fact that S_+^n is convex can be verified directly from the definition, but it also follows from what we already knew. Indeed, denoting by $S_n = \{A \in \mathbb{R}^{n\times n} \mid A^\top = A\}$ the set of all $n \times n$ symmetric matrices, we can write S_+^n as an (infinite) intersection of halfspaces $S_+^n = \bigcap_{x\in\mathbb{R}^n\setminus\{0\}} \{A \in S_n \mid x^\top Ax \geqslant 0\}$.
- See Boyd-Vandenberghe for lots of other examples.

1.2 Convex functions

Definition 1.3 (Convex function)**.** A function $f\colon \Omega \to \mathbb{R}$ is *convex* if for all $x, y \in \Omega$ and all scalars $\gamma \in [0,1]$ we have $f(\gamma x + (1-\gamma)y) \leqslant \gamma f(x) + (1-\gamma)f(y)$.

Jensen (1905) showed that for continuous functions, convexity follows from the "midpoint" condition that for all $x, y \in \Omega$,

$$f\left(\frac{x+y}{2}\right) \leqslant \frac{f(x)+f(y)}{2}.$$

This result sometimes simplifies the proof that a function is convex in cases where we already know that it's continuous.

Definition 1.4. The *epigraph* of a function $f\colon \Omega \to \mathbb{R}$ is defined as

$$\operatorname{epi}(f) = \{(x,t) \mid f(x) \leqslant t\}.$$

Fact 1.5. *A function is convex if and only if its epigraph is convex.*

Convex functions enjoy the property that local minima are also global minima. Indeed, suppose that $x \in \Omega$ is a local minimum of $f\colon \Omega \to \mathbb{R}$ meaning that any point in a neighborhood

around x has larger function value. Now, for every $y \in \Omega$, we can find a small enough γ such that

$$f(x) \leqslant f((1-\gamma)x + \gamma y) \leqslant (1-\gamma)f(x) + \gamma f(y)\,.$$

Therefore, $f(x) \leqslant f(y)$ and so x must be a global minimum.

1.2.1 First-order characterization

It is helpful to relate convexity to Taylor's theorem, which we recall now. We define the *gradient* of a differentiable function $f\colon \Omega \to \mathbb{R}$ at $x \in \Omega$ as the vector of partial derivatives

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_i}\right)_{i=1}^{n}.$$

We note the following simple fact that relates linear forms of the gradient to a one-dimensional derivative evaluated at 0. It's a consequence of the multivariate chain rule.

Fact 1.6. *Assume $f\colon \Omega \to \mathbb{R}$ is differentiable and let $x, y \in \Omega$. Then,*

$$\nabla f(x)^\top y = \left.\frac{\partial f(x+\gamma y)}{\partial \gamma}\right|_{\gamma=0}.$$

Taylor's theorem implies the following statement.

Proposition 1.7. *Assume $f\colon \Omega \to \mathbb{R}$ is continuously differentiable along the line segment between two points x and y. Then,*

$$f(y) = f(x) + \nabla f(x)^\top (y-x) + \int_0^1 (1-\gamma)\frac{\partial^2 f(x+\gamma(y-x))}{\partial \gamma^2}\,\mathrm{d}\gamma$$

Proof. Apply a second order Taylor's expansion to $g(\gamma) = f(x+\gamma(y-x))$ and apply Fact 1.6 to the first-order term. ■

Among differentiable functions, convexity is equivalent to the property that the first-order Taylor approximation provides a global lower bound on the function.

Proposition 1.8. *Assume $f\colon \Omega \to \mathbb{R}$ is differentiable. Then, f is convex if and only if for all $x, y \in \Omega$ we have*

$$f(y) \geqslant f(x) + \nabla f(x)^\top (y-x)\,. \tag{1}$$

Proof. First, suppose f is convex, then by definition

$$\begin{aligned} f(y) &\geqslant \frac{f((1-\gamma)x + \gamma y) - (1-\gamma)f(x)}{\gamma} \\ &\geqslant f(x) + \frac{f(x+\gamma(y-x)) - f(x)}{\gamma} \\ &\to f(x) + \nabla f(x)^\top (y-x) \quad \text{as } \gamma \to 0 \qquad \text{(by Fact 1.6.)} \end{aligned}$$

On the other hand, fix two points $x, y \in \Omega$ and $\gamma \in [0,1]$. Putting $z = \gamma x + (1-\gamma)y$ we get from applying Equation 1 twice,

$$f(x) \geqslant f(z) + \nabla f(z)^\top (x-z) \quad \text{and} \quad f(y) \geqslant f(z) + \nabla f(z)^\top (y-z)$$

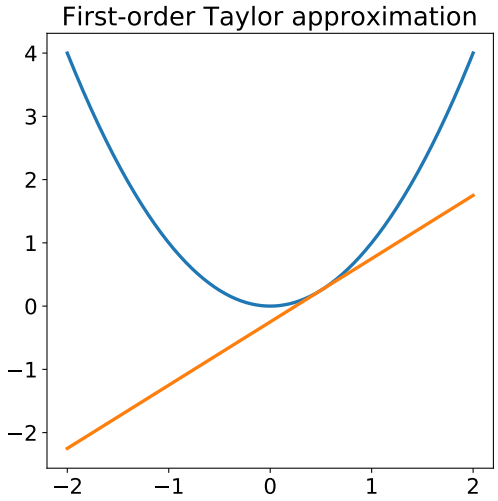


Figure 1: Taylor approximation of $f(x) = x^2$ at 0.5.

Adding these inequalities scaled by γ and $(1-\gamma)$, respectively, we get $\gamma f(x) + (1-\gamma) f(y) \geqslant f(z)$, which establishes convexity. ■

A direct consequence of Proposition 1.8 is that if $\nabla f(x) = 0$ vanishes at a point x, then x must be a global minimizer of f.

Remark 1.9 (Subgradients)**.** *Of course, not all convex functions are differentiable. The absolute value $f(x) = |x|$, for example, is convex but not differentiable at 0. Nonetheless, for every x, we can find a vector g such that*

$$f(y) \geqslant f(x) + g^\top (y - x)\,.$$

Such a vector is called a subgradient *of f at x. The existence of subgradients is often sufficient for optimization.*

1.2.2 Second-order characterization

We define the *Hessian* matrix of $f : \Omega \to \mathbb{R}$ at a point $x \in \Omega$ as the matrix of second order partial derivatives:

$$\nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)_{i,j \in [n]}\,.$$

Schwarz's theorem implies that the Hessian at a point x is symmetric provided that f has continuous second partial derivatives in an open set around x.

In analogy with Fact 1.6, we can relate quadratic forms in the Hessian matrix to one-dimensional derivatives using the chain rule.

Fact 1.10. *Assume that $f : \Omega \to \mathbb{R}$ is twice differentiable along the line segment from x to y. Then,*

$$y^\top \nabla^2 f(x + \gamma y) y = \frac{\partial^2 f(x + \gamma y)}{\partial \gamma^2}\,.$$

Proposition 1.11. *If f is twice continuously differentiable on its domain Ω, then f is convex if and only if $\nabla^2 f(x) \succeq 0$ for all $x \in \Omega$.*

Proof. Suppose f is convex and our goal is to show that the Hessian is positive semidefinite. Let $y = x + \alpha u$ for some arbitrary vector u and scalar α. Proposition 1.8 shows

$$f(y) - f(x) - \nabla f(x)^\top (y - x) \geqslant 0$$

Hence, by Proposition 1.7,

$$\begin{aligned} 0 &\leqslant \int_0^1 (1-\gamma) \frac{\partial^2 f(x + \gamma(y-x))}{\partial \gamma^2} \, \mathrm{d}\gamma \\ &= (1-\gamma) \frac{\partial^2 f(x + \gamma(y-x))}{\partial \gamma^2} \quad \text{for some } \gamma \in (0,1) \qquad \text{(by the mean value theorem)} \\ &= (1-\gamma)(y-x)^\top \nabla^2 f(x + \gamma(y-x))(y-x) \,. \qquad \text{(by Fact 1.10)} \end{aligned}$$

Plugging in our choice of y, this shows $0 \leqslant u^\top \nabla^2 f(x + \alpha\gamma u) u$. Letting α tend to zero establishes that $\nabla^2 f(x) \succeq 0$. (Note that γ generally depends on α but is always bounded by 1.)

Now, suppose the Hessian is positive semidefinite everywhere in Ω and our goal is to show that the function f is convex. Using the same derivation as above, we can see that the second-order error term in Taylor's theorem must be nonnegative. Hence, the first-oder approximation is a global lower bound and so the function f is convex by Proposition 1.8. ■

1.3 Convex optimization

Much of this course will be about different ways of minimizing a convex function $f \colon \Omega \to \mathbb{R}$ over a convex domain Ω :

$$\min_{x \in \Omega} f(x)$$

Convex optimization is not necessarily easy! For starters, convex sets do not necessarily enjoy compact descriptions. When solving computational problems involving convex sets, we need to worry about how to represent the convex set we're dealing with. Rather than asking for an explicit description of the set, we can instead require a computational abstraction that highlights essential operations that we can carry out. The Separation Theorem motivates an important computational abstraction called *separation oracle*.

Definition 1.12. A *separation oracle* for a convex set K is a device, which given any point $x \notin K$ returns a hyperplane separating x from K.

Another computational abstraction is a *first-order oracle* that given a point $x \in \Omega$ returns the gradient $\nabla f(x)$. Similarly, a *second-order oracle* returns $\nabla^2 f(x)$. A function value oracle or *zeroth-order oracle* only returns $f(x)$. First-order methods are algorithms that make do with a first-order oracle.

1.3.1 What is efficient?

Classical complexity theory typically quantifies the resource consumption (primarily running time or memory) of an algorithm in terms of the bit complexity of the input. This approach

can be cumbersome in convex optimization and most textbooks shy away from it. Instead, it's customary in optimization to quantify the cost of the algorithm in terms of how often it accesses one of the oracles we mentioned.

The definition of "efficient" is not completely cut and dry in optimization. Typically, our goal is to show that an algorithm finds a solution x with $f(x) = \min_{x\in\Omega} f(x) + \epsilon$ for some additive error $\epsilon > 0$. The cost of the algorithm will depend on the target error. Highly practical algorithms often have a polynomial dependence on ϵ, such as $O(1/\epsilon)$ or even $O(1/\epsilon^2)$. Other algorithms achieve $O(\log(1/\epsilon))$ steps in theory, but are prohibitive in their actual computational cost. Technically, if we think of the parameter ϵ as being part of the input, it takes only $O(\log(1/\epsilon))$ bits to describe the error parameter. Therefore, an algorithm that depends more than logarithmically on $1/\epsilon$ may not be polynomial time algorithm in its input size.

In this course, we will make an attempt to highlight both the theoretical performance and practical appeal of an algorithm. Moreover, we will discuss other performance criteria such as robustness to noise. How well an algorithm performs is rarely decided by a single criterion, and usually depends on the application at hand.

2 Lecture 2: Gradient method

In this lecture we encounter the fundamentally important *gradient method* and a few ways to analyze its convergence behavior. The goal here is to solve a problem of the form

$$\min_{x\in\Omega} f(x)$$

where we'll make some additional assumptions on the function $f\colon \Omega \to \mathbb{R}$. The technical exposition closely follows the corresponding chapter in Bubeck's text [Bub15].

2.1 Gradient descent

For a differentiable function f, the basic gradient method starting from an initial point x_0 is defined by the iterative description

$$x_{t+1} = x_t - \eta \nabla_t f(x_t) \qquad\qquad (t \geqslant 0)$$

where η_t is the so-called *step size* that may vary with t.

The first assumption that leads to a convergence analysis is that the gradients of the function aren't too big over the domain.

Definition 2.1 (L-Lipschitz)**.** A differentiable function f is said to be *L-Lipschitz* over the domain Ω if for all $x \in \Omega$, we have

$$\|\nabla f(x)\| \leqslant L\,.$$

Fact 2.2. *If a function f is L-Lipschitz, it implies that the difference between two points in the range is bounded,*

$$|f(x) - f(y)| \leqslant L\|x - y\|$$

How can we ensure that $x_{t+1} \in \Omega$? One natural approach is to "project" each iterate back onto the domain Ω.

Definition 2.3 (Projection)**.** The *projection* of a point x onto a set Ω is defined as

$$\Pi_\Omega(x) = \arg\min_{y \in \Omega} \|x - y\|$$

Example 2.4. A projection onto the Euclidean ball B_2 is just normalization:

$$\Pi_{B_2}(x) = \frac{x}{\|x\|}$$

A crucial property of projections is that when $x \in \Omega$, we have for any y (possibly outside Ω):

$$\|\Pi_\Omega(y) - x\|^2 \leqslant \|y - x\|^2$$

That is, the projection of y onto a convex set containing x is closer to x.

Lemma 2.5.

$$\|\Pi_\Omega(y) - x\|^2 \leqslant \|y - x\|^2 - \|y - \Pi_\Omega(y)\|^2$$

Which follows from the Pythagorean theorem. Note that this lemma implies the above property.

2.1.1 Modifying gradient descent with projections

So now we can modify our original procedure to use two steps.

$$y_{t+1} = x_t - \eta \nabla f(x_t)$$

$$x_{t+1} = \Pi_\Omega(y_{t+1})$$

And we are guaranteed that $x_{t+1} \in \Omega$. Note that computing the projection may be the hardest part of your problem, as you are computing an arg min. However, there are convex sets for which we know explicitly how to compute the projection (see Example 2.4).

2.2 Convergence rate of gradient descent for Lipschitz functions

Theorem 2.6 (Projected Gradient Descent for L-Lipschitz Functions)**.** *Assume that function f is convex, differentiable, and closed with bounded gradients. Let L be the Lipschitz constant of f over the convex domain Ω. Let R be the upper bound on the distance $\|x_1 - x^*\|_2$ from the initial point x_1 to the optimal point $x^* = \arg\min_{x \in \Omega} f(x)$. Let t be the number of iterations of project gradient descent. If the learning rate η is set to $\eta = \frac{R}{L\sqrt{(t)}}$, then*

$$f\left(\frac{1}{t}\sum_{s=1}^{t} x_s\right) - f\left(x^*\right) \leqslant \frac{RL}{\sqrt{t}}.$$

This means that the difference between the functional value of the average point during the optimization process from the optimal value is bounded above by a constant proportional to $\frac{1}{\sqrt{t}}$.

Before proving the theorem, recall the "Fundamental Theorem of Optimization", which is that an inner product can be written as a sum of norms: $u^\top v = \frac{1}{2}(\|u\|^2 + \|v\|^2 - \|u - v\|^2)$. This property can be seen from $\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2u^\top v$.

Proof of Theorem 2.6. The proof begins by first bounding the difference in function values $f(x_s) - f(x^*)$.

$$
\begin{aligned}
f(x_s) - f(x^*) &\leqslant \nabla f(x_s)^\top (x_s - x^*) && (2)\\
&= \frac{1}{\eta}(x_s - y_{s+1})^\top (x_s - x^*) && (3)\\
&= \frac{1}{2\eta}\left(\|x_s - x^*\|^2 + \|x_s - y_{s+1}\|^2 - \|y_{s+1} - x^*\|^2\right) && (4)\\
&= \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2\right) + \frac{\eta}{2}\|\nabla f(x_s)\|^2 && (5)\\
&\leqslant \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2\right) + \frac{\eta L^2}{2} && (6)\\
&\leqslant \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2\right) + \frac{\eta L^2}{2} && (7)
\end{aligned}
$$

Equation 2 comes from the definition of convexity. Equation 3 comes from the update rule for projected gradient descent. Equation 4 comes from the "Fundamental Theorem of Optimization." Equation 5 comes from the update rule for projected gradient descent. Equation 6 is because f is L-Lipschitz. Equation 7 comes from Lemma 2.5.

Now, sum these differences from $s = 1$ to $s = t$:

$$
\begin{aligned}
\sum_{s=1}^{t} f(x_s) - f(x^*) &\leqslant \frac{1}{2\eta}\sum_{s=1}^{t}\left(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2\right) + \frac{\eta L^2 t}{2} && (8)\\
&= \frac{1}{2\eta}\left(\|x_1 - x^*\|^2 - \|x_t - x^*\|^2\right) + \frac{\eta L^2 t}{2} && (9)\\
&\leqslant \frac{1}{2\eta}\|x_1 - x^*\|^2 + \frac{\eta L^2 t}{2} && (10)\\
&\leqslant \frac{R^2}{2\eta} + \frac{\eta L^2 t}{2} && (11)
\end{aligned}
$$

Equation 9 is because Equation 8 is a telescoping sum. Equation 10 is because $\|x_t - x^*\|^2 \geqslant 0$. Equation 11 is by the assumption that $\|x_1 - x^*\|^2 \leqslant R^2$.

Finally,

$$
\begin{aligned}
f\left(\frac{1}{t}\sum_{s=1}^{t} x_s\right) - f(x^*) &\leqslant \frac{1}{t}\sum_{s=1}^{t} f(x_s) - f(x^*) && \text{(by convexity)}\\
&\leqslant \frac{R^2}{2\eta t} + \frac{\eta L^2}{2} && \text{(by Equation 11)}\\
&\leqslant \frac{RL}{\sqrt{t}} && \text{(for } \eta = R/L\sqrt{t}.\text{)}
\end{aligned}
$$

■

2.3 Convergence rate for smooth functions

The next property we'll encounter is called *smoothness* and it often leads to stronger convergence guarantees.

Definition 2.7 (β-smoothness)**.** A continuously differentiable function f is β smooth if the gradient ∇f is β-Lipschitz, i.e

$$\|\nabla f(x) - \nabla f(y)\| \leqslant \beta \|x - y\|$$

Lemma 2.8. *Let f be a β-smooth function on $\mathbb{R}^n$. Then for any $x, y \in \mathbb{R}^n$, one has*

$$|f(x) - f(y) - \nabla f(y)^\top (x - y)| \leqslant \frac{\beta}{2}\|x - y\|^2$$

Proof. First represent $f(x) - f(y)$ as an integral, apply Cauchy-Schwarz and then β-smoothness:

$$\begin{aligned}
&|f(x) - f(y) - \nabla f(y)^\top (x - y)| \\
&\qquad = \left| \int_0^1 \nabla f(y + t(x - y))^\top (x - y) dt - \nabla f(y)^\top (x - y) \right| \\
&\qquad \leqslant \int_0^1 \|\nabla f(y + t(x - y)) - \nabla f(y)\| \cdot \|x - y\| dt \\
&\qquad \leqslant \int_0^1 \beta t \|x - y\|^2 dt \\
&\qquad = \frac{\beta}{2}\|x - y\|^2
\end{aligned}$$

■

We also need the following lemma.

Lemma 2.9. *Let f be a β-smooth function, then for any $x, y \in \mathbb{R}^n$, one has*

$$f(x) - f(y) \leqslant \nabla f(x)^\top (x - y) - \frac{1}{2\beta}\|\nabla f(x) - \nabla f(y)\|$$

Proof. Let $z = y - \frac{1}{\beta}(\nabla f(y) - \nabla f(x))$. Then one has

$$\begin{aligned}
&f(x) - f(y) \\
&= f(x) - f(z) + f(z) - f(y) && (12) \\
&\leqslant \nabla f(x)^\top (x - z) + \nabla f(y)^\top (z - y) + \frac{\beta}{2}\|z - y\|^2 && (13) \\
&= \nabla f(x)^\top (x - y) + (\nabla f(x) - \nabla f(y))^\top (y - z) + \frac{1}{2\beta}\|\nabla f(x) - \nabla f(y)\|^2 && (14) \\
&= \nabla f(x)^\top (x - y) - \frac{1}{2\beta}\|\nabla f(x) - \nabla f(y)\|^2 && (15)
\end{aligned}$$

■

We will show that gradient descent with the update rule

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

attains a faster rate of convergence under the smoothness condition.

Theorem 2.10. *Let f be convex and β-smooth on $\mathbb{R}^n$ then gradient descent with $\eta = \frac{1}{\beta}$ satisfies*

$$f(x_t) - f(x^*) \leqslant \frac{2\beta \|x_1 - x^*\|^2}{t-1}$$

To prove this we will need the following two lemmas.

Proof. By the update rule and lemma 2.8 we have

$$f(x_{s+1}) - f(x_s) \leqslant -\frac{1}{2\beta} \|\nabla f(x_s)\|^2$$

In particular, denoting $\delta_s = f(x_s) - f(x^*)$ this shows

$$\delta_{s+1} \leqslant \delta_s - \frac{1}{2\beta} \|\nabla f(x_s)\|^2$$

One also has by convexity

$$\delta_s \leqslant \nabla f(x)s)^\top (x_s - x^*) \leqslant \|x_s - x^*\| \cdot \|\nabla f(x_s)\|$$

We will prove that $\|x_s - x^*\|$ is decreasing with s, which with the two above displays will imply

$$\delta_{s+1} \leqslant \delta_s - \frac{1}{2\beta \|x_1 - x^*\|^2} \delta_s^2$$

We solve the recurrence as follows. Let $w = \frac{1}{2\beta \|x_1 - x^*\|^2}$, then

$$w\delta_s^2 + \delta_{s+1} \leqslant \delta_s \iff w\frac{\delta_s}{\delta_{s+1}} + \frac{1}{\delta_s} \leqslant \frac{1}{\delta_{s+1}} \implies \frac{1}{\delta_{s+1}} - \frac{1}{\delta_s} \geqslant w \implies \frac{1}{\delta_t} \geqslant w(t-1)$$

To finish the proof it remains to show that $\|x_s - x^*\|$ is decreasing with s. Using lemma 2.9 one immediately gets

$$(\nabla f(x) - \nabla f(y))^\top (x - y) \geqslant \frac{1}{\beta} \|\nabla f(x) - \nabla f(y)\|^2$$

We use this and the fact that $\nabla f(x^*) = 0$

$$\begin{aligned}
\|x_{s+1} - x^*\|^2 &= \|x_s - \frac{1}{\beta} \nabla f(x_s) - x^*\|^2 && (16)\\
&= \|x_s - x^*\|^2 - \frac{2}{\beta} \nabla f(x_s)^\top (x_s - x^*) + \frac{1}{\beta^2} \|\nabla f(x_s)\|^2 && (17)\\
&\leqslant \|x_s - x^*\|^2 - \frac{1}{\beta^2} \|\nabla f(x_s)\|^2 && (18)\\
&\leqslant \|x_s - x^*\|^2 && (19)
\end{aligned}$$

which concludes the proof. ■

3 Lecture 3: Strong convexity

This lecture introduces the notion of α-strong convexity and combines it with β-smoothness to develop the concept of condition number. Adding an assumption of, respectively, strong convexity or conditioning improves the rates of error decay for gradient descent proved in the previous lecture from $O(1/\sqrt{t})$ to $O(1/t)$ and $O(1/t)$ to $O(\mathrm{e}^{-t})$.

The technical part follows the corresponding chapter in Bubeck's text [Bub15].

3.1 Reminders

Recall that we had (at least) two definitions apiece for convexity and smoothness: a general definition for all functions and a more compact definition for (twice-)differentiable functions.

A function f is convex if, for each input, there exists a globally valid *linear* lower bound on the function: $f(y) \geqslant f(x) + g^\top(x)(y - x)$. For differentiable functions, the role of g is played by the gradient.

A function f is β-smooth if, for each input, there exists a globally valid *quadratic* upper bound on the function, with (finite) quadratic parameter β: $f(y) \leqslant f(x) + g^\top(x)(y - x) + \frac{\beta}{2}\|x - y\|^2$. More poetically, a smooth, convex function is "trapped between a parabola and a line". Since β is covariant with affine transformations, e.g. changes of units of measurement, we will frequently refer to a β-smooth function as simply smooth.

For twice-differentiable functions, these properties admit simple conditions for smoothness in terms of the Hessian, or matrix of second partial derivatives. A $\mathcal{D}^2$ function f is convex if $\nabla^2 f(x) \succeq 0$ and it is β-smooth if $\nabla^2 f(x) \preceq \beta I$.

We furthermore defined the notion of L-Lipschitzness. A function f is L-Lipschitz if the amount that it "stretches" its inputs is bounded by L: $|f(x) - f(y)| \leqslant L\|x - y\|$. Note that for differentiable functions, β-smoothness is equivalent to β-Lipschitzness of the gradient.

3.2 Strong convexity

With these three concepts, we were able to prove two error decay rates for gradient descent (and its projective, stochastic, and subgradient flavors). However, these rates were substantially slower than what's observed in certain settings in practice.

Noting the asymmetry between our linear lower bound (from convexity) and our quadratic upper bound (from smoothness) we introduce a new, more restricted function class by upgrading our lower bound to second order.

Definition 3.1 (α-Strong Convexity)**.** A function $f : \Omega \to \mathbb{R}$ is *α-strongly convex* if, for all $x, y \in \Omega$, the following inequality holds for some $\alpha > 0$:

$$f(y) \geqslant f(x) + g(x)^\top(y - x) + \frac{\alpha}{2}\|x - y\|^2$$

As with smoothness, we will often shorten "α-strongly convex" to "strongly convex". A strongly convex, smooth function is one that can be "squeezed between two parabolas". If β-smoothness is a good thing, then α-convexity guarantees we don't have too much of a good thing.

Once again, twice-differentiable functions afford a quick condition: a D^2 function is *alpha*-strongly convex if $\nabla^2 f(x) \succeq \alpha I$.

Once again, note that α changes under affine transformations. Conveniently enough, for α-strongly convex, β-smooth functions, we can define a basis-independent quantity that combines these properties:

Definition 3.2 (Condition Number)**.** An α-strongly convex, β-smooth function f has *condition number* $\frac{\beta}{\alpha}$.

For a positive-definite quadratic function f, this definition of the condition number corresponds with the perhaps more familiar definition of the condition number of a matrix.

3.3 A look back and ahead

The following table summarizes the results from the previous lecture and the results to be obtained in this lecture. In both, the value ϵ is the difference between f at some value x' computed from the outputs of gradient descent and f calculated at an optimizer x^*.

	Convex	Strongly Convex
Lipschitz	$\epsilon \leqslant O(1/\sqrt{t})$	$\epsilon \leqslant O(1/t)$
Smooth	$\epsilon \leqslant O(1/t)$	$\epsilon \leqslant O(e^{-t})$

Table 1: Bounds on error ϵ as a function of number of steps taken t for gradient descent applied to various classes of functions.

Since a rate that is exponential in terms of the magnitude of the error is linear in terms of the bit precision, this rate of convergence is termed *linear*. We now move to prove these rates.

3.4 Convergence rate strongly convex functions

Theorem 3.3. *Assume $f\colon \Omega \to \mathbb{R}$ is* α-strongly convex *and* L-Lipschitz*. Let x^* be an optimizer of f, and let x_s be the updated point at step s using projected gradient descent. Let the max number of iterations be t with an adaptive step size $\eta_s = \frac{2}{\alpha(s+1)}$, then*

$$f\left(\sum_{s=1}^{t} \frac{2s}{t(t+1)} x_s\right) - f(x^*) \leqslant \frac{2L^2}{\alpha(t+1)}$$

The theorem implies the convergence rate of projected gradient descent for α-strongly convex functions is similar to that of β-smooth functions with a bound on error $\epsilon \leqslant O(1/t)$. In order to prove Theorem 3.3, we need the following proposition.

Proposition 3.4 (Jensen's inequality)**.** *Assume $f\colon \Omega \to \mathbb{R}$ is a convex function and $x_1, x_2, ...,$ $, x_n, \sum_{i=1}^n \gamma_i x_i / \sum_{i=1}^n \gamma_i \in \Omega$ with weights $\gamma_i > 0$, then*

$$f\left(\frac{\sum_{i=1}^n \gamma_i x_i}{\sum_{i=1}^n \gamma_i}\right) \leqslant \frac{\sum_{i=1}^n \gamma_i f(x_i)}{\sum_{i=1}^n \gamma_i}$$

For a graphical "proof" follow this link.

Proof of Theorem 3.3. Recall the two steps update rule of projected gradient descent

$$y_{s+1} = x_s - \eta_s \nabla f(x_s)$$
$$x_{s+1} = \Pi_\Omega(y_{s+1})$$

First, the proof begins by exploring an upper bound of difference between function values $f(x_s)$ and $f(x^*)$.

$$\begin{aligned}
f(x_s) - f(x^*) &\leqslant \nabla f(x_s)^\top (x_s - x^*) - \frac{\alpha}{2}\|x_s - x^*\|^2 \\
&= \frac{1}{\eta_s}(x_s - y_{s+1})^\top (x_s - x^*) - \frac{\alpha}{2}\|x_s - x^*\|^2 && \text{(by update rule)} \\
&= \frac{1}{2\eta_s}(\|x_s - x^*\|^2 + \|x_s - y_{s+1}\|^2 - \|y_{s+1} - x^*\|^2) - \frac{\alpha}{2}\|x_s - x^*\|^2 \\
& && \text{(by "Fundamental Theorem of Optimization")} \\
&= \frac{1}{2\eta_s}(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2) + \frac{\eta_s}{2}\|\nabla f(x_s)\|^2 - \frac{\alpha}{2}\|x_s - x^*\|^2 \\
& && \text{(by update rule)} \\
&\leqslant \frac{1}{2\eta_s}(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2) + \frac{\eta_s}{2}\|\nabla f(x_s)\|^2 - \frac{\alpha}{2}\|x_s - x^*\|^2 \\
& && \text{(by Lemma 2.5)} \\
&\leqslant (\frac{1}{2\eta_s} - \frac{\alpha}{2})\|x_s - x^*\|^2 - \frac{1}{2\eta_s}\|x_{s+1} - x^*\|^2 + \frac{\eta_s L^2}{2} && \text{(by Lipschitzness)}
\end{aligned}$$

By multiplying s on both sides and substituting the step size η_s by $\frac{2}{\alpha(s+1)}$, we get

$$s(f(x_s) - f(x^*)) \leqslant \frac{L^2}{\alpha} + \frac{\alpha}{4}(s(s-1)\|x_s - x^*\|^2 - s(s+1)\|x_{s+1} - x^*\|^2)$$

Finally, we can find the upper bound of the function value shown in Theorem 3.3 obtained using t steps projected gradient descent

$$\begin{aligned}
f\left(\sum_{s=1}^{t} \frac{2s}{t(t+1)} x_s\right) &\leqslant \sum_{s=1}^{t} \frac{2s}{t(t+1)} f(x_s) && \text{(by Proposition 3.4)} \\
&\leqslant \frac{2}{t(t+1)} \sum_{s=1}^{t} \left(s f(x^*) + \frac{L^2}{\alpha} + \frac{\alpha}{4}(s(s-1)\|x_s - x^*\|^2 - s(s+1)\|x_{s+1} - x^*\|^2) \right) \\
&= \frac{2}{t(t+1)} \sum_{s=1}^{t} s f(x^*) + \frac{2L^2}{\alpha(t+1)} - \frac{\alpha}{2}\|x_{t+1} - x^*\|^2 && \text{(by telescoping sum)} \\
&\leqslant f(x^*) + \frac{2L^2}{\alpha(t+1)}
\end{aligned}$$

This concludes that solving an optimization problem with a strongly convex objective function with projected gradient descent has a convergence rate is of the order $\frac{1}{t+1}$, which is faster compared to the case purely with Lipschitzness. ■

3.5 Convergence rate for smooth and strongly convex functions

Theorem 3.5. *Assume $f : \mathbb{R}^n \to \mathbb{R}$ is α-*strongly convex *and β-*smooth*. Let x^* be an optimizer of f, and let x_t be the updated point at step t using gradient descent with a constant step size $\frac{1}{\beta}$, i.e. using the update rule $x_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t)$. Then,*

$$\|x_{t+1} - x^*\|^2 \leqslant \exp\left(-t\frac{\alpha}{\beta}\right)\|x_1 - x^*\|^2$$

In order to prove Theorem 3.5, we require use of the following lemma.

Lemma 3.6. *Assume f as in Theorem 3.5. Then $\forall x, y \in \mathbb{R}^n$ and an update of the form $x^+ = x - \frac{1}{\beta}\nabla f(x)$,*

$$f(x^+) - f(y) \leqslant \nabla f(x)^\top (x - y) - \frac{1}{2\beta}\|\nabla f(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2$$

Proof of Lemma 3.6.

$$\begin{aligned}
f(x^+) - f(x) + f(x) - f(y) &\leqslant \nabla f(x)^\top (x^+ - x) + \frac{\beta}{2}\|x^+ - x\|^2 && \text{(Smoothness)} \\
&\quad + \nabla f(x)^\top (x - y) - \frac{\alpha}{2}\|x - y\|^2 && \text{(Strong convexity)} \\
&= \nabla f(x)^\top (x^+ - y) + \frac{1}{2\beta}\|\nabla f(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2 \\
& && \text{(Definition of } x^+\text{)} \\
&= \nabla f(x)^\top (x - y) - \frac{1}{2\beta}\|\nabla f(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2 \\
& && \text{(Definition of } x^+\text{)}
\end{aligned}$$

■

Now with Lemma 3.6 we are able to prove Theorem 3.5.

Proof of Theorem 3.5.

$$\begin{aligned}
\|x_{t+1} - x^*\|^2 &= \|x_t - \frac{1}{\beta}\nabla f(x_t) - x^*\|^2 \\
&= \|x_t - x^*\|^2 - \frac{2}{\beta}\nabla f(x_t)^\top (x_t - x^*) + \frac{1}{\beta^2}\|\nabla f(x_t)\|^2 \\
&\leqslant (1 - \frac{\alpha}{\beta})\|x_t - x^*\|^2 && \text{(Use of Lemma 3.6 with } y = x^*, x = x_t\text{)} \\
&\leqslant (1 - \frac{\alpha}{\beta})^t\|x_1 - x^*\|^2 \\
&\leqslant \exp\left(-t\frac{\alpha}{\beta}\right)\|x_1 - x^*\|^2
\end{aligned}$$

■

We can also prove the same result for the constrained case using projected gradient descent.

Theorem 3.7. *Assume $f : \Omega \to \mathbb{R}$ is α-*strongly convex *and β-*smooth*. Let x^* be an optimizer of f, and let x_t be the updated point at step t using projected gradient descent with a constant step size $\frac{1}{\beta}$, i.e. using the update rule $x_{t+1} = \Pi_\Omega(x_t - \frac{1}{\beta}\nabla f(x_t))$ where Π_Ω is the projection operator. Then,*

$$\|x_{t+1} - x^*\|^2 \leqslant \exp\left(-t\frac{\alpha}{\beta}\right)\|x_1 - x^*\|^2$$

As in Theorem 3.5, we will require the use of the following Lemma in order to prove Theorem 3.7.

Lemma 3.8. *Assume f as in Theorem 3.5. Then $\forall x, y \in \Omega$, define $x^+ \in \Omega$ as $x^+ = \Pi_\Omega(x - \frac{1}{\beta}\nabla f(x))$ and the function $g : \Omega \to \mathbb{R}$ as $g(x) = \beta(x - x^+)$. Then*

$$f(x^+) - f(y) \leqslant g(x)^\top (x - y) - \frac{1}{2\beta}\|g(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2$$

Proof of Lemma 3.8. The following is given by the Projection Lemma, for all x, x^+, y defined as in Theorem 3.7.

$$\nabla f(x)^\top (x^+ - y) \leqslant g(x)^\top (x^+ - y)$$

Therefore, following the form of the proof of Lemma 3.6,

$$\begin{aligned} f(x^+) - f(x) + f(x) - f(y) &\leqslant \nabla f(x)^\top (x^+ - y) + \frac{1}{2\beta}\|\nabla g(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2 \\ &\leqslant \nabla g(x)^\top (x^+ - y) + \frac{1}{2\beta}\|\nabla g(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2 \\ &= \nabla g(x)^\top (x - y) - \frac{1}{2\beta}\|\nabla g(x)\|^2 - \frac{\alpha}{2}\|x - y\|^2 \end{aligned}$$

■

The proof of Theorem 3.7 is exactly as in Theorem 3.5 after substituting the appropriate projected gradient descent update in place of the standard gradient descent update, with Lemma 3.8 used in place of Lemma 3.6.

4 Lecture 4: Some applications of gradient methods

This lecture was a sequence of code examples that you can find here:

Lecture 4
(opens in your browser)

5 Lecture 5: Conditional gradient method (Frank-Wolfe)

In this lecture we discuss the conditional gradient method, also known as the Frank-Wolfe (FW) algorithm [FW56]. The motivation for this approach is that the projection step in projected gradient descent can be computationally inefficient in certain scenarios. The conditional gradient method provides an appealing alternative.

5.1 The algorithm

Conditional gradient side steps the projection step using a clever idea.

We start from some point $x_0 \in \Omega$. Then, for time steps $t = 1$ to T, where T is our final time step, we set

$$x_{t+1} = x_t + \eta_t(\bar{x}_t - x_t)$$

where

$$\bar{x}_t = \arg\min_{x \in \Omega} f(x_t) + \nabla f(x_t)^\top (x - x_t).$$

This expression simplifies to:

$$\bar{x}_t = \arg\min_{x \in \Omega} \nabla f(x_t)^\top x$$

Note that we need step size $\eta_t \in [0, 1]$ to guarantee $x_{t+1} \in \Omega$.

So, rather than taking a gradient step and projecting onto the constraint set. We optimize a liner function (defined by the gradient) inside the constraint set.

5.2 Conditional gradient convergence analysis

As it turns out, conditional gradient enjoys a convergence guarantee similar to the one we saw for projected gradient descent.

Theorem 5.1 (Convergence Analysis)**.** *Assume we have a function $f\colon \Omega \to \mathbb{R}$ that is convex, β-smooth and attains its global minimum at a point $x^* \in \Omega$. Then, Frank-Wolfe achieves*

$$f(x_t) - f(x^*) \leqslant \frac{2\beta D^2}{t+2}$$

with step size

$$\eta_t = \frac{2}{t+2}.$$

Here, D is the diameter of Ω, defined as $D = \max_{x-y \in \Omega} \|x - y\|$.

Note that we can trade our assumption of the existence of x^* for a dependence on L, the Lipschitz constant, in our bound.

Proof of Theorem 5.1. By smoothness and convexity, we have

$$f(y) \leqslant f(x) + \nabla f(x)^\top (x - x_t) + \frac{\beta}{2}\|x - y\|^2$$

Letting $y = x_{t+1}$ and $x = x_t$, combined with the progress rule of conditional gradient descent, the above equation yields:

$$f(x_{t+1}) \leqslant f(x_t) + \eta_t \nabla f(x_t)^\top (\bar{x}_t - x_t) + \frac{\eta_t^2 \beta}{2}\|\bar{x}_t - x_t\|^2$$

We now recall the definition of D from Theorem 5.1 and observe that $\|\bar{x}_t - x_t\|^2 \leqslant D^2$. Thus, we rewrite the inequality:

$$f(x_{t+1}) \leqslant f(x_t) + \eta_t \nabla f(x_t)^\top (x_t^* - x_t) + \frac{\eta_t^2 \beta D^2}{2}$$

Because of convexity, we also have that

$$\nabla f(x_t)^\top (x^* - x_t) \leqslant f(x^*) - f(x_t)$$

Thus,

$$f(x_{t+1}) - f(x^*) \leqslant (1 - \eta_t)(f(x_t) - f(x^*)) + \frac{\eta_t^2 \beta D^2}{2} \tag{20}$$

We use induction in order to prove $f(x_t) - f(x^*) \leqslant \frac{2\beta D^2}{t+2}$ based on Equation 20 above.
First step: Base Case $t = 0$
Since $f(x_{t+1}) - f(x^*) \leqslant (1 - \eta_t)(f(x_t) - f(x^*)) + \frac{\eta_t^2 \beta D^2}{2}$, when t=0, $\eta_t = \frac{2}{0+2} = 1, then$

$$\begin{aligned}
f(x_1) - f(x^*) &\leqslant (1 - \eta_t)(f(x_t) - f(x^*)) + \frac{\beta}{2}\|x_1 - x^*\|^2 \\
&= (1-1)(f(x_t) - f(x^*)) + \frac{\beta}{2}\|x_1 - x^*\|^2 \\
&\leqslant \frac{\beta D^2}{2} \\
&\leqslant \frac{2\beta D^2}{3}
\end{aligned}$$

Thus, the induction hypothesis holds for our base case.

Proceeding by induction, we assume that $f(x_t) - f(x^*) \leqslant \frac{2\beta D^2}{t+2}$ holds for all integers up to t and we show the claim for $t + 1$.

By Equation 20,

$$\begin{aligned}
f(x_{t+1}) - f(x^*) &\leqslant \left(1 - \frac{2}{t+2}\right)(f(x_t) - f(x^*)) + \frac{4}{2(t+2)}\beta D^2 \\
&\leqslant \left(1 - \frac{2}{t+2}\right)\frac{2\beta D^2}{t+2} + \frac{4}{2(t+2)}\beta D^2 \\
&= \beta D^2 \left(\frac{2t}{(t+2)^2} + \frac{2}{(t+2)^2}\right) \\
&= 2\beta D^2 \cdot \frac{t+1}{(t+2)^2} \\
&= 2\beta D^2 \cdot \frac{t+1}{t+2} \cdot \frac{1}{t+2} \\
&\leqslant 2\beta D^2 \cdot \frac{t+2}{t+3} \cdot \frac{1}{t+2} \\
&= 2\beta D^2 \frac{1}{t+3}
\end{aligned}$$

Thus, the inequality also holds for the $t + 1$ case.

■

5.3 Application to nuclear norm optimization problems

The code for the following examples can be found here.

5.3.1 Nuclear norm projection

The *nuclear norm* (sometimes called *Schatten* 1-*norm* or *trace norm*) of a matrix A, denoted $\|A\|_*$, is defined as the sum of its singular values

$$\|A\|_* = \sum_i \sigma_i(A)\,.$$

The norm can be computed from the singular value decomposition of A. We denote the unit ball of the nuclear norm by

$$B_*^{m\times n} = \{A \in \mathbb{R}^{m\times n} \mid \|A\|_* \leqslant 1\}.$$

How can we project a matrix A onto B_*? Formally, we want to solve

$$\min_{X\in B_*} \|A - X\|_F^2$$

Due to the rotational invariance of the Frobenius norm, the solution is obtained by projecting the singular values onto the unit simplex. This operation corresponds to shifting all singular values by the same parameter θ and clipping values at 0 so that the sum of the shifted and clipped values is equal to 1. This algorithm can be found in [DSSSC08].

5.3.2 Low-rank matrix completion

Suppose we have a partially observable matrix Y, of which the missing entries are filled with 0 and we would like to find its completion form projected on a nuclear norm ball. Formally we have the objective function

$$\min_{X\in B_*} \frac{1}{2}\|Y - P_O(X)|_F^2$$

where P_O is a linear projection onto a subset of coordinates of X specified by O. In this example $P_O(X)$ will generate a matrix with corresponding observable entries as in Y while other entries being 0. We can have $P_O(X) = X \odot O$ where O is a matrix with binary entries. Calculate the gradient of this function we will have

$$\nabla f(X) = Y - X \odot O$$

We can use projected gradient descent to solve this problem but it is more efficient to use Frank-Wolfe algorithm. We need to solve the linear optimization oracle

$$\bar{X}_t \in \arg\min_{X\in B_*} \nabla f(X_t)^\top X$$

To simplify this problem, we need a simple fact that follows from the singular value decomposition.

Fact 5.2. *The unit ball of the nuclear norm is the convex hull of rank-*1 *matrices*

$$\mathrm{conv}\{uv^\top \,|\, \|u\| = \|v\| = 1, u \in \mathbb{R}^m, v \in \mathbb{R}^n\} = \{X \in \mathbb{R}^{m\times n} \mid \|X\|_* = 1\}\,.$$

From this fact it follows that the minimum of $\nabla f(X_t)^\top X$ is attained at a rank-1 matrix $uv^\top$ for unit vectors u and v. Equivalently, we can maximize $-\nabla f(X_t)^\top uv^\top$ over all unit vectors u and v. Put $Z = -\nabla f(X_T)$ and note that

$$Z^\top uv^\top = \mathrm{tr}(Z^\top uv^\top) = \mathrm{tr}(u^\top Zv) = u^\top Zv\,.$$

Another way to see this is to note that the dual norm of a nuclear norm is operator norm,

$$\|Z\| = \max_{\|X\|_* \leqslant 1} \langle Z, X\rangle\,.$$

Either way, we see that to run Frank-Wolfe over the nuclear norm ball we only need a way to compute the top left and singular vectors of a matrix. One way of doing this is using the classical power method:

- Pick a random unit vector x_1 and let $y_1 = A^\top x / \|A^\top x\|$.
- From $k = 1$ to $k = T - 1$:
  - Put $x_{k+1} = \frac{Ay_k}{\|Ay_k\|}$
  - Put $y_{k+1} = \frac{A^\top x_{k+1}}{\|A^\top x_{k+1}\|}$
- Return x_T and y_T as approximate top left and right singular vectors.

6 Lecture 6: Discovering acceleration

In this lecture, we seek to find methods that converge faster than those discussed in previous lectures. To derive this accelerated method, we start by considering the special case of optimizing quadratic functions. Our exposition loosely follows Chapter 17 in Lax's excellent text [Lax07].

6.1 Quadratics

Definition 6.1 (Quadratic function)**.** A quadratic function $f\colon \mathbb{R}^n \to \mathbb{R}$ takes the form:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c,$$

where $A \in S^n$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$.

Note that substituting $n = 1$ into the above definition recovers the familiar univariate quadratic function $f(x) = ax^2 + bx + c$ where $a, b, c \in \mathbb{R}$, as expected. There is one subtlety in this definition: we restrict A to be symmetric. In fact, we could allow $A \in \mathbb{R}^{n\times n}$ and this would define the same class of functions, since for any $A \in \mathbb{R}^{n\times n}$ there is a symmetric matrix $\tilde{A} = \frac{1}{2}\left(A + A^T\right)$ for which:

$$x^T Ax = x^T \tilde{A}x \;\forall x \in \mathbb{R}^n.$$

Restricting $A \in S^n$ ensures each quadratic function has a *unique* representation.

The gradient and Hessian of a general quadratic function take the form:

$$\nabla f(x) = Ax - b$$
$$\nabla^2 f(x) = A.$$

Note provided A is non-singular, the quadratic has a unique critical point at:

$$x^* = A^{-1}b.$$

When $A \succ 0$, the quadratic is *strictly convex* and this point is the unique global minima.

6.2 Gradient descent on a quadratic

In this section we will consider a quadratic $f(x)$ where A is positive definite, and in particular that:

$$\alpha I \preceq A \preceq \beta I,$$

for some $0 < \alpha < \beta$. This implies that f is α-strongly convex and β-smooth.

From Theorem 3.7 we know that under these conditions, gradient descent with the appropriate step size converges linearly at the rate $\exp\left(-t\frac{\alpha}{\beta}\right)$. Clearly the size of $\frac{\alpha}{\beta}$ can dramatically affect the convergence guarantee. In fact, in the case of a quadratic, this is related to the *condition number* of the matrix A.

Definition 6.2 (Condition number)**.** Let A be a real matrix. Its *condition number* (with respect to the Euclidean norm) is:

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

the ratio of its largest and smallest eigenvalues.

So in particular, we have that $\kappa(A) \leqslant \frac{\beta}{\alpha}$; henceforth, we will assume that α, β correspond to the minimal and maximal eigenvalues of A so that $\kappa(A) = \frac{\beta}{\alpha}$. It follows that gradient descent with a constant step size $\frac{1}{\beta}$ converges at:

$$||x_{t+1} - x^*\|^2 \leqslant \exp\left(-t\frac{1}{\kappa}\right)\|x_1 - x^*\|^2.$$

In many cases, f is ill-conditioned and κ can easily take values in the hundreds or thousands. In this case, case convergence could be very slow: note that at $t > \kappa$, the error may have been reduced by only a factor of $3\times$. Can we do better than this?

In the case of a quadratic, we can of course use the analytic solution $x^* = A^{-1}b$. However, it will prove instructive to consider applying gradient descent to quadratic functions, and derive the convergence bound that we previously proved for any strongly convex smooth functions. This exercise will show us where we are losing performance, and suggest a method that can attain better guarantees.

6.2.1 Convergence analysis

Theorem 6.3. *Assume $f : \mathbb{R}^n \to \mathbb{R}$ is a quadratic where the quadratic coefficient matrix has a condition number κ. Let x^* be an optimizer of f, and let x_t be the updated point at step t using gradient descent with a constant step size $\frac{1}{\beta}$, i.e. using the update rule $x_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t)$. Then:*

$$\|x_{t+1} - x^*\|^2 \leqslant \exp\left(-t\frac{1}{\kappa}\right)\|x_1 - x^*\|^2.$$

Proof. Write:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c,$$

where $A \in S^n$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. A gradient descent update with step size η_t takes the form:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t) = x_t - \eta_t\left(Ax_t - b\right)$$

Subtracting x^* from both sides of this equation and using the property that $Ax^* - b = \nabla f(x^*) = 0$:

$$\begin{aligned} x_{t+1} - x^* &= \left(x_t - \eta_t\left(Ax_t - b\right)\right) - \left(x^* - \eta_t\left(Ax^* - b\right)\right) \\ &= (I - \eta_t A)(x_t - x^*) \\ &= \prod_{k=1}^{t}(I - \eta_k A)(x_1 - x^*)\,. \end{aligned}$$

Thus,

$$\|x_{t+1} - x^*\|_2 \leqslant \left\|\prod_{k=1}^{t}(I - \eta_t A)\right\|_2 \|x_1 - x^*\|_2 \leqslant \left(\prod_{k=1}^{t}\|I - \eta_k A\|_2\right)\|x_1 - x^*\|_2\,.$$

Set $\eta_k = \frac{1}{\beta}$ for all k. Note that $\frac{\alpha}{\beta}I \preceq \frac{1}{\beta}A \preceq I$, so:

$$\max_{A\in\mathbb{R}^{n\times n}}\left\|I - \frac{1}{\beta}A\right\|_2 = 1 - \frac{\alpha}{\beta} = 1 - \frac{1}{\kappa}.$$

It follows that

$$\|x_{t+1} - x^*\|_2 \leqslant \left(1 - \frac{1}{\kappa}\right)^t \|x_1 - x^*\|_2 \leqslant \exp\left(-\frac{t}{\kappa}\right)\|x_1 - x^*\|_2\,.$$

■

6.3 Connection to polynomial approximation

In the previous section, we proved an upper bound on the convergence rate. In this section, we would like to improve on this. To see how, think about whether there was any point in the argument above where we were careless? One obvious candidate is that our choice of step size,

$\eta_k = \frac{1}{\beta}$, was chosen rather arbitrarily. In fact, by choosing the sequence η_k we can select *any* degree-t polynomial of the form:

$$p(A) = \prod_{k=1}^{t} (I - \eta_k A) .$$

Note that:

$$\|p(A)\| = \max_{x \in \lambda(A)} |p(x)|$$

where $p(A)$ is a matrix polynomial, and $p(t)$ is the corresponding scalar polynomial. In general, we may not know the set of eigenvalues $\lambda(A)$, but we do know that all eigenvalues are in the range $[\alpha, \beta]$. Relaxing the upper bound, we get

$$\|p(A)\| \leqslant \max_{x \in [\alpha, \beta]} |p(x)| .$$

We can see now that we want a polynomial $p(a)$ that takes on small values in $[\alpha, \beta]$, while satisfying the additional normalization constraint $p(0) = 1$.

6.3.1 A simple polynomial solution

A naive solution chooses a uniform step size $\eta_t = \frac{2}{\alpha+\beta}$. Note that

$$\max_{x \in [\alpha, \beta]} \left| 1 - \frac{2}{\alpha + \beta} x \right| = \frac{\beta - \alpha}{\alpha + \beta} \leqslant \frac{\beta}{\alpha} = \kappa,$$

recovering the same convergence rate we proved previously. The resulting polynomial $p_t(x)$

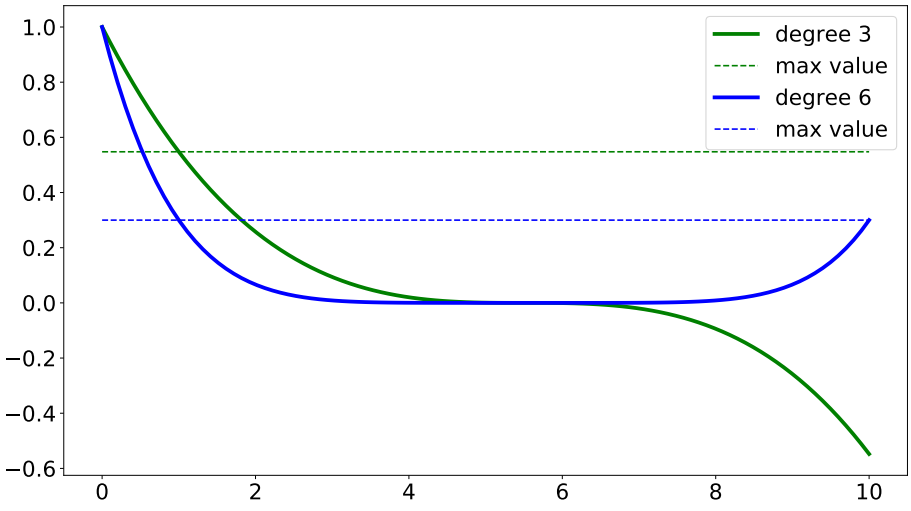


Figure 2: Naive Polynomial

is plotted in Figure 2 for degrees $t = 3$ and $t = 6$, with $\alpha = 1$ and $\beta = 10$. Note that doubling the degree from three to six only halves the maximum absolute value the polynomial attains in $[\alpha, \beta]$, explaining why convergence is so slow.

6.4 Chebyshev polynomials

Fortunately, we can do better than this by speeding up gradient descent using Chebyshev polynomials. We will use Chebyshev polynomials of the first kind, defined by the recurrence relation:

$$T_0(a) = 1, \quad T_1(a) = a$$
$$T_k(a) = 2aT_{k-1}(a) - T_{k-2}(a), \text{ for } k \geqslant 2\,.$$

Figure 3 plots the first few Chebyshev polynomials.

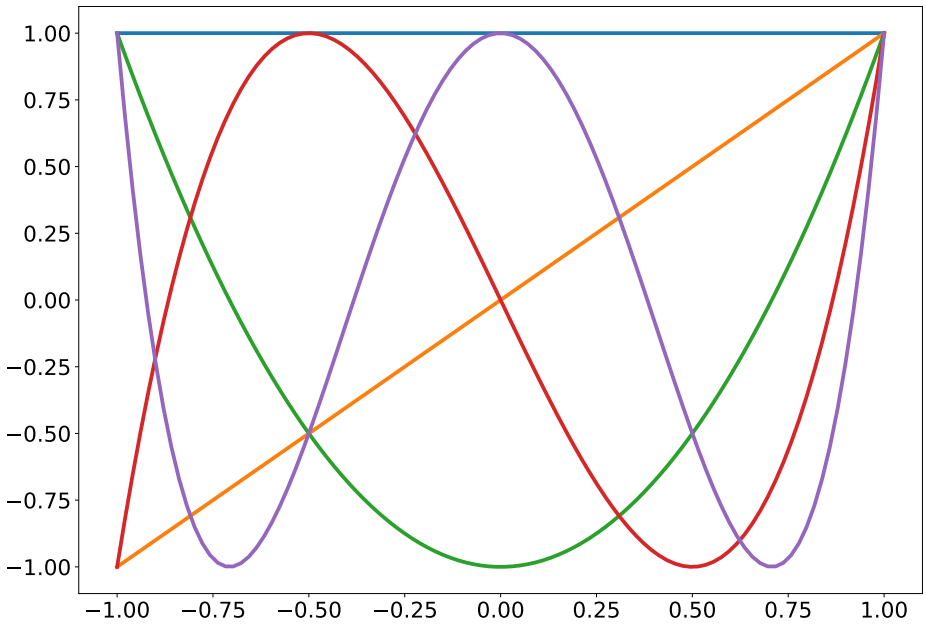


Figure 3: Chebychev polynomials of varying degrees.

Why Chebyshev polynomials? Suitably rescaled, they minimize the absolute value in a desired interval $[\alpha, \beta]$ while satisfying the normalization constraint of having value 1 at the origin.

Recall that the eigenvalues of the matrix we consider are in the interval $[\alpha, \beta]$. We need to rescale the Chebyshev polynomials so that they're supported on this interval and still attain value 1 at the origin. This is accomplished by the polynomial

$$P_k(a) = \frac{T_k\left(\frac{\alpha+\beta-2a}{\beta-\alpha}\right)}{T_k\left(\frac{\alpha+\beta}{\beta-\alpha}\right)}.$$

We see on figure 4 that doubling the degree has a much more dramatic effect on the magnitude of the polynomial in the interval $[\alpha, \beta]$.

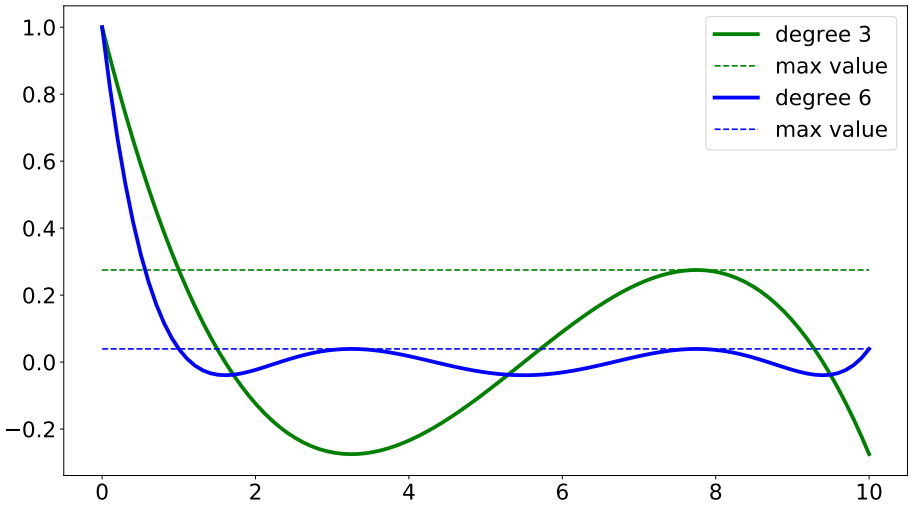


Figure 4: Rescaled Chebyshev

Let's compare on figure 5 this beautiful Chebyshev polynomial side by side with the naive polynomial we saw earlier. The Chebyshev polynomial does much better: at around 0.3 for degree 3 (needed degree 6 with naive polynomial), and below 0.1 for degree 6.

6.4.1 Accelerated gradient descent

The Chebyshev polynomial leads to an accelerated version of gradient descent. Before we describe the iterative process, let's first see what error bound comes out of the Chebyshev polynomial.

So, just how large is the polynomial in the interval $[\alpha, \beta]$? First, note that the maximum value is attained at α. Plugging this into the definition of the rescaled Chebyshev polynomial, we get the upper bound for any $a \in [\alpha, \beta]$,

$$|P_k(a)| \leqslant |P_k(\alpha)| = \frac{|T_k(1)|}{|T_K\left(\frac{\beta+\alpha}{\beta-\alpha}\right)|} \leqslant \left| T_K\left(\frac{\beta+\alpha}{\beta-\alpha}\right)^{-1} \right|.$$

Recalling the condition number $\kappa = \beta/\alpha$, we have

$$\frac{\beta+\alpha}{\beta-\alpha} = \frac{\kappa+1}{\kappa-1}.$$

Typically κ is large, so this is $1+\epsilon$, $\epsilon \approx \frac{2}{\kappa}$. Therefore, we have

$$|P_k(a)| \quad \leqslant |T_k(1+\epsilon)^{-1}|.$$

To upper bound $|P_k|$, we need to lower bound $|T_k(1+\epsilon)|$.

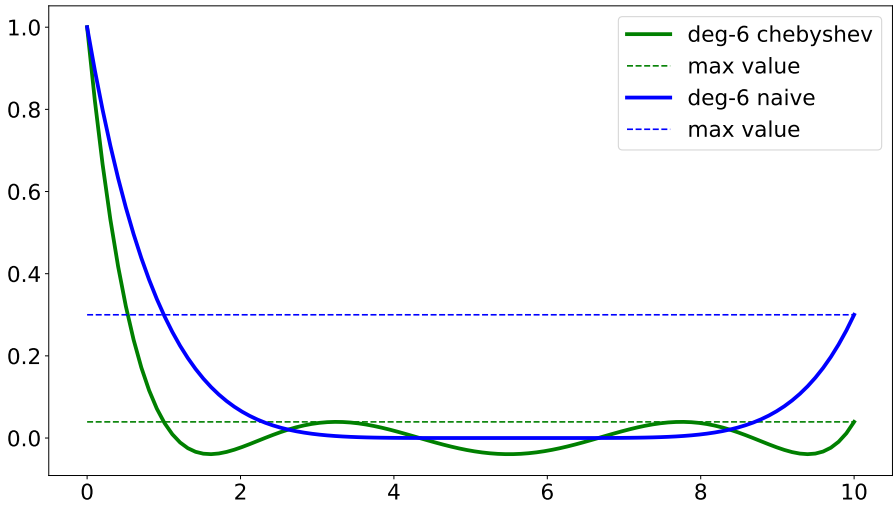


Figure 5: Rescaled Chebyshev VS Naive Polynomial

Fact: for $a > 1$, $T_k(a) = \cosh(k \cdot \operatorname{arccosh}(a))$ where:

$$\cosh(a) = \frac{e^a + e^{-a}}{2}, \quad \operatorname{arccosh}(a) = \ln\left(x + \sqrt{x^2 - 1}\right).$$

Now, letting $\phi = \operatorname{arccosh}(1+\epsilon)$:

$$e^{\phi} = 1 + \epsilon + \sqrt{2\epsilon + \epsilon^2} \geqslant 1 + \sqrt{\epsilon}.$$

So, we can lower bound $|T_k(1+\epsilon)|$:

$$\begin{aligned} |T_k(1+\epsilon)| &= \cosh(k\operatorname{arccosh}(1+\epsilon)) \\ &= \cosh(k\phi) \\ &= \frac{(e^{\phi})^k + (e^{-\phi})^k}{2} \\ &\geqslant \frac{(1+\sqrt{\epsilon})^k}{2}. \end{aligned}$$

Then, the reciprocal is what we needed to upper bound the error of our algorithm, so we have:

$$|P_k(a)| \leqslant |T_k(1+\epsilon)^{-1}| \leqslant 2(1+\sqrt{\epsilon})^{-k}.$$

Thus, this establishes that the Chebyshev polynomial achieves the error bound:

$$\begin{aligned} \|x_{t+1} - x^*\| &\leqslant 2(1+\sqrt{\epsilon})^{-t} \|x_0 - x^*\| \\ &\approx 2\left(1 + \sqrt{\frac{2}{\kappa}}\right)^{-t} \|x_0 - x^*\| \\ &\leqslant 2\exp\left(-t\sqrt{\frac{2}{\kappa}}\right) \|x_0 - x^*\|. \end{aligned}$$

This means that for large κ, we get quadratic savings in the degree we need before the error drops off exponentially. Figure 6 shows the different rates of convergence, we clearly see that the

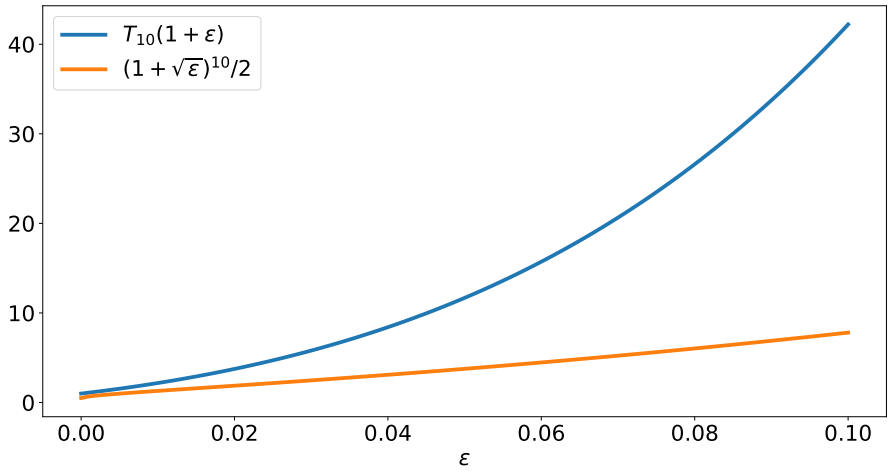


Figure 6: Convergence for naive polynomial and Chebyshev

6.4.2 The Chebyshev recurrence relation

Due to the recursive definition of the Chebyshev polynomial, we directly get an iterative algorithm out of it. Transferring the recursive definition to our rescaled Chebyshev polynomial, we have:

$$P_{K+1}(a) = (\eta_k a + \gamma_k)P_k(a) + \mu_k P_{k-1}(a).$$

where we can work out the coefficients η_k, γ_k, μ_k from the recurrence definition. Since $P_k(0) = 1$, we must have $\gamma_k + \mu_k = 1$. This leads to a simple update rule for our iterates:

$$\begin{aligned} x_{k+1} &= (\eta_k A + \gamma_k)x_k + (1 - \gamma_K)x_{k-1} - \eta_k b \\ &= (\eta_k A + (1 - \mu_k))x_k + \mu_k x_{k-1} - \eta_k b \\ &= x_k - \eta_k(Ax_k - b) + \mu_k(x_k - x_{k-1}). \end{aligned}$$

We see that the update rule above is actually very similar to plain gradient descent except for the additional term $\mu_k(x_k - x_{k-1})$. This term can be interpreted as a *momentum* term, pushing the algorithm in the direction of where it was headed before. In the next lecture, we'll dig deeper into momentum and see how to generalize the result for quadratics to general convex functions.

7 Lecture 7: Nesterov's accelerated gradient descent

Previously, we saw how we can accelerate gradient descent for minimizing quadratics $f(x) = x^\top Ax + b^\top x$, where A is a positive definite matrix. In particular, we achieved a quadratic

improvement in the dependence on the condition number of the matrix A than what standard gradient descent achieved. The resulting update rule had the form

$$x_{t+1} = x_t - \eta_t \nabla f(x_t) + \mu(x_t - x_{t-1}),$$

where we interpreted the last term as a form of "momentum". In this simple form, the update rule is sometimes called Polyak's *heavy ball method*.

To get the same accelerated convergence rate for general smooth convex functions that we saw for quadratics, we will have to work a bit harder and look into Nesterov's celebrated *accelerated gradient method* [Nes83, Nes04]

Specifically, we will see that Nesterov's method achieves a convergence rate of $\mathcal{O}\left(\frac{\beta}{t^2}\right)$ for β-smooth functions. For smooth functions which are also α-strongly convex, we will achieve a rate of $\exp\left(-\Omega\left(\sqrt{\frac{\beta}{\alpha}}t\right)\right)$.

The update rule is a bit more complicated than the plain momentum rule and proceeds as follows:

$$\begin{aligned} x_0 &= y_0 = z_0, \\ x_{t+1} &= \tau z_t + (1-\tau) y_t && (t \geqslant 0) \\ y_t &= x_t - \frac{1}{\beta}\nabla f(x_t) && (t \geqslant 1) \\ z_t &= z_{t-1} - \eta \nabla f(x_t) && (t \geqslant 1) \end{aligned}$$

Here, the parameter β is the smoothness constant of the function we're minimizing. The step size η and the parameter τ will be chosen below so as to give us a convergence guarantee.

7.1 Convergence analysis

We first show that for a simple setting of the step sizes, the algorithm reduces its initial error from some value d to $\frac{d}{2}$. We will then repeatedly restart the algorithm to continue reducing the error. This is a slight departure from Nesterov's method which does not need restrating, albeit requiring a much more delicate step size schedule that complicates the analysis.

Lemma 7.1. *Suppose $f\colon \mathbb{R}^n \to \mathbb{R}$ is a convex, β-smooth function that attains its minimum at a point $x^* \in \mathbb{R}^n$. Assume that the initial point satisfies $\|x_0 - x^*\| \leqslant R$ and $f(x_0) - f(x^*) \leqslant d$. Put $\eta = \frac{R}{\sqrt{d\beta}}$, and τ s.t. $\frac{1-\tau}{\tau} = \eta\beta$.*

Then after $T = 4R\sqrt{\frac{\beta}{d}}$ steps, we have

$$f(\bar{x}) - f(x^*) \leqslant \frac{d}{2},$$

where $\bar{x} = \frac{1}{T}\sum_{k=0}^{T-1} x_t$.

Proof. In Lecture 2, we showed the following properties for smooth and convex functions:

$$f(y_t) - f(x_t) \leqslant -\frac{1}{2\beta}\|\nabla f(x_t)\|^2 \tag{21}$$

By the "Fundamental Theorem of Optimization" (see Lecture 2), we have for all $u \in \mathbb{R}^n$:

$$\eta\langle \nabla f(x_{t+1}), z_t - u\rangle = \frac{\eta^2}{2}\|\nabla f(x_{t+1})\|^2 + \frac{1}{2}\|z_t - u\|^2 - \frac{1}{2}\|z_{t+1} - u\|^2 . \tag{22}$$

Substituting the first equation yields

$$\eta\langle \nabla f(x_{t+1}, z_t - u\rangle \leqslant \eta^2\beta(f(x_{t+1}) - f(y_{t+1})) + \frac{1}{2}\|z_t - u\|^2 - \frac{1}{2}\|z_{t+1} - u\|^2 \tag{23}$$

Working towards a term that we can turn into a telescoping sum, we compute the following difference

$$\begin{aligned}
&\eta\langle \nabla f(x_{t+1}), x_{t+1} - u\rangle - \eta\langle \nabla f(x_{t+1}), z_t - u\rangle \\
&= \eta\langle \nabla f(x_{t+1}), x_{t+1} - z_t\rangle \\
&= \frac{1-\tau}{\tau}\eta\langle \nabla f(x_{t+1}), y_t - x_{t+1}\rangle \\
&\leqslant \frac{1-\tau}{\tau}\eta(f(y_t) - f(x_{t+1})) \quad \text{(by convexity).}
\end{aligned} \tag{24}$$

Combining (23) and (24), and setting $\frac{1-\tau}{\tau} = \eta\beta$ yield for all $u \in \mathbb{R}^n$:

$$\eta\langle \nabla f(x_{t+1}), x_{t+1} - u\rangle \leqslant \eta^2\beta(f(y_t) - f(y_{t+1})) + \frac{1}{2}\|z_t - u\|^2 - \frac{1}{2}\|z_{t+1} - u\|^2.$$

Proceeding as in our basic gradient descent analysis, we apply this inequality for $u = x^*$, sum it up from $k = 0$ to T and exploit the telescoping effect:

$$\begin{aligned}
\eta T(f(\bar{x}) - f(x^*)) &\leqslant \sum_{k=0}^{T} \eta\langle \nabla f(x_{t+1}), x_{t+1} - x^*\rangle \\
&\leqslant \eta^2\beta d + R^2,
\end{aligned}$$

which can be rewritten as

$$\begin{aligned}
f(\bar{x}) - f(x^*) &\leqslant \frac{\eta\beta d}{T} + \frac{R^2}{\eta T} \\
&\leqslant \frac{2\sqrt{\beta d}}{T}R && \text{(since } \eta = R/\sqrt{\beta d}\text{)} \\
&\leqslant \frac{d}{2} && \text{(since } T \geqslant 4R\sqrt{\beta/D}.\text{)}
\end{aligned}$$

■

This lemma appears in work by Allen-Zhu and Orecchia [AZO17], who interpret Nesterov's method as a coupling of two ways of analyzing gradient descent. One is the the inequality in (21) that is commonly used in the analysis of gradient descent for smooth functions. The other is Equation 22 commonly used in the convergence analysis for non-smooth functions. Both were shown in our Lecture 2.

Theorem 7.2. *Under the assumptions of Lemma 7.1, by restarting the algorithm repeatedly, we can find a point x such that*

$$f(x) - f(x^*) \leqslant \epsilon$$

with at most $O(R\sqrt{\beta/\epsilon})$ gradient updates.

Proof. By Lemma 7.1, we can go from error d to $d/2$ with $CR\sqrt{\beta/d}$ gradient updates for some constant C. Initializing each run with the output of the previous run, we can there for successively reduce the error from an initial value d to $d/2$ to $d/4$ and so on until we reach error ϵ after $O(\log(d/\epsilon))$ runs of the algorithm. The total number of gradient steps we make is

$$CR\sqrt{\beta/d} + CR\sqrt{2\beta/d} + \cdots + CR\sqrt{\beta/\epsilon} = O\left(R\sqrt{\beta/\epsilon}\right).$$

Note that the last run of the algorithm dominates the total number of steps up to a constant factor. ■

7.2 Strongly convex case

We can prove a variant of Lemma 7.1 that applies when the function is also α-strongly convex, ultimately leading to a linear convergence rate. The idea is just a general trick to convert a convergence rate for a smooth function to a convergence rate in domain using the definition of strong convexity.

Lemma 7.3. *Under the assumption of Lemma 7.1 and the additional assumption that the function f is α-strongly convex, we can find a point x with $T = O\left(\sqrt{\frac{\beta}{\alpha}}\right)$ gradient updates such that*

$$\|\bar{x} - x^*\|^2 \leqslant \frac{1}{2}\|x_0 - x^*\|^2.$$

Proof. Noting that $\|x_0 - x^*\|^2 \leqslant R^2$, we can apply Theorem 7.2 with error parameter $\epsilon = \frac{\alpha}{4}\|x_0 - x^*\|^2$ to find a point x such that

$$f(x) - f(x^*) \leqslant \frac{\alpha}{4}\|x_0 - x^*\|^2,$$

while only making $O\left(\sqrt{\beta/\alpha}\right)$ many steps. From the definition of strong convexity it follows that

$$\frac{\alpha}{2}\|x - x^*\|^2 \leqslant f(x) - f(x^*).$$

Combining the two inequalities gives the statement we needed to show. ■

We see from the lemma that for strongly convex function we actually reduce the distance to the optimum in domain by a constant factor at each step. We can therefore repeatedly apply the lemma to get a linear convergence rate.

Table 2 compares the bounds on error $\epsilon(t)$ as a function of the total number of steps when applying Nesterov's method and ordinary gradient descent method to different functions.

	Nesterov's Method	Ordinary GD Method
β-smooth, convex	$O\left(\beta/t^2\right)$	$O\left(\beta/t\right)$
β-smooth, α-strongly convex	$\exp\left(-\Omega(t\sqrt{\alpha/\beta})\right)$	$\exp\left(-\Omega(t\alpha/\beta)\right)$

Table 2: Bounds on error ϵ as a function of number of iterations t for different methods.

8 Lecture 8: Conjugate gradients and Krylov subspaces

In this lecture, we'll develop a unified view of solving linear equations $Ax = b$ and eigenvalue problems $Ax = \lambda x$. In particular, we will justify the following picture.

	$Ax = b$	$Ax = \lambda x$
Basic	Gradient descent	Power method
Accelerated	Chebyshev iteration	Chebyshev iteration
Accelerated and step size free	Conjugate gradient	Lanczos

What we saw last time was the basic gradient descent method and Chebyshev iteration for solving quadratics. Chebyshev iteration requires step sizes to be carefully chosen. In this section, we will see how we can get a "step-size free" accelerated method, known as *conjugate gradient*.

What ties this all together is the notion of a Krylov subspace and its corresponding connection to low-degree polynomials.

Our exposition follows the excellent Chapter VI in Trefethen-Bau [TD97].

8.1 Krylov subspaces

The methods we discuss all have the property that they generate a sequence of points iteratively that is contained in a subspace called the *Kyrlov subspace*.

Definition 8.1 (Krylov subspace)**.** For a matrix $A \in \mathbb{R}^{nxn}$ and a vector $b \in \mathbb{R}^n$, the *Krylov sequence* of order t is $b, Ab, A^2b, ..., A^tb$. We define the *Krylov subspace* as

$$K_t(A, b) = \text{span}(\{b, Ab, A^2b, \ldots, A^t\}) \subseteq \mathbb{R}^n \,.$$

Krylov subspace naturally connect to polynomial approximation problems. To see this, recall that a degree t matrix polynomial is an expression of the form $p(A) = \sum_{i=1}^{t} \alpha_i A^i$.

Fact 8.2 (Polynomial connection)**.** *The Krylov subspace satisfies*

$$K_t(A, b) = \{p(A)b : deg(p) \leqslant t\} \ .$$

Proof. Note that

$$v \in K_t(A, b) \iff \exists \alpha_i : \ v = \alpha_0 b + \alpha_1 Ab + \cdots \alpha_t A^t b$$

■

From here on, suppose we have a symmetric matrix $A \in \mathbb{R}^{n\times n}$ that has orthonormal eigenvectors $u_1 \dots u_n$ and ordered eigenvalues $\lambda_1 \geqslant \lambda_2 \dots \geqslant \lambda_n$. Recall, this means

$$\begin{aligned}\langle u_i, u_j\rangle &= 0, \quad \text{for } i \neq j\\ \langle u_i, u_i\rangle &= 1\end{aligned}$$

Using that $A = \sum_i \lambda_i u_i u_i^\top$, it follows

$$p(A)u_i = p(\lambda_i)u_i\,.$$

Now suppose we write b in the eigenbasis of A as

$$b = \alpha_1 u_1 + \dots + \alpha_n u_n$$

with $\alpha_i = \langle u_i, b\rangle$. It follows that

$$p(A)b = \alpha_1 p(\lambda_1)u_1 + \alpha_2 p(\lambda_2)u_2 + \dots + \alpha_n p(\lambda_n)u_n\,.$$

8.2 Finding eigenvectors

Given these ideas, one natural approach to finding eigenvectors is to find a polynomial p such that

$$p(A)b \approx \alpha_1 u_1\,.$$

Ideally, we would have $p(\lambda_1) = 1$ and $p(\lambda_i) = 0$ for $i > 1$, but this is in general impossible unless we maket he degree of our polynomial as high as the number of distinct eigenvalues of A. Keep in mind that the degree ultimately determines the number of steps that our iterative algorithm makes. We'd therefore like to keep it as small as possible.

That's why we'll settle for an approximate solution that has $p(\lambda_1) = 1$ and makes $\max_{i>1} p(\lambda_i)$ as small as possible. This will give us a close approximation to the top eigenvalue. In practice, we don't know the value λ_1 ahead of time. What we therefore really care about is the ratio $p(\lambda_1)/p(\lambda_2)$ so that no matter what λ_1, the second eigenvalue will get mapped to a much smaller value by p.

We consider the following simple polynomial $p(\lambda) = \lambda^t$ that satisfies

$$p(\lambda_2)/p(\lambda_1) = \left(\frac{\lambda_2}{\lambda_1}\right)^t$$

In the case where $\lambda_1 = (1+\epsilon)\lambda_2$ we need $t = O(1/\epsilon)$ to make the ratio small.

The next lemma turns a small ratio into an approximation result for the top eigenvector. To state the lemma, we recall that $\tan\angle(a,b)$ is the tangent of the angle between a and b.

Lemma 8.3. $\tan\angle(p(A)b, u_1) \leqslant \max_{j>1} \frac{|p(\lambda_j)|}{|p(\lambda_1)|} \tan\angle(b, u_1)$

Proof. We define $\theta = \angle(u_1, b)$. By this, we get

$$\begin{aligned}\sin^2\theta &= \sum_{j>1} \alpha_j^2\\ \cos^2\theta &= |\alpha_1|^2\\ \tan^2\theta &= \sum_{j>1} \frac{|\alpha_j^2|}{|\alpha_1|^2}\end{aligned}$$

Now we can write:

$$\tan^2 \angle(p(A)b, u_1) = \sum_{j>1} \frac{|p(\lambda_j)\alpha_j|^2}{|p(\lambda_1)\alpha_1|^2} \leqslant \max_{j>1} \frac{|p(\lambda_j)|^2}{|p(\lambda_1)|^2} \sum_{j>1} \frac{\alpha_j|^2}{|\alpha_1|^2}$$

We note that this last sum $\sum_{j>1} \frac{\alpha_j|^2}{|\alpha_1|^2} = \tan\theta$ and we obtain our desired result. ■

Applying the lemma to $p(\lambda) = \lambda^t$ and $\lambda_1 = (1+\epsilon)\lambda_2$, we get

$$\tan \angle(p(A)b, u_1) \leqslant (1+\epsilon)^{-t} \tan \angle(u_1, b)\,.$$

If there is a big gap between λ_1 and λ_2 this converges quickly but it can be slow if $\lambda_1 \approx \lambda_2$. It worth noting that if we choose $b \in \mathbb{R}^n$ to be a random direction, then

$$\mathbb{E}\left[\tan \angle(u_1, b)\right] = O\left(\sqrt{n}\right)\,.$$

Going one step further we can also see that the expression $p(A)b = A^t b$ can of course be built iteratively by repeatedly multiplying by A. For reasons of numerical stability it makes sense to normalize after each matrix-vector multiplication. This preserved the direction of the iterate and therefore does not change our convergence analysis. The resulting algorithms is the well known power method, defined recursively as follows:

$$x_0 = \frac{b}{\|b\|}$$
$$x_t = \frac{Ax_{t-1}}{\|Ax_{t-1}\|}$$

This method goes back more than hundred years to a paper by Müntz in 1913, but continues to find new applications today.

8.3 Applying Chebyshev polynomials

As we would expect from the development for quadratics, we can use Chebyshev polynomials to get a better solution the polynomial approximation problem that we posed above. The idea is exactly the same with the small difference that we normalize our Chebyshev polynomial slightly differently. This time around, we want to ensure that $p(\lambda_1) = 1$ so that we are picking out the first eigenvalue with the correct scaling.

Lemma 8.4. *A suitably rescaled degree t Chebyshev polynomial achieves*

$$\min_{p(\lambda_1)=1} \max_{\lambda\in[\lambda_2,\lambda_n]} p(\lambda) \leqslant \frac{2}{(1+\max\{\sqrt{\epsilon},\epsilon\})^t}$$

where $\epsilon = \frac{\lambda_1}{\lambda_2} - 1$ *quantifies the gap between the first and second eigenvalue.*

Note that the bound is much better than the previous one when ϵ is small. In the case of quadratics, the relevant "ϵ-value" was the inverse condition number. For eigenvalues, this turns into the gap between the first and second eigenvalue.

	$Ax = b$	$Ax = \lambda x$
ϵ	$\frac{1}{\kappa} = \frac{\alpha}{\beta}$	$\frac{\lambda_1}{\lambda_2} - 1$

As we saw before, Chebyshev polynomials satisfy a recurrence relation that can be used to derive an iterative method achieving the bound above. The main shortcoming of this method is that it needs information about the location of the first and second eigenvalue. Instead of describing this algorithm, we move on to an algorithm that works without any such information.

8.4 Conjugate gradient method

At this point, we switch back to linear equations $Ax = b$ for a symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$. The method we'll see is called *conjugate gradient* and is an important algorithm for solving linear equations. Its eigenvalue analog is the Lanczos method. While the ideas behind these methods are similar, the case of linear equations is a bit more intuitive.

Definition 8.5 (Conjugate gradient method)**.** We want to solve $Ax = b$, with $A \succ 0$ symmetric. The conjugate gradient method maintains a sequence of three points:

$$\begin{aligned} x_0 &= 0 && (\text{“candidate solution”}) \\ r_0 &= b && (\text{“residual”}) \\ p_0 &= r_0 && (\text{“search direction”}) \end{aligned}$$

For $t \geqslant 1$:

$$\begin{aligned} \eta_t &= \frac{\|r_t\|}{\langle p_{t-1}, Ap_{t-1} \rangle} && (\text{“step size”}) \\ x_t &= x_{t-1} + \eta_t p_{t-1} \\ r_t &= r_{t-1} - \eta_t A r_{t-1} \\ p_t &= r_t + \frac{\|r_t\|^2}{\|r_{t-1}\|^2} p_{t-1} \end{aligned}$$

Lemma 8.6. *The following three equations must always be true for the conjugate gradient method algorithm:*

- $\text{span}(\{r_0, ...r_{t-1}\}) = K_t(A, b)$
- *For $j < t$ we have $\langle r_t, r_j \rangle = 0$ and in particular $r_t \perp K_t(A, b)$.*
- *The search directions are conjugate $p_i^\top A p_j = 0$ for $i \neq j$.*

Proof. Proof by induction (see Trefethen and Bau). Show that the conditions are true initially and stay true when the update rule is applied. ■

Lemma 8.7. *Let $\|u\|_A = \sqrt{u^\top Au}$ and $\langle u, v \rangle_A = u^\top Av$ and $e_t = x^* - x_t$. Then e_t minimizes $\|x^* - x\|_A$ over all vectors $x \in K_{t-1}$.*

Proof. We know that $x_t \in K_t$. Let $x \in K_t$ and define $x = x_t - \delta$. Then, $e = x^* - x = e_t + \delta$. We compute the error in the A norm:

$$\begin{aligned}\|x^* - x\|_A^2 &= (e_t + \delta)^\top A(e_t + \delta) \\ &= e_t^\top A e_t + \delta^\top A\delta + 2e_t^\top A\delta\end{aligned}$$

Bby definition $e_t^\top A = r_t$. Note that $\delta \in K_t$. By Lemma 8.6, we have that $r_t \perp K_t(A, b)$. Therefore, $2e_t^\top A\delta = 0$ and hence,

$$\|e\|_A^2 = \|x^* - x\|_A^2 = e_t^\top A e_t + \delta^\top A\delta \geqslant \|e_t\|_A .$$

In the last step we used that $A \succ 0$. ■

What the lemma shows, in essence, is that conjugate gradient solves the polynomial approximation problem:

$$\min_{p\colon \deg(p)\leqslant t, p(0)=1} \|p(A)e_0\|_A .$$

Moreover, it's not hard to show that

$$\min_{p\colon \deg(p)\leqslant t, p(0)=1} \frac{\|p(A)e_0\|_A}{\|e_0\|_A} \leqslant \min_{p\colon \deg(p)\leqslant t, p(0)=1} \max_{\lambda\in\Lambda(A)} |p(\lambda)| .$$

In other words, the error achieved by conjugate gradient is no worse that the error of the polynomial approximation on the RHS, which was solved by the Chebyshev approximation. From here it follows that conjugate gradient must converge at least as fast in $\|\cdot\|_A$-norm than Chebyshev iteration.

9 Lower bounds and trade-offs

Coming soon

10 Lecture 10: Stochastic optimization

The goal in stochastic optimization is to minimize functions of the form

$$f(x) = \mathop{\mathbb{E}}_{z\sim\mathcal{D}} g(x, z)$$

which have stochastic component given by a distribution $\mathcal{D}$. In the case where the distribution has finite support, the function can be written as

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x) .$$

To solve these kinds of problems, we examine the stochastic gradient descent method and some if its many applications.

10.1 The stochastic gradient method

Following Robbins-Monro [RM51], we define the stochastic gradient method as follows.

Definition 10.1 (Stochastic gradient method)**.** The stochastic gradient method starts from a point $x_0 \in \Omega$ and proceeds according to the update rule

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t)$$

where $i_t \in \{1, \ldots, m\}$ is either selected at random at each step, or cycled through a random permutation of $\{1, \ldots, m\}$.

Either of the two methods for selecting i_t above, lead to the fact that

$$\mathbb{E}\, \nabla f_{i_t}(x) = \nabla f(x)$$

This is also true when $f(x) = \mathbb{E}\, g(x, z)$ and at each step we update according to $\nabla g(x, z)$ for randomly drawn $z \sim \mathcal{D}$.

10.1.1 Sanity check

Let us check that on a simple problem that the stochastic gradient descent yields the optimum. Let $p_1, \ldots, p_m \in \mathbb{R}^n$, and define $f \colon \mathbb{R}^n \to \mathbb{R}_+$:

$$\forall x \in \mathbb{R}^n,\ f(x) = \frac{1}{2m} \sum_{i=1}^{m} \|x - p_i\|_2^2$$

Note that here $f_i(x) = \frac{1}{2}\|x - p_i\|_2^2$ and $\nabla f_i(x) = x - p_i$. Moreover,

$$x^* = \arg\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{m} \sum_{i=1}^{m} p_i$$

Now, run SGM with $\eta_t = \frac{1}{t}$ in cyclic order i.e. $i_t = t$ and $x_0 = 0$:

$$\begin{aligned}
x_0 &= 0 \\
x_1 &= 0 - \frac{1}{1}(0 - p_1) = p_1 \\
x_2 &= p_1 - \frac{1}{2}(p_1 - p_2) = \frac{p_1 + p_2}{2} \\
&\vdots \\
x_n &= \frac{1}{m} \sum_{i=1}^{m} p_i = x^*
\end{aligned}$$

10.2 The Perceptron

The New York Times wrote in 1958 that the Perceptron [Ros58] was:

> *the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.*

So, let's see.

Definition 10.2 (Perceptron)**.** Given labeled points $((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathbb{R}^n \times \{-1, 1\})^m$, and and initial point $w_0 \in \mathbb{R}^n$, the Perceptron is the following algorithm. For $i_t \in \{1, \ldots, m\}$ selected uniformly at random,

$$w_{t+1} = w_t(1 - \gamma) + \eta \begin{cases} y_{i_t} x_{i_t} & \text{if } y_{i_t} \langle w_t, x_{i_t} \rangle < 1 \\ 0 & \text{otherwise} \end{cases}$$

Reverse-engineering the algorithm, the Perceptron is equivalent to running the SGM on the Support Vector Machine (SVM) objective function.

Definition 10.3 (SVM)**.** Given labeled points $((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathbb{R}^n \times \{-1, 1\})^m$, the SVM objective function is:

$$f(w) = \frac{1}{n} \sum_{i=1}^{m} \max(1 - y_i \langle w, x_i \rangle,\ 0) + \lambda \|w\|_2^2$$

The loss function $\max(1 - z,\ 0)$ is known as the Hinge Loss. The extra term $\lambda \|w\|_2^2$ is known as the regularization term.

10.3 Empirical risk minimization

We have two spaces of objects $\mathcal{X}$ and $\mathcal{Y}$, where we think of $\mathcal{X}$ as the space of *instances* or *examples*, and $\mathcal{Y}$ is a the set of *labels* or *classes*.

Our goal is to *learn* a function $h \colon \mathcal{X} \to \mathcal{Y}$ which outputs an object $y \in \mathcal{Y}$, given $x \in \mathcal{X}$. Assume there is a joint distribution $\mathcal{D}$ over the space $\mathcal{X} \times \mathcal{Y}$ and the training set consists of m instances $(x_1, y_1), \ldots, (x_m, y_m)$ drawn i.i.d. from $\mathcal{D}$.

We also define a non-negative real-valued loss function $\ell(y', y)$ to measure the difference between the prediction y' and the true outcome y.

Definition 10.4. The risk associated with h(x) is defined as the expectation of the loss function:

$$R[h] = \mathbb{E}_{\mathcal{X} \times \mathcal{Y} \in \mathcal{D}} \ell(h(x), y)$$

The ultimate goal of a learning algorithm is to find h^* among a class of functions $\mathcal{H}$ that minimizes $R[h]$:

$$h^* = \arg\min_{h \in \mathcal{H}} R[h]$$

In general, the risk $R[h]$ can not be computed because the joint distribution is unknown. Therefore,

The *empirical risk* is defined by averaging the loss function of the training set:

$$R_n[h] = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

An empirical risk minimizer is any point $h^* \in \arg\min_{h \in \mathcal{H}} R_n[h]$.

The stochastic gradient method can be thought of as minimizing the risk directly, if each example is only used once. In cases where we make multiple passes over the training set, it is better to think of it as minimizing the empirical risk, which can give different solutions than minimizing the risk. We'll develop tools to relate risk and empirical risk in the next lecture.

10.4 Online Learning

An interesting variant of this learning setup is called *online learning*. It arises when we do not have a set of training data, but rather must make decisions one-by-one.

Taking advice from experts. Imagine we have access to the predictions of n experts. We start from an initial distribution over experts, given by weights $w_1 \in \Delta_n = \{w \in \mathbb{R}^n : \sum_i w_i = 1, w_i \geqslant 0\}$.

At each step $t = 1, \ldots, T$:

- we randomly choose an expert according to w_t
- nature deals us a loss function $f_t \in [-1, 1]^n$, specifying for each expert i the loss $f_t[i]$ incurred by the prediction of expert i at time t.
- we incur the expected loss $\mathbb{E}_{i \sim w_t} f_t[i] = \langle w_t, f_t \rangle$.
- we get to update our distribution to from w_t to w_{t+1}.

At the end of the day, we measure how well we performed relative to the best fixed distribution over experts in hindsight. This is called *regret*:

$$R = \sum_{t=1}^{T} \langle w_t, f_t \rangle - \min_{w \in \Delta_n} \sum_{t=1}^{T} \langle w, f_t \rangle$$

This is a relative benchmark. Small regret does not say that the loss is necessarily small. It only says that had we played the same strategy at all steps, we couldn't have done much better even with the benefit of hindsight.

10.5 Multiplicative weights update

Perhaps the most important online learning algorithm is the *multiplicative weights update*. Starting from the uniform distribution w_1, proceed according to the following simple update rule for $t > 1$,

$$v_t[i] = w_{t-1}[i] e^{-\eta f_t[i]} \qquad \text{(exponential weights update)}$$
$$w_t = v_t / (\textstyle\sum_i v_t[i]) \qquad \text{(normalize)}$$

The question is *how do we bound the regret* of the multiplicative weights update? We could do a direct analysis, but instead we'll relate multiplicative weights to gradient descent and use the convergence guarantees we already know.

10.6 Multiplicative weights as mirror descent

Recall that mirror descent requires a mirror map $\phi : \Omega \to R$ over a domain $\Omega \in \mathbb{R}^n$ where ϕ is strongly convex and continuously differentiable.

The associated projection is

$$\Pi_\Omega^\phi(y) = \arg\min_{x \in \Omega} \mathcal{D}_\phi(x, y)$$

where $\mathcal{D}_\phi(x, y)$ is Bregman divergence.

Definition 10.5. The Bregman divergence measures how good the first order approximation of the function ϕ is:

$$\mathcal{D}_\phi(x, y) = \phi(x) - \phi(y) - \nabla\phi(y)^\intercal(x - y)$$

The mirror descent update rule is:

$$\nabla\phi(y_{t+1}) = \nabla\phi(x_t) - \eta g_t$$
$$x_{t+1} = \Pi_\Omega^\phi(y_{t+1})$$

where $g_t \in \partial f(x_t)$. In the first homework, we proved the following results.

Theorem 10.6. *let $\|\cdot\|$ be arbitrary norm and suppose that ϕ is α-strongly convex **w.r.t.** $\|\cdot\|$ on Ω. Suppose that f_t is L-lipschitz **w.r.t.** $\|\cdot\|$, we have:*

$$\frac{1}{T}\sum_{t=1}^{T} f_t(x_t) \leqslant \frac{\mathcal{D}_\phi(x^*, x_0)}{T\eta} + \eta\frac{L^2}{2\alpha}$$

Multiplicative weights are an instance of the Mirror Descent where $\Phi(w) = \sum_{i=1}^m w_i \log(w_i)$ is the negative entropy function. We have that

$$\nabla\Phi(w) = 1 + \log(w),$$

where the logarithm is elementwise. The update rule in Mirror Descent becomes:

$$\nabla\Phi(v_{t+1}) = \nabla\Phi(w_t) - \eta_t f_t$$
$$\implies v_{t+1} = w_t e^{-\eta_t f_t}$$

Now comes the projection step. The Bregman divergence is, for all $(x, y) \in \Omega^2$:

$$D_\Phi(x, y) = \Phi(x) - \Phi(y) - \nabla\Phi(y)^T(x - y)\,.$$

If we write out this expression and simplify, we recover the well-known *relative entropy*. The projection

$$\Pi_\Omega^\Phi(y) = \arg\min_{x\in\Omega} D_\Phi(x, y)$$

turns out to just correspond to the normalization step in the update rule.

Concrete rate of convergence. To get a concrete rate of convergence from the preceeding theorem, we still need to determine what value of the strong convexity constant α we get in our setting. Here, we choose the norm to be the ℓ_∞-norm. It follows from Pinsker's inequality that Φ is $1/2$-strongly convex with respect to the ℓ_∞-norm. Moreover, in the ℓ_∞-norm all gradients are bounded by 1, since the loss ranges in $[1, 1]$. Finally, the relative entropy between the initial uniform distribution any any other distribution is at most $\log(n)$. Putting these facts together and balancing the step size η, we get the normalized regret bound

$$O\left(\sqrt{\frac{\log n}{T}}\right).$$

In particular, this shows that the normalized regret of the multiplicative update rule is vanishing over time.

11 Lecture 14: Algorithms using Duality

The Lagrangian duality theory from the previous lecture can be used to design improved optimization algorithms which perform the optimization on the dual function. Oftentimes, passing to the dual can simplify computation or enable parallelization.

11.1 Review

Recall the *primal problem*

$$\begin{aligned} &\min_{x\in\Omega} f(x) \\ &\text{s.t. } Ax = b \end{aligned}$$

The corresponding dual problem is obtained by considering the *Lagrangian*

$$L(x,\lambda) = f(x) + \lambda^T(Ax - b)$$

where λ_i are called *Lagrange multipliers*. The *dual function* is defined as

$$g(\lambda) = \inf_{x\in\Omega} L(x,\lambda)$$

and the *dual problem* is

$$\sup_{\lambda\in\mathbb{R}^m} g(\lambda)$$

Definition 11.1 (Concave functions)**.** A function f is concave $\iff$ $-f$ is convex.

Fact 11.2. *The dual function is always concave (even if f and Ω are not convex).*

Proof. For any $x \in \Omega$, $L(x,\lambda)$ is a linear function of λ so $g(\lambda)$ is an infimum over a family of linear functions, hence concave. ■

11.2 Dual gradient ascent

Concavity of the dual function $g(\lambda)$ ensures existence of subgradients, so the subgradient method can be applied to optimize $g(\lambda)$. The *dual gradient ascent* algorithm is as follows:

Start from initial λ_0. For all $t \geqslant 0$:

$$x_t = \arg\inf_{x\in\Omega} L(x,\lambda_t)$$

$$\lambda_{t+1} = \lambda_t + \eta(Ax_t - b)$$

This yields the $O(1/\sqrt{t})$ convergence rate obtained by the subgradient method.

11.3 Augmented Lagrangian method / method of multipliers

Whereas dual gradient ascent updates λ_{t+1} by taking a step in a (sub)gradient direction, a method known as the *dual proximal method* can be motivated by starting with using the proximal operator [PB14] as an update rule for iteratively optimizing λ:

$$\lambda_{t+1} = \text{prox}_{\eta_t g}(\lambda_t) = \arg\sup_{\lambda} \underbrace{\underbrace{\inf_{x\in\Omega} f(x) + \lambda^T(Ax-b)}_{g(\lambda)} - \underbrace{\frac{1}{2\eta_t}\|\lambda-\lambda_t\|^2}_{\text{proximal term}}}_{h(\lambda)}$$

Notice that this expression includes a proximal term which makes $h(\lambda)$ strongly convex.

However, this update rule is not always directly useful since it requires optimizing $h(\lambda)$ over λ, which may not be available in closed form. Instead, notice that if we can interchange inf and sup (e.g. strong duality, Sion's theorem applied when Ω is compact) then we can rewrite

$$\begin{aligned}\sup_{\lambda}\inf_{x\in\Omega} f(x) + \lambda^T(Ax-b) - \frac{1}{2\eta_t}\|\lambda-\lambda_t\|^2 &= \inf_{x\in\Omega}\sup_{\lambda} f(x) + \lambda^T(Ax-b) - \frac{1}{2\eta_t}\|\lambda_t-\lambda\|^2 \\ &= \inf_{x\in\Omega} f(x) + \lambda_t^T(Ax-b) + \frac{\eta_t}{2}\|Ax-b\|^2\end{aligned}$$

where the inner sup is optimized in closed-form by $\lambda = \lambda_t + \eta_t(Ax-b)$. To isolate the remaining optimization over x, we make the following definition.

Definition 11.3 (Augmented Lagrangian)**.** The *augmented Lagrangian* is

$$L_\eta(x,\lambda) = f(x) + \lambda_t^T(Ax-b) + \frac{\eta_t}{2}\|Ax-b\|^2$$

The *augmented Lagrangian method* (aka Method of Multipliers) is defined by the following iterations:

$$x_t = \arg\inf_{x\in\Omega} L_{\eta_t}(x,\lambda_t)$$

$$\lambda_{t+1} = \lambda_t + \eta_t(Ax_t - b)$$

While the iterations look similar to dual gradient ascent, there are some noteworthy differences

- The method of multipliers can speed up convergence (e.g. for non-smooth functions), but computing x_t may be more difficult due to the additional $\frac{\eta_t}{2}\|Ax-b\|^2$ term in the augmented Lagrangian
- $L(x,\lambda_t)$ is convex in x, but $L_\eta(x,\lambda_t)$ is strongly convex in λ (if A is full-rank)
- Convergence at a $O(1/t)$ rate. More precisely, for constant step size η, we can show show the method of multipliers satisfies

$$g(\lambda_t) - g^* \geqslant -\frac{\|\lambda^*\|^2}{2\eta t}$$

11.4 Dual decomposition

A major advantage of dual decomposition that it can lead to update rules which are trivially parallelizable.

Suppose we can partition the primal problem into blocks of size $(n_i)_{i=1}^N$, i.e.

$$x^T = ((x^{(1)})^T, \cdots, (x^{(N)})^T) \qquad x_i \in \mathbb{R}^{n_i}, \sum_{i=1}^N n_i = n$$

$$A = [A_1|\cdots|A_N] \qquad Ax = \sum_{i=1}^N A_i x^{(i)}$$

$$f(x) = \sum_{i=1}^N f_i(x^{(i)})$$

Then the Lagrangian is also separable in x

$$L(x, \lambda) = \sum_{i=1}^N \left(f_i(x^{(i)}) + \lambda^T A_i x^{(i)} - \frac{1}{N} \lambda^T b \right) = \sum_{i=1}^N L_i(x^{(i)}, \lambda)$$

Each term in the sum consists of one non-interacting partition $(x^{(i)}, A_i, f_i)$, so minimization of each term in the sum can occur in parallel. This leads to the *dual decomposition algorithm*:

- In parallel on worker nodes: $x_t^{(i)} = \arg\inf_{x^{(i)}} L_i(x^{(i)}, \lambda_t)$
- On a master node: $\lambda_{t+1} = \lambda_t + \eta(Ax - b)$

Example 11.4 (Consensus optimization)**.** Consensus optimization is an application that comes up in distributed computing which can be solved using dual decomposition. Given a graph $G = (V, E)$,

$$\min_x \sum_{v \in V} f_v(x_v) : x_v = x_u \; \forall (u, v) \in E$$

This problem is separable over $v \in V$, so dual decomposition applies.

Example 11.5 (Network utility maximization)**.** Suppose we have a network with k links, each with capacity c_i. We are interested in allocating N different flows with fixed routes over these links such that utility is maximized and resource constraints are not exceeded. Let $x_i \in \mathbb{R}^N$ represent the amount of flow i allocated and $U_i : \mathbb{R} \to \mathbb{R}$ a convex utility function which returns the amount of utility obtained from having x_i amount of flow i. The optimization problem is

$$\max_x \sum_{i=1}^N U_i(x_i) : Rx \leqslant c$$

where R is a $k \times N$ matrix whose (k, i)th entry gives the amount of the capacity of link k is consumed by flow i.

To rewrite the primal problem in standard form (i.e. as a minimization), take negatives:

$$\min_x - \sum_i U_i(x^{(i)}) : Rx \leqslant c$$

The dual problem is then

$$\max_{\lambda \geqslant 0} \min_{x} \sum_{i} -U_i(x^{(i)}) + \lambda^T (Rx - c)$$

where the $Rx \leqslant c$ primal inequality constraint results in the $\lambda \geqslant 0$ constraint. The second term can be rewritten as $\lambda^T \left(\sum_i R_i x_i - \frac{1}{N} c \right)$, showing that the dual splits over i and hence dual decomposition applies. This leads to a parallel algorithm which each worker node computes

$$\arg\max_{x_i} U_i(x_i) - \lambda^T R_i x_i$$

and the master node computes

$$\lambda_{t+1} = [\lambda_t + \eta(Rx - c)]_+$$

We take the positive part because of the $\lambda \geqslant 0$ constraint.

Aside: In resource allocation problems, the values of the dual variables λ at the optimal point have an economic interpretation as "prices" to the resources. In this example, λ_k should be interpreted as the price per unit of flow over link k.

11.5 ADMM — Alternating direction method of multipliers

While dual decomposition can be used to parallelize dual subgradient ascent, it doesn't work with the augmented Lagrangian. This is because the coupling between the decision variables introduced by the $\|Ax - b\|^2$ term prevents the augmented Lagrangian from being separable over x.

The goal of the alternating direction method of multipliers (ADMM) is to obtain the best of both worlds: we would like both the parallelism offered by the method of multipliers as well as the faster convergence rate of the augmented Lagrangian. We will see that similar to dual decomposition, ADMM partitions the decision variables into two blocks. Also, similar to the method of multiplers, ADMM uses the augmented Lagrangian $L_\eta(x, z, \lambda_t)$.

Consider a problem of the form

$$\min_{x,z} f(x) + g(z) : Ax + Bz \leqslant c$$

In other words, we can split the objective and constraints into two blocks x and z.

The method of multipliers would jointly optimize the augmented Lagrangian on both blocks in one single optimization step:

$$\begin{aligned}(x_{t+1}, z_{t+1}) &= \inf_{x,z} L_\eta(x, z, \lambda_t) \\ \lambda_{t+1} &= \lambda_t + \eta(Ax_{t+1} + Bz_{t+1} - c)\end{aligned}$$

In contrast, ADMM alternates (the "A" in "ADMM") between optimizing the augmented Lagrangian over x and z:

$$\begin{aligned}x_{t+1} &= \inf_{x} L_\eta(x, z_t, \lambda_t) \\ z_{t+1} &= \inf_{z} L_\eta(x_{t+1}, z, \lambda_t) \\ \lambda_{t+1} &= \lambda_t + \eta(Ax_{t+1} + Bz_{t+1} - c)\end{aligned}$$

Unlike the method of multipliers, this is not parallelizable since x_{t+1} must be computed before z_{t+1}. Also, convergence guarantees are weaker: rather than getting a convergence rate we only get an asymptotic convergence guarantee.

Theorem 11.6. *Assume*

- *f, g have a closed, non-empty, convex epigraph*
- *L_0 has a saddle x^*, z^*, λ^*, i.e.:*

$$\forall x, z, \lambda : L_0(x^*, z^*, \lambda) \leqslant L_0(x^*, z^*, \lambda^*) \leqslant L(x, z, \lambda^*)$$

Then, as $t \to \infty$, ADMM satisfies

$$\begin{aligned} f(x_t) + g(z_t) &\to p^* \\ \lambda_t &\to \lambda^* \end{aligned}$$

Aside: Saddles are useful because inf and the sup can be swapped. To see this, note the saddle condition

$$L(x^*, \lambda) \leqslant L(x^*, \lambda^*) \leqslant L(x, \lambda^*)$$

implies that

$$\begin{aligned} \inf_x \sup_\lambda L(x, \lambda) &\leqslant \sup_\lambda L(x^*, \lambda) \\ &\leqslant L(x^*, \lambda^*) \\ &= \inf_x L(x, \lambda^*) \\ &\leqslant \sup_\lambda \inf_x L(x, \lambda) \end{aligned}$$

12 List of contributors

Many thanks to the students of EE227C for their generous help in creating these lecture notes.

Lecture 2: Michael Cheng, Neil Thomas, Morris Yau

Lecture 3:

Lecture 5: Victoria Cheng, Kun Qian, Zeshi Zheng

Lecture 6: Adam Gleave, Andy Deng, Mathilde Badoual

Lecture 7: Aurelien Bibaut, Zhi Chen, Michael Zhang

Lecture 8: Eugene Vinitsky

Lecture 14: Feynman Liang

13 Acknowledgments

These notes build on an earlier course by Ben Recht, as well as an upcoming textbook by Recht and Wright. Some chapters also closely follow Bubeck's monograph on the topic [Bub15].

References

[AZO17] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *Proc. 8th ITCS*, 2017.

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[DSSSC08] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. 25th ICML*, pages 272–279. ACM, 2008.

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[Lax07] Peter D. Lax. *Linear Algebra and Its Applications*. Wiley, 2007.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Doklady AN SSSR (translated as Soviet Mathematics Doklady)*, 269:543–547, 1983.

[Nes04] Yurii Nesterov. *Introductory Lectures on Convex Programming. Volume I: A basic course*. Kluwer Academic Publishers, 2004.

[PB14] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[RM51] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[TD97] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. SIAM, 1997.