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

April 27, 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

Part I
Gradient methods

Taylor's approximation tells us that

$$f(x+\delta) \approx f(x) + \delta f'(x) + \frac{1}{2}\delta^2 f''(x)\,.$$

This approximation directly reveals that if we move from x to $x+\delta$ where $\delta = -\eta \cdot f'(x)$ for sufficiently small $\eta > 0$, we generally expect to decrease the function value by about $\eta f'(x)^2$.

We will generalize this simple idea to functions of many variables with the help of multivariate versions of Taylor's theorem. The simple greedy way of decreasing the function value is known as *gradient descent*. Gradient descent converges to points at which the first derivatives vanish. For the broad class of *convex* functions such points turn out to be globally minimal.

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.

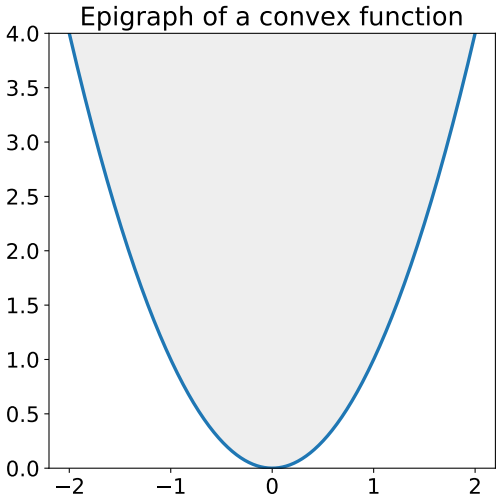
1.2 Convex functions

Definition 1.3 (Convex function)**.** A function $f : \Omega \to \mathbb{R}$ 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 $f : \Omega \to \mathbb{R}$ 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 $f : \Omega \to \mathbb{R}$ 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 $f : \Omega \to \mathbb{R}$ 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 $f : \Omega \to \mathbb{R}$ is differentiable and let $x, y \in \Omega$. Then,*

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

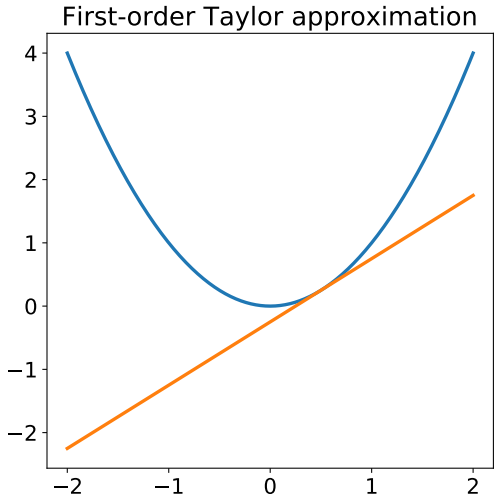


Figure 1: Taylor approximation of $f(x) = x^2$ at 0.5.

Taylor's theorem implies the following statement.

Proposition 1.7. *Assume $f : \Omega \to \mathbb{R}$ 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 $f : \Omega \to \mathbb{R}$ 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,

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

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 non-negative. Hence, the first-order 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 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 $f : \Omega \to \mathbb{R}$. 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_1 is defined by the iterative update rule

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)\,, \qquad t = 1, 2, \ldots$$

where the scalar η_t is the so-called *step size*, sometimes called *learning rate*, that may vary with t. There are numerous ways of choosing step sizes that have a significant effect on the performance of gradient descent. What we will see in this lecture are several choices of step sizes that ensure the convergence of gradient descent by virtue of a theorem. These step sizes are not necessarily ideal for practical applications.

2.1.1 Projections

In cases where the constraint set Ω is not all of $\mathbb{R}^n$, the gradient update can take us outside the domain Ω. How can we ensure that $x_{t+1} \in \Omega$? One natural approach is to

"project" each iterate back onto the domain Ω. As it turns out, this won't really make our analysis more difficult and so we include from the get-go.

Definition 2.1 (Projection)**.** The *projection* of a point x onto a set Ω is defined as

$$\Pi_\Omega(x) = \arg\min_{y \in \Omega} \|x - y\|_2 .$$

Example 2.2. 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. In fact, a stronger claim is true that follows from the Pythagorean theorem.

Lemma 2.3.

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

So, now we can modify our original procedure as displayed in Figure 2.

Starting from $x_1 \in \Omega$, repeat:	
$y_{t+1} = x_t - \eta \nabla f(x_t)$	(gradient step)
$x_{t+1} = \Pi_\Omega(y_{t+1})$	(projection)

Figure 2: Projected gradient descent

And we are guaranteed that $x_{t+1} \in \Omega$. Note that computing the projection may be computationally the hardest part of the problem. However, there are convex sets for which we know explicitly how to compute the projection (see Example 2.2). We will see several other non-trivial examples in later lectures.

2.2 Lipschitz functions

The first assumption that leads to a convergence analysis is that the gradients of the objective function aren't too big over the domain. This turns out to follow from a natural Lipschitz continuity assumption.

Definition 2.4 (L-Lipschitz)**.** A function $f : \Omega \to \mathbb{R}$ is *L-Lipschitz* if for every $x, y \in \Omega$, we have

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

Fact 2.5. *If the function f is L-Lipschitz, differentiable, and convex, then*

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

We can now prove our first convergence rate for gradient descent.

Theorem 2.6. *Assume that function f is convex, differentiable, and L-Lipschitz 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 an optimal point $x^* \in \arg\min_{x\in\Omega} f(x)$. Let $x_1, \ldots, x_t$ be the sequence of iterates computed by t steps of projected gradient descent with constant step size $\eta = \frac{R}{L\sqrt{t}}$. 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:

$$u^\top v = \frac{1}{2}(\|u\|^2 + \|v\|^2 - \|u - v\|^2) \tag{2}$$

This property follows from the more familiar identity $\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2u^\top v$.

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

$$\begin{aligned}
f(x_s) - f(x^*) &\leqslant \nabla f(x_s)^\top (x_s - x^*) && \text{(by convexity)}\\
&= \frac{1}{\eta}(x_s - y_{s+1})^\top (x_s - x^*) && \text{(by the update rule)}\\
&= \frac{1}{2\eta}\left(\|x_s - x^*\|^2 + \|x_s - y_{s+1}\|^2 - \|y_{s+1} - x^*\|^2\right) && \text{(by Equation 2)}\\
&= \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2\right) + \frac{\eta}{2}\|\nabla f(x_s)\|^2 \\
& && \text{(by the update rule)}\\
&\leqslant \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2\right) + \frac{\eta L^2}{2} && \text{(Lipschitz condition)}\\
&\leqslant \frac{1}{2\eta}\left(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2\right) + \frac{\eta L^2}{2} && \text{(Lemma 2.3)}
\end{aligned}$$

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

$$\begin{aligned}
\sum_{s=1}^{t} f(x_s) - f(x^*) &\leqslant \frac{1}{2\eta} \sum_{s=1}^{t} \left(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2 \right) + \frac{\eta L^2 t}{2} \\
&= \frac{1}{2\eta} \left(\|x_1 - x^*\|^2 - \|x_t - x^*\|^2 \right) + \frac{\eta L^2 t}{2} && \text{(telescoping sum)} \\
&\leqslant \frac{1}{2\eta} \|x_1 - x^*\|^2 + \frac{\eta L^2 t}{2} && \text{(since } \|x_t - x^*\| \geqslant 0\text{)} \\
&\leqslant \frac{R^2}{2\eta} + \frac{\eta L^2 t}{2} && \text{(since } \|x_1 - x^*\| \leqslant R\text{)}
\end{aligned}$$

Finally,

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

■

2.3 Smooth functions

The next property we'll encounter is called *smoothness*. The main point about smoothness is that it allows us to control the second-order term in the Taylor approximation. This often leads to stronger convergence guarantees at the expense of a relatively strong assumption.

Definition 2.7 (Smoothness)**.** A continuously differentiable function f is β smooth if the gradient gradient map $\nabla f \colon \mathbb{R}^n \to \mathbb{R}^n$ is β-Lipschitz, i.e,

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

We will need a couple of technical lemmas before we can analyze gradient descent for smooth functions. It's safe to skip the proof of these technical lemmas on a first read.

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

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

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

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

■

We also need the following lemma.

Lemma 2.9. *Let f be a β-smooth convex function, then for every $x, y \in \mathbb{R}^n$, we have*

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

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

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

Here, the inequality follows from convexity and smoothness. ■

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 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, we get

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

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

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

■

3 Strong convexity

This lecture introduces the notion of strong convexity and combines it with smoothness to develop the concept of condition number. While smoothness gave as an upper bound on the second-order term in Taylor's approximation, strong convexity will give us a lower bound. Taking together, these two assumptions are quite powerful as they lead to a much faster convergence rate of the form $\exp(-\Omega(t))$. In words, gradient descent on smooth and strongly convex functions decreases the error multiplicatively by some factor strictly less than 1 in each iteration.

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 (Strong convexity)**.** A function $f\colon \Omega \to \mathbb{R}$ 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.

A twice differentiable function is α-strongly convex if $\nabla^2 f(x) \succeq \alpha I$.

Once again, note that the parameter α changes under affine transformations. Conveniently enough, for α-strongly convex, β-smooth functions, we can define a basis-independent quantity called the *condition number*.

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 the matrix defining the quadratic.

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	$\epsilon \leqslant O(1/t)$	$\epsilon \leqslant e^{-\Omega(t)}$

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.

3.3 Convergence rate strongly convex functions

For no good reason we begin with a convergence bound for strongly convex Lipschitz functions, in which we obtain a $O(1/t)$ rate of convergence.

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

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

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

$$\begin{aligned} f(x_s) - f(x^*) &\leqslant \nabla f(x_s)^\top (x_s - x^*) - \frac{\alpha}{2}\|x_s - x^*\|^2 \\ &= \frac{1}{\eta_s}(x_s - y_{s+1})^\top (x_s - x^*) - \frac{\alpha}{2}\|x_s - x^*\|^2 && \text{(by update rule)} \\ &= \frac{1}{2\eta_s}(\|x_s - x^*\|^2 + \|x_s - y_{s+1}\|^2 - \|y_{s+1} - x^*\|^2) - \frac{\alpha}{2}\|x_s - x^*\|^2 \\ & && \text{(by "Fundamental Theorem of Optimization")} \\ &= \frac{1}{2\eta_s}(\|x_s - x^*\|^2 - \|y_{s+1} - x^*\|^2) + \frac{\eta_s}{2}\|\nabla f(x_s)\|^2 - \frac{\alpha}{2}\|x_s - x^*\|^2 \\ & && \text{(by update rule)} \\ &\leqslant \frac{1}{2\eta_s}(\|x_s - x^*\|^2 - \|x_{s+1} - x^*\|^2) + \frac{\eta_s}{2}\|\nabla f(x_s)\|^2 - \frac{\alpha}{2}\|x_s - x^*\|^2 \\ & && \text{(by Lemma 2.3)} \\ &\leqslant (\frac{1}{2\eta_s} - \frac{\alpha}{2})\|x_s - x^*\|^2 - \frac{1}{2\eta_s}\|x_{s+1} - x^*\|^2 + \frac{\eta_s L^2}{2} && \text{(by Lipschitzness)} \end{aligned}$$

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. ■

3.4 Convergence rate for smooth and strongly convex functions

Theorem 3.5. *Assume $f\colon \mathbb{R}^n \to \mathbb{R}$ 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 $f\colon \Omega \to \mathbb{R}$ 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 $g\colon \Omega \to \mathbb{R}$ 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 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 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 as summarized in Figure 3.

Starting from $x_0 \in \Omega$, repeat:

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

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

Figure 3: Conditional gradient

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 $f : \Omega \to \mathbb{R}$ 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,

$$f(x_{t+1}) - f(x^*) \leqslant (1 - \eta_t)(f(x_t) - f(x^*)) + \frac{\eta_t^2 \beta D^2}{2} \tag{3}$$

We use induction in order to prove $f(x_t) - f(x^*) \leqslant \frac{2\beta D^2}{t+2}$ based on Equation 3 above.

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 $t = 0$, we have $\eta_t = \frac{2}{0+2} = 1$. Hence,

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

Thus, the induction hypothesis holds for our base case.

Inductive step. 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 3,

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

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. Calculating the gradient of this function, we 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

$$\tilde{X}_t \in \operatorname*{argmin}_{X \in B_*} \nabla f(X_t)^\top X$$

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*

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

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

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

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 described in Figure 4.

- 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.

Figure 4: Power method

Part II
Accelerated gradient methods

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 $f : \mathbb{R}^n \to \mathbb{R}$ takes the form:

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

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:

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

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 $f\colon \mathbb{R}^n \to \mathbb{R}$ 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 simple solution has a uniform step size $\eta_t = \frac{2}{\alpha+\beta}$. Note that

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

recovering the same convergence rate we proved previously. The resulting polynomial $p_t(x)$ is plotted in Figure 5 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:

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

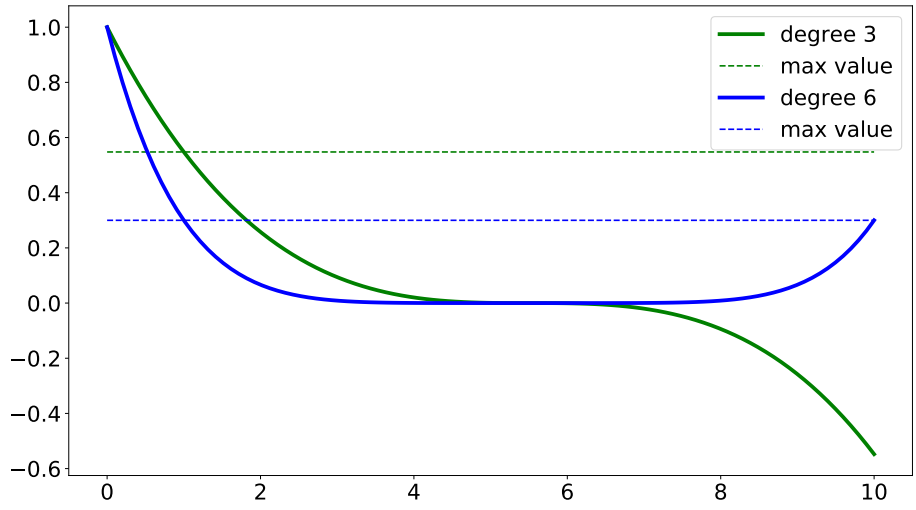


Figure 5: Naive Polynomial

Figure 6 plots the first few Chebyshev polynomials.

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

Let's compare on figure 8 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.

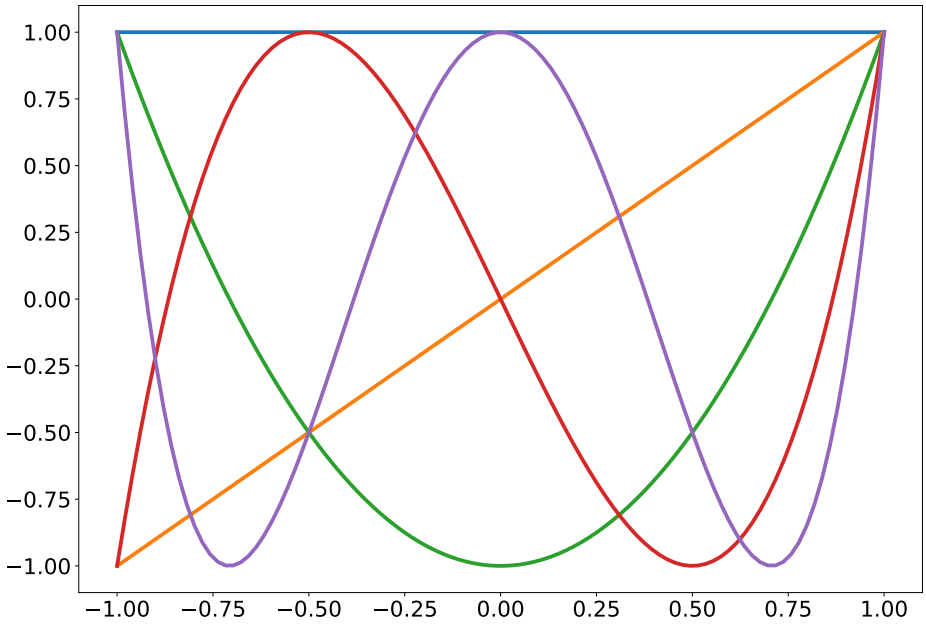


Figure 6: Chebychev polynomials of varying degrees.

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 $|T_k(1+\epsilon)|$.
Fact: for $a > 1$, $T_k(a) = \cosh\left(k \cdot \operatorname{arccosh}(a)\right)$ where:

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

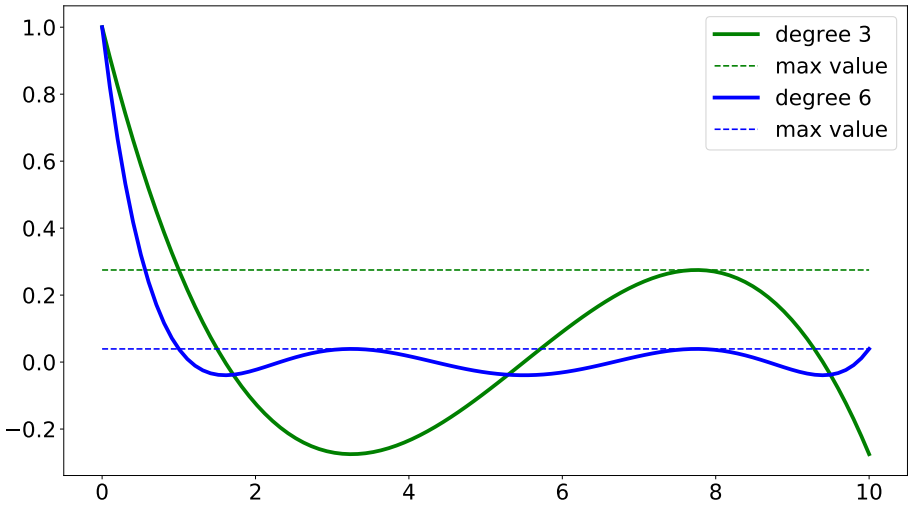


Figure 7: Rescaled Chebyshev

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 $|T_k(1+\epsilon)|$:

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

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:

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

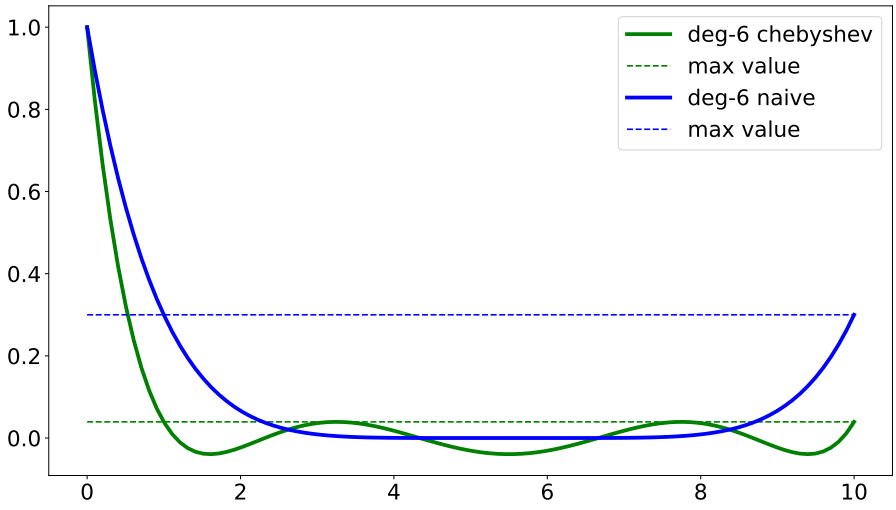


Figure 8: Rescaled Chebyshev VS Naive Polynomial

This means that for large κ, we get quadratic savings in the degree we need before the error drops off exponentially. Figure 9 shows the different rates of convergence, we clearly see that the

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.

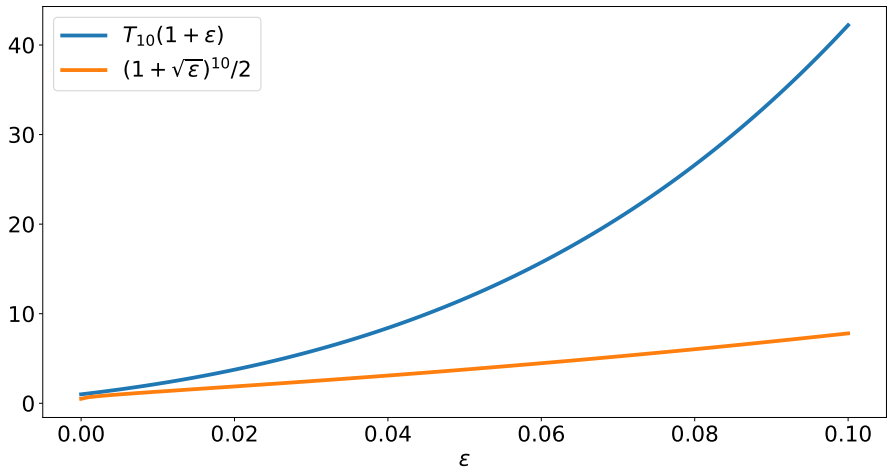


Figure 9: Convergence for naive polynomial and Chebyshev

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 restarting, 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:

$$f(y_t) - f(x_t) \leqslant -\frac{1}{2\beta}\|\nabla f(x_t)\|^2 \tag{4}$$

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

$$\eta\langle \nabla f(x_{t+1}), z_t - u\rangle = \frac{\eta^2}{2}\|\nabla f(x_{t+1})\|^2 + \frac{1}{2}\|z_t - u\|^2 - \frac{1}{2}\|z_{t+1} - u\|^2 . \tag{5}$$

Substituting the first equation yields

$$\eta\langle \nabla f(x_{t+1}, z_t - u\rangle \leqslant \eta^2\beta(f(x_{t+1}) - f(y_{t+1})) + \frac{1}{2}\|z_t - u\|^2 - \frac{1}{2}\|z_{t+1} - u\|^2 \tag{6}$$

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

$$\begin{aligned}
&\eta\langle \nabla f(x_{t+1}), x_{t+1} - u\rangle - \eta\langle \nabla f(x_{t+1}), z_t - u\rangle \\
&= \eta\langle \nabla f(x_{t+1}), x_{t+1} - z_t\rangle \\
&= \frac{1-\tau}{\tau}\eta\langle \nabla f(x_{t+1}), y_t - x_{t+1}\rangle \\
&\leqslant \frac{1-\tau}{\tau}\eta(f(y_t) - f(x_{t+1})) \quad \text{(by convexity).}
\end{aligned} \tag{7}$$

Combining (6) and (7), 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 (4) that is commonly used in the analysis of gradient descent for smooth functions. The other is Equation 5 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 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 *Krylov 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 $u_1 \ldots u_n$ and ordered eigenvalues $\lambda_1 \geqslant \lambda_2 \ldots \geqslant \lambda_n$. 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

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

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

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

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 make 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

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

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 $t \geqslant 1$:

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

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* $\|u\|_A = \sqrt{u^\top A u}$ *and* $\langle u, v \rangle_A = u^\top A v$ *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:

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

By 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, $2 e_t^\top A \delta = 0$ 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:

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

Moreover, it's not hard to show that

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

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 with robustness

In the first part of this lecture, we study whether the convergence rates derived in previous lectures are tight. For several classes of optimization problems (smooth, strongly convex, etc), we prove the answer is indeed yes. The highlight of this analysis is to show the $O(1/t^2)$ rate achieved by Nesterov's accelerated gradient method is optimal (in a weak technical sense) for smooth, convex functions.

In the second part of this lecture, we go beyond studying convergence rates and look towards other ways of comparing algorithms. We show the improved rates of accelerated gradient methods come at a cost in robustness to noise. In particular, if we restrict ourselves to only using approximate gradients, the standard gradient method suffers basically no slowdown, whereas the accelerated gradient method accumulates errors linearly in the number of iterations.

Table 3: Upper Bounds from Lectures 2-8

Function class	Algorithm	Rate
Convex, Lipschitz	Gradient descent	$RL/\sqrt{t}$
Strongly convex, Lipschitz	Gradient descent	$L^2/(\alpha t)$
Convex, smooth	Accelerated gradient descent	$\beta R^2/t^2$

9.1 Lower bounds

Before launching into a discussion of lower bounds, it's helpful to first recap the upper bounds obtained thus far. For a convex function f, Table (3) summarizes the assumptions and rates proved in the first several lectures.

Each of the rates in Table (3) is obtained using some variant of the gradient method. These algorithms can be thought of as a procedure that maps a history of points and subgradients $(x_1, g_1, \ldots, x_t, g_t)$ to a new point x_{t+1}. To prove lower bounds, we restrict the class of algorithms to similar procedures. Formally, define a black-box procedure as follows.

Definition 9.1 (Black-Box Procedure)**.** A *black-box procedure* generates a sequence of points $\{x_t\}$ such that

$$x_{t+1} \in x_0 + \text{span}\{g_1, \ldots, g_t\},$$

and $g_s \in \partial f(x_s)$.

Throughout, we will further assume $x_0 = 0$. As expected, gradient descent is a black-box procedure. Indeed, unrolling the iterates, x_{t+1} is given by

$$\begin{aligned} x_{t+1} &= x_t - \eta \nabla f(x_t) \\ &= x_{t-1} - \eta \nabla f(x_{t-2}) - \eta \nabla f(x_{t-1}) \\ &= x_0 - \sum_{i=0}^{t} \eta \nabla f(x_i). \end{aligned}$$

We now turn to proving lower bounds on the convergence rate for any black-box procedure. Our first theorem concerns the constrained, non-smooth case. The theorem is originally from [**?**], but the presentation will follow [**?**].

Theorem 9.2 (Constrainted, Non-Smooth f)**.** *Let $t \leqslant n$, $L, R > 0$. There exists a convex L-Lipschitz function f such that any black-box procedure satisfies*

$$\min_{1 \leqslant s \leqslant t} f(x_s) - \min_{x \in B_2(R)} f(x) \geqslant \frac{RL}{2(1+\sqrt{t})}. \tag{8}$$

Furthermore, there is an α-strongly convex, L-Lipschitz function f such that

$$\min_{1 \leqslant s \leqslant t} f(x_s) - \min_{x \in B_2(\frac{L}{2\alpha})} f(x) \geqslant \frac{L^2}{8\alpha t}. \tag{9}$$

The proof strategy is to exhibit a convex function f so that, for any black-box procedure, $\operatorname{span}\{g_1, g_2, \ldots, g_i\} \subset \operatorname{span}\{e_1, \ldots, e_i\}$, where e_i is the i-th standard basis vector. After t steps for $t < n$, at least $n - t$ coordinates are exactly 0, and the theorem follows from lower bounding the error for each coordinate that is identically zero.

Proof. Consider the function

$$f(x) = \gamma \max_{1 \leqslant i \leqslant t} x[i] + \frac{\alpha}{2}\|x\|^2,$$

for some $\gamma, \alpha \in \mathbb{R}$. In the strongly convex case, γ is a free parameter, and in the Lipschitz case both α and γ are free parameters. By the subdifferential calculus,

$$\partial f(x) = \alpha x + \gamma \operatorname{conv}\{e_i : i \in \operatorname*{argmax}_{1 \leqslant j \leqslant t} x(j)\}.$$

The function f is evidently α-strongly convex. Furthermore, if $\|x\| \leqslant R$ and $g \in \partial f(x)$, then $\|g\| \leqslant \alpha R + \gamma$, so f is $(\alpha R + \gamma)$-Lipschitz on $B_2(R)$.

Suppose the gradient oracle returns $g_i = \alpha x + \gamma e_i$, where i is the first coordinate such that $x[i] = \max_{1 \leqslant j \leqslant t} x[j]$. An inductive argument then shows

$$x_s \in \operatorname{span}\{e_1, \ldots, e_{s-1}\}$$

Consequently, for $s \leqslant t$, $f(x_s) \geqslant 0$. However, consider $y \in \mathbb{R}^n$ such that

$$y[i] = \begin{cases} -\frac{\gamma}{\alpha t} & \text{if } 1 \leqslant i \leqslant t \\ 0 & \text{otherwise.} \end{cases}$$

Since $0 \in \partial f(y)$, y is an minimizer of f with objective value

$$f(y) = \frac{-\gamma^2}{\alpha t} + \frac{\alpha}{2}\frac{\gamma^2}{\alpha^2 t} = -\frac{\gamma^2}{2\alpha t},$$

and hence $f(x_s) - f(y) \geqslant \frac{\gamma^2}{2\alpha t}$. We conclude the proof by appropriately choosing α and γ. In the convex, Lipschitz case, set

$$\alpha = \frac{L}{R}\frac{1}{1+\sqrt{t}} \quad \text{and} \quad \gamma = L\frac{\sqrt{t}}{1+\sqrt{t}}.$$

Then, f is L-Lipschitz,

$$\|y\| = \sqrt{t\left(\frac{-\gamma}{\alpha t}\right)^2} = \frac{\gamma}{\alpha\sqrt{t}} = R$$

and hence

$$f(x_s) - \min_{x \in B_2(R)} f(x) = f(x_s) - f(y) \geqslant \frac{\gamma^2}{2\alpha t} = \frac{RL}{2(1+\sqrt{t})}.$$

In the strongly-convex case, set $\gamma = \frac{L}{2}$ and take $R = \frac{L}{2\alpha}$. Then, f is L-Lipschitz,

$$\|y\| = \frac{\gamma}{\alpha\sqrt{t}} = \frac{L}{2\alpha\sqrt{t}} = \frac{R}{\sqrt{t}} \leqslant R,$$

and therefore

$$f(x_s) - \min_{x \in B_2(L/2\alpha)} f(x) = f(x_s) - f(y) \geqslant \frac{LR}{4t} = \frac{L^2}{8\alpha t}.$$

■

Next, we study the smooth, convex case and show the $O(1/t^2)$ rate achieved by accelerated gradient descent is optimal.

Theorem 9.3 (Smooth-f)**.** *Let $t \leqslant \frac{n-1}{2}$, $\beta > 0$. There exists a β-smooth convex quadratic f such that any black-box method satisfies*

$$\min_{1 \leqslant s \leqslant t} f(x_s) - f(x^\star) \geqslant \frac{3\beta\|x_0 - x^\star\|_2^2}{32(t+1)^2}. \tag{10}$$

Similar to the previous theorem, the proof strategy is to exhibit a pathological convex function. In this case, we choose what Nesterov calls "the worst-function in the world" [**?**].

Proof. Without loss of generality, let $n = 2t + 1$. Let $L \in \mathbb{R}^{n \times n}$ be the tridiagonal matrix

$$L = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots -1 & 2 \end{bmatrix}.$$

The matrix L is almost the Laplacian of the cycle graph (in fact, it's the Laplacian of the chain graph).[1] Notice

$$x^\top L x = x[1]^2 + x[n]^2 + \sum_{i=1}^{n-1} (x[i] - x[i+1])^2,$$

and, from this expression, it's a simple to check $0 \preceq L \preceq 4I$. Define the following β-smooth convex function

$$f(x) = \frac{\beta}{8} x^\top L x - \frac{\beta}{4}\langle x, e_1 \rangle.$$

[1] https://en.wikipedia.org/wiki/Laplacian_matrix

The optimal solution $x^\star$ satisfies $Lx^\star = e_1$, and solving this system of equations gives

$$x^\star[i] = 1 - \frac{i}{n+1},$$

which has objective value

$$\begin{aligned} f(x^\star) &= \frac{\beta}{8} x^{\star\top} L x^\star - \frac{\beta}{4}\langle x^\star, e_1\rangle \\ &= -\frac{\beta}{8}\langle x^\star, e_1\rangle = -\frac{\beta}{8}(1 - \frac{1}{n+1}). \end{aligned}$$

Similar to the proof of (9.2), we can argue

$$x_s \in \operatorname{span}\{e_1, \ldots, e_{s-1}\},$$

so if $x_0 = 0$, then $x_s[i] = 0$ for $i \geqslant s$ for any black-box procedure. Let $x_s^\star = \operatorname{argmin}_{x:\, i \geqslant s, x[i]=0} f(x)$. Notice $x_s^\star$ is the solution of a smaller $s \times s$ Laplacian system formed by the first s rows and columns of L, so

$$x_s^\star[i] = \begin{cases} 1 - \frac{i}{s+1} & \text{if } i < s \\ 0 & \text{otherwise,} \end{cases}$$

which has objective value $f(x_s^\star) = -\frac{\beta}{8}(1 - \frac{1}{s+1})$. Therefore, for any $s \leqslant t$,

$$\begin{aligned} f(x_s) - f(x^\star) &\geqslant f(x_t^\star) - f(x^\star) \\ &\geqslant \frac{\beta}{8}\left(\frac{1}{t+1} - \frac{1}{n+1}\right) \\ &= \frac{\beta}{8}\left(\frac{1}{t+1} - \frac{1}{2(t+1)}\right) \\ &= \frac{\beta}{8}\frac{1}{2(t+1)}. \end{aligned}$$

To conclude, we bound the initial distance to the optimum. Recalling $x_0 = 0$,

$$\begin{aligned} \|x_0 - x^\star\|^2 = \|x^\star\|^2 &= \sum_{i=1}^n (1 - \frac{i}{n+1})^2 \\ &= n - \frac{2}{n+1}\sum_{i=1}^n i + \frac{1}{(n+1)^2}\sum_{i=1}^n i^2 \\ &\leqslant n - \frac{2}{n+1}\sum_{i=1}^n i + \frac{1}{(n+1)^2}\int_1^{n+1} x^2\, dx \\ &\leqslant n - \frac{2}{n+1}\frac{n(n+1)}{2} + \frac{1}{(n+1)^2}\frac{(n+1)^3}{3} \\ &= \frac{(n+1)}{3} \\ &= \frac{2(t+1)}{3}. \end{aligned}$$

Combining the previous two displays, for any $s \leqslant t$,

$$f(x_s) - f(x^\star) \geqslant \frac{\beta}{8}\frac{1}{2(t+1)} \geqslant \frac{3\beta\|x_0 - x^\star\|^2}{32(t+1)^2}.$$

■

9.2 Robustness and acceleration trade-offs

The first part of the course focused almost exclusively on convergence rates for optimization algorithms. From this perspective, a better algorithm is one with a faster rate of convergence. A theory of optimization algorithms that stops with rates of convergence is incomplete. There are often other important algorithm design goals, e.g. robustness to noise or numerical errors, that are ignored by focusing on converges rates, and when these goals are of primary importance, excessive focus on rates can lead practitioners to choose the wrong algorithm. This section deals with one such case.

In the narrow, technical sense of the previous section, Nesterov's Accelerated Gradient Descent is an "optimal" algorithm, equipped with matching upper and lower bounds on it's rate of convergence. A slavish focus on convergence rates suggests one should then always use Nesterov's method. Before coronating Nesterov's method, however, it is instructive to consider how it performs in the presence of noise.

Figure (10) shows compares the performance of vanilla gradient descent and Nesterov's accelerated gradient descent on the function f used in the proof of Theorem (**??**). In the noiseless case, the accelerated method obtains the expected speed-up over gradient descent. However, if we add a small amount of spherical noise to the gradients, the speed-up not only disappears, but gradient descent begins to outperform the accelerated method, which begins to diverge after a large number of iterations.

The preceding example is not wickedly pathological in any sense. Instead, it is illustrative of a much broader phenomenon. Work by Devolder, Glineur and Nesterov (DGN) [DGN14] shows there is a fundamental trade-off between acceleration and robustness, in a sense made precise below.

First, define the notion of an inexact gradient oracle. Recall for a β-smooth convex function f and any $x, y \in \Omega$,

$$0 \leqslant f(x) - (f(y) + \langle \nabla f(y), x - y\rangle) \leqslant \frac{\beta}{2}\|x - y\|^2. \tag{11}$$

For any $y \in \Omega$, an exact first-order oracle then returns a pair $(f(y), g(y)) = (f(y), \nabla f(y))$ that satisfies (11) exactly for every $x \in \Omega$. An inexact oracle, returns a pair so that (11) holds up to some slack δ.

Definition 9.4 (Inexact-Oracle)**.** Let $\delta > 0$. For any $y \in \Omega$, a δ-inexact oracle returns a pair $(f_\delta(y), g_\delta(y))$ such that

$$0 \leqslant f(x) - (f(y) + \langle \nabla f(y), x - y\rangle) \leqslant \frac{\beta}{2}\|x - y\|^2 + \delta$$

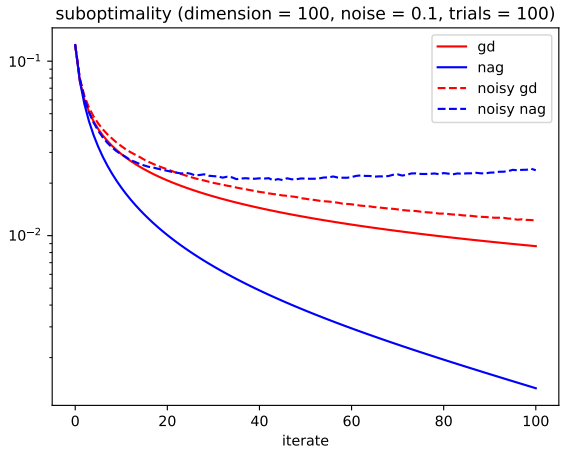


Figure 10: The optimality gap for iterations of gradient descent and Nesterov accelerated gradient descent applied to the worst function in the world with dimension $n = 100$. Notice with exact oracle gradients, acceleration helps significantly. However, when adding uniform spherical random noise with radius $\delta = 0.1$ to the gradient, stochastic gradient descent remains robust while stochastic accelerated gradient accumulates error. The stochastic results are averaged over 100 trials.

for every $x \in \Omega$.

Consider running gradient descent with a δ-inexact oracle. DGN [DGN14] show, after t steps,

$$f(x_t) - f(x^\star) \leqslant \frac{\beta R^2}{2t} + \delta.$$

Comparing this rate with Table (3), the plain gradient method is not affected by the inexact oracle and doesn't accumulate errors. On the other hand, if the accelerated gradient method is run with a δ-inexact oracle, then after t steps,

$$f(x_t) - f(x^\star) \leqslant \frac{4\beta R^2}{(t+1)^2} + \frac{1}{3}(t+3)\delta.$$

In other words, the accelerated gradient method accumulates errors linearly with the number of steps! Moreover, this slack is not an artifact of the analysis. Any black-box method must accumulate errors if it is accelerated in the exact case, as the following theorem makes precise.

Theorem 9.5 ([DGN14] Theorem 6)**.** *Consider a black-box method with convergence rate* $O\left(\frac{\beta R^2}{t^p}\right)$ *when using an exact oracle. With a* δ*-inexact oracle, suppose the algorithm achieves a rate*

$$f(x_t) - f(x^\star) \leqslant O\left(\frac{\beta R^2}{t^p}\right) + O\left(t^q \delta\right), \tag{12}$$

then $q \geqslant p - 1$.

In particular, for any accelerated method has $p > 1$, and consequently $q > 1$ so the method accumulates at least $O(t^{p-1}\delta)$ error with the number of iterations.

Part III
Stochastic optimization

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,

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

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 $S = ((x_1, y_1), \ldots, (x_m, y_m))$ 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* of a function $h \colon \mathcal{X} \to \mathcal{Y}$ is defined as

$$R[h] = \mathbb{E}_{(x,y) \sim \mathcal{D}} \ell(h(x), y)\,.$$

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

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

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

Therefore, we instead minimize a proxy objective known as *empirical risk* and defined by averaging the loss function over the training set:

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

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

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.

Specifically, we will interpret multiplicative weights as an instance of 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

$$\Pi_\Omega^\phi(y) = \operatorname*{argmin}_{x \in \Omega} \mathcal{D}_\phi(x, y)$$

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:

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

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:

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

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

$$\Pi_\Omega^\Phi(y) = \operatorname*{argmin}_{x\in\Omega} D_\Phi(x, y)$$

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 preceding 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 Learning, stability, regularization

In this lecture we take a look at machine learning, and empirical risk minimization in particular. We define the distribution of our data as D over $X \times Y$, where $X \subseteq \mathbb{R}^d$ and Y is some discrete set of class labels. For instance, in a binary classification tasks with two labels Y might be $Y = \{-1, 1\}$.

- A "model" is specified by a set of parameters $w \in \Omega \subseteq \mathbb{R}^n$
- The "loss function" is denoted by $\ell \colon \Omega \times (X \times Y) \to \mathbb{R}$, note that $\ell(w, z)$ gives the loss of model w on instance z.
- The risk of a model is defined as $R(w) = \mathbb{E}_{z \sim D}[\ell(w, z)]$.

Our goal is to find a model w that minimizes $R(w)$.

One way to accomplish this is to use stochastic optimization directly on the population objective:

$$w_{t+1} = w_t - \eta \nabla \ell(w_t, z_t) \quad z \sim D$$

When given a finite data set, however, it is usually effective to make multiple passes over the data. In this case, the stochastic gradient method may no longer optimize risk directly.

11.1 Empirical risk and generalization error

Consider a finite sample Suppose $S = ((x_1, y_1), ..., (x_m, y_m)) \in (X \times Y)^m$, where $z_i = (x_i, y_i)$ represents the i-th labeled example. The empirical risk is defined as

$$R_S(w) = \frac{1}{m} \sum_{i=1}^{m} \ell(w, z_i).$$

Empirical risk minimization is commonly used as a proxy for minimizing the unknown population risk. But how good is this proxy? Ideally, we would like that the point w that we find via empirical risk minimization has $R_S(w) \approx R(w)$. However, this may not be the case, since the risk $R(w)$ captures loss on unseen example, while the empirical

risk $R_S(w)$ captures loss on seen examples. Generally, we expect to do much better on seen examples than unseen examples. This performance gap between seen and unseen examples is what we call *generalization error*.

Definition 11.1 (Generalization error)**.** We define the *generalization error* of a model w as

$$\epsilon_{\text{gen}}(w) = R(w) - R_s(w)\,.$$

Note the following tautological, yet important identity:

$$R(w) = R_S(w) + \epsilon_{\text{gen}}(w) \tag{13}$$

What this shows in particular is that if we manage to make the empirical risk $R_S(w)$ small through optimization, then all that remains to worry about is generalization error.

So, how can we bound generalization error? The fundamental relationship we'll establish in this lecture is that generalization error equals an algorithmic robustness property that we call *algorithmic stability*. Intuitively, algorithmic stability measures how sensitive an algorithm is to changes in a single training example.

11.2 Algorithmic stability

To introduce the idea of stability, we choose two independent samples $S = (z_1, ..., z_m)$ and $S' = (z'_1, ..., z'_m)$, each drawn independently and identically from D. Here, the second sample S' is called a *ghost sample* and mainly serves an analytical purpose.

Correlating the two samples in a single point, we introduce the hybrid sample $S^{(i)}$ as:

$$S^{(i)} = (z_1, \ldots, z_{i-1}, z'_i, z_{i+1}, \ldots, z_m)$$

Note that here the i-th example comes from S', while all others come from S.

With this notation at hand, we can introduce a notion of average stability.

Definition 11.2 (Average stability)**.** The *average stability* of an algorithm $A : (X \times Y)^m \to \Omega$:

$$\Delta(A) = \mathop{\mathbb{E}}_{S,S'}\left[\frac{1}{m}\sum_{i=1}^{m}\Big(\ell(A(S), z'_i) - \ell(A(S^{(i)}), z'_i)\Big)\right]$$

This definition can be interpreted as comparing the performance of the algorithm on an unseen versus a seen example. This is the intuition why average stability, in fact, equals generalization error.

Theorem 11.3.

$$\mathbb{E}[\epsilon_{gen}(A)] = \Delta(A)$$

Proof. Note that

$$\mathbb{E}[\epsilon_{\text{gen}}(A)] = \mathbb{E}\left[R(A(S)) - R_S(A(S))\right],$$
$$\mathbb{E}\left[R_S(A(S))\right]] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\ell(A(S), z_i)\right],$$
$$\mathbb{E}[R(A(S))] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\ell(A(S), z_i')\right].$$

At the same time, since z_i and z_i' are identically distributed and independent of the other examples, we have

$$\mathbb{E}\,\ell(A(S), z_i) = \mathbb{E}\,\ell(A(S^{(i)}), z_i').$$

Applying this identity to each term in the empirical risk above, and comparing with the definition of $\Delta(A)$, we conclude $\mathbb{E}[R(A(S)) - R_S(A(S))] = \Delta(A)$ ■

11.2.1 Uniform stability

While average stability gave us an exact characterization of generalization error, it can be hard to work with the expectation over S and S'. Uniform stability replaces the averages by suprema, leading to a stronger but useful notion [BE02].

Definition 11.4 (Uniform stability)**.** The uniform stability of an algorithm A is defined as

$$\Delta_{\sup}(A) = \sup_{\mathcal{S},\mathcal{S}'\in(X\times Y)^m}\sup_{i\in[m]}|\ell(A(S), z_i') - \ell(A(S^{(i)}, z_i')|$$

Since uniform stability upper bounds average stability, we know that uniform stability upper bounds generalization error (in expectation).

Corollary 11.5. $\mathbb{E}[\epsilon_{gen}(A)] \leqslant \Delta_{\sup}(A)$

This corollary turns out to be surprisingly useful since many algorithms are uniformly stable. For instance, strongly convex loss function is sufficient for stability, and hence generalization as we will show next.

11.3 Stability of empirical risk minimization

The next theorem due to [SSSSS10] shows that empirical risk minimization of a strongly convex loss function is uniformly stable.

Theorem 11.6. *Assume $\ell(w, z)$ is α-strongly convex over the domain Ω and L-Lipschitz. Let $\widehat{w}_S = \arg\min_{w\in\Omega}\frac{1}{m}\sum_{i=1}^{m}\ell(w, z_i)$ denote the empirical risk minimizer (ERM). Then, ERM satisfies*

$$\Delta_{\sup}(ERM) \leqslant \frac{4L^2}{\alpha m}.$$

An interesting point is that there is no explicit reference to the complexity of the class. In the following we present the proof.

Proof. Denote by $\widehat{w}_S$ the empirical risk minimizer on a sample S. Fix arbitrary samples S, S' of size m and an index $i \in [m]$. We need to show that

$$|(\ell(\widehat{w}_{S^{(i)}}, z_i') - \ell(\widehat{w}_S, z_i'))| \leqslant \frac{4L^2}{\alpha m}.$$

On one hand, by strong convexity we know that

$$R_S(\widehat{w}_{S^{(i)}}) - R_S(\widehat{w}_S) \geqslant \frac{\alpha}{2}\|\widehat{w}_S - \widehat{w}_{S^{(i)}}\|^2.$$

On the other hand,

$$\begin{aligned}
&R_S(\widehat{w}_{S^{(i)}}) - R_S(\widehat{w}_S) \\
&= \frac{1}{m}(\ell(\widehat{w}_{S^{(i)}}, z_i) - \ell(\widehat{w}_S, z_i)) + \frac{1}{m}\sum_{i \neq j}(\ell(\widehat{w}_{S^{(i)}}, z_j) - \ell(\widehat{w}_S, z_j)) \\
&= \frac{1}{m}(\ell(\widehat{w}_{S^{(i)}}, z_i) - \ell(\widehat{w}_S, z_i)) + \frac{1}{m}(\ell(\widehat{w}_S, z_i') - \ell(\widehat{w}_{S^{(i)}}, z_i')) + \left(R_{S^{(i)}}(\widehat{w}_{S^{(i)}}) - R_{S^{(i)}}(\widehat{w}_S)\right) \\
&\leqslant \frac{1}{m}|\ell(\widehat{w}_{S^{(i)}}, z_i) - \ell(\widehat{w}_S, z_i)| + \frac{1}{m}|(\ell(\widehat{w}_S, z_i') - \ell(\widehat{w}_{S^{(i)}}, z_i'))| \\
&\leqslant \frac{2L}{m}\|\widehat{w}_{S^{(i)}} - \widehat{w}_S\|.
\end{aligned}$$

Here, we used that

$$R_{S^{(i)}}(\widehat{w}_{S^{(i)}}) - R_{S^{(i)}}(\widehat{w}_S)) \leqslant 0$$

and the fact that ℓ is $L-$lipschitz.

Putting it all together $\|\widehat{w}_{S^{(i)}} - \widehat{w}_S\| \leqslant \frac{4L}{\alpha m}$. Then by the Lipschitz condition we have that

$$\frac{1}{m}|(\ell(\widehat{w}_{S^{(i)}}, z_i') - \ell(\widehat{w}_S, z_i'))| \leqslant L\|\widehat{w}_{S^{(i)}} - \widehat{w}_S\| \leqslant \frac{4L^2}{\alpha m}.$$

Hence, $\Delta_{\text{sup}}(\text{ERM}) \leqslant \frac{4L^2}{\alpha m}$.

■

11.4 Regularization

Not all the ERM problems are strongly convex. However, if the problem is convex we can consider the regularized objective

$$r(w, z) = \ell(w, z) + \frac{\alpha}{2}\|w\|^2$$

The regularized loss $r(w, z)$ is $\alpha-$strongly convex. The last term is named ℓ_2-regularization, weight decay or Tikhonov regularization depending on the field you work on. Therefore, we now have the following chain of implications:

$$\text{regularization} \Rightarrow \text{strong convexity} \Rightarrow \text{uniform stability} \Rightarrow \text{generalization}$$

We can also show that solving the regularized objective also solves the unregularized objective. Assume that $\Omega \subseteq \mathcal{B}_2(R)$, by setting $\alpha \approx \frac{L^2}{R^2 m}$ we can show that the minimizer of the regularized risk also minimizes the unregularized risk up to error $\mathcal{O}(\frac{LR}{\sqrt{m}})$. Moreover, by the previous theorem the generalized error will also be $\mathcal{O}(\frac{LR}{\sqrt{m}})$. See Theorem 3 in [SSSSS10] for details.

11.5 Implicit regularization

In implicit regularization the algorithm itself regularizes the objective, instead of explicitly adding a regularization term. The following theorem describes the regularization effect of the Stochastic Gradient Method (SGM).

Theorem 11.7. *Assume $\ell(\cdot, z)$ is convex, β-smooth and L-Lipschitz. If we run SGM for T steps, then the algorithm has uniform stability*

$$\Delta_{\sup}(SGM_T) \leqslant \frac{2L^2}{m} \sum_{t=1}^{T} \eta_t$$

Note for $\eta_t \approx \frac{1}{m}$ then $\Delta_{\sup}(\mathrm{SGM}_T) = \mathcal{O}(\frac{\log(T)}{m})$, and for $\eta_t \approx \frac{1}{\sqrt{m}}$ and $T = \mathcal{O}(m)$ then $\Delta_{\sup}(\mathrm{SGM}_T) = \mathcal{O}(\frac{1}{\sqrt{m}})$. See [HRS15] for proof.

12 Coordinate descent

There are many classes of functions for which it is very cheap to compute directional derivatives along the standard basis vectors $e_i, i \in [n]$. For example,

$$f(x) = \|x\|^2 \quad \text{or} \quad f(x) = \|x\|_1 \tag{14}$$

This is especially true of common regularizers, which often take the form

$$R(x) = \sum_{i=1}^{n} R_i(x_i) \ . \tag{15}$$

More generally, many objectives and regularizes exhibit "group sparsity"; that is,

$$R(x) = \sum_{j=1}^{m} R_j(x_{S_j}) \tag{16}$$

where each $S_j, j \in [m]$ is a subset of $[n]$, and similarly for $f(x)$. Examples of functions with block decompositions and group sparsity include:

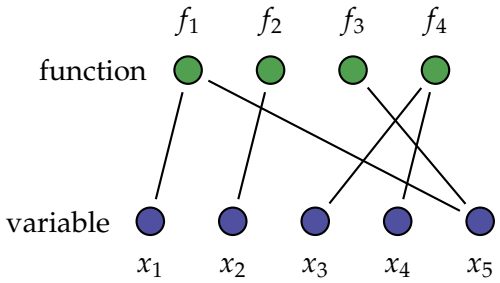


Figure 11: Example of the bipartite graph between component functions $f_i, i \in [m]$ and variables $x_j, j \in [n]$ induced by the group sparsity structure of a function $f : \mathbb{R}^n \to \mathbb{R}^m$. An edge between f_i and x_j conveys that the ith component function depends on the jth coordinate of the input.

1. Group sparsity penalties;
2. Regularizes of the form $R(U^\top x)$, where R is coordinate-separable, and U has sparse columns and so $(U^\top x) = u_i^\top x$ depends only on the nonzero entries of U_i;
3. Neural networks, where the gradients with respect to some weights can be computed "locally"; and
4. ERM problems of the form

$$f(x) := \sum_{i=1}^{n} \phi_i(\langle w^{(i)}, x \rangle) \tag{17}$$

where $\phi_i : \mathbb{R} \to \mathbb{R}$, and $w^{(i)}$ is zero except in a few coordinates.

12.1 Coordinate descent

Denote $\partial_i f = \frac{\partial f}{x_i}$. For each round $t = 1, 2, \ldots$, the coordinate descent algorithm chooses an index $i_t \in [n]$, and computes

$$x_{t+1} = x_t - \eta_t \partial_{i_t} f(x_t) \cdot e_{i_t} . \tag{18}$$

This algorithm is a special case of stochastic gradient descent. For

$$\begin{aligned} \mathbb{E}[x_{t+1}|x_t] &= x_t - \eta_t \, \mathbb{E}[\partial_{i_t} f(x_t) \cdot e_{i_t}] && (19) \\ &= x_t - \frac{\eta_t}{n} \sum_{i=1}^{n} \partial_i f(x_t) \cdot e_i && (20) \\ &= x_t - \eta_t \nabla f(x_t) . && (21) \end{aligned}$$

Recall the bound for SGD: If $\mathbb{E}[g_t] = \nabla f(x_t)$, then SGD with step size $\eta = \frac{1}{BR}$ satisfies

$$\mathbb{E}[f(\frac{1}{T}\sum_{t=1}^{T} x_t)] - \min_{x\in\Omega} f(x) \leqslant \frac{2BR}{\sqrt{T}} \tag{22}$$

where R^2 is given by $\max_{x\in\Omega} \|x - x_1\|_2^2$ and $B = \max_t \mathbb{E}[\|g_t\|_2^2]$. In particular, if we set $g_t = n\partial_{x_{i_t}} f(x_t) \cdot e_{i_t}$, we compute that

$$\mathbb{E}[\|g_t\|_2^2] = \frac{1}{n}\sum_{i=1}^{n} \|n \cdot \partial_{x_i} f(x_t) \cdot e_i\|_2^2 = n\|\nabla f(x_t)\|_2^2 \,. \tag{23}$$

If we assume that f is L-Lipschitz, we additionally have that $\mathbb{E}[\|g_t\|^2] \leqslant nL^2$. This implies the first result:

Proposition 12.1. *Let f be convex and L-Lipschitz on $\mathbb{R}^n$. Then coordinate descent with step size $\eta = \frac{1}{nR}$ has convergence rate*

$$\mathbb{E}[f(\frac{1}{T}\sum_{t=1}^{T} x_t)] - \min_{x\in\Omega} f(x) \leqslant 2LR\sqrt{n/T} \tag{24}$$

12.2 Importance sampling

In the above, we decided on using the uniform distribution to sample a coordinate. But suppose we have more fine-grained information. In particular, what if we knew that we could bound $\sup_{x\in\Omega} \|\nabla f(x)_i\|_2 \leqslant L_i$? An alternative might be to sample in a way to take L_i into account. This motivates the "importance sampled" estimator of $\nabla f(x)$, given by

$$g_t = \frac{1}{p_{i_t}} \cdot \partial_{i_t} f(x_t) \text{ where } i_t \sim \text{Cat}(p_1, \ldots, p_n) \,. \tag{25}$$

Note then that $\mathbb{E}[g_t] = \nabla f(x_t)$, but

$$\begin{aligned} \mathbb{E}[\|g_t\|_2^2] &= \sum_{i=1}^{n} (\partial_{i_t} f(x_t))^2 / p_i^2 && (26) \\ &\leqslant \sum_{i=1}^{n} L_i^2 / p_i^2 && (27) \end{aligned}$$

In this case, we can get rates

$$\mathbb{E}[f(\frac{1}{T}\sum_{t=1}^{T} x_t)] - \min_{x\in\Omega} f(x) \leqslant 2R\sqrt{1/T} \cdot \sqrt{\sum_{i=1}^{n} L_i^2 / p_i^2} \tag{28}$$

In many cases, if the values of L_i are heterogeneous, we can optimize the values of p_i.

12.3 Importance sampling for smooth coordinate descent

In this section, we consider coordinate descent with a *biased* estimator of the gradient. Suppose that we have, for $x \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, the inequality

$$|\partial_{x_i} f(x) - \partial_{x_i} f(x + \alpha e_i)| \leqslant \beta_i |\alpha| \tag{29}$$

where β_i are possibly heterogeneous. Note that if that f is twice-continuously differentiable, the above condition is equivalent to $\nabla^2_{ii} f(x) \leqslant \beta_i$, or $\operatorname{diag}(\nabla^2 f(x)) \preceq \operatorname{diag}(\boldsymbol{\beta})$. Define the distribution p^γ via

$$p_i^\gamma = \frac{\beta_i^\gamma}{\sum_{j=1}^n \beta_j^\gamma} \tag{30}$$

We consider gradient descent with the rule called RCD(γ)

$$x_{t+1} = x_t - \frac{1}{\beta_{i_t}} \cdot \partial_{i_t}(x_t) \cdot e_{i_t}, \text{ where } i_t \sim p^\gamma \tag{31}$$

Note that as $\gamma \to \infty$, coordinates with larger values of β_i will be selected more often. Also note that this is *not generally* equivalent to SGD, because

$$\mathbb{E}\left[\frac{1}{\beta_{i_t}} \partial_{i_t}(x_t) e_i\right] = \frac{1}{\sum_{j=1}^n \beta_j^\gamma} \cdot \sum_{i=1}^n \beta_i^{\gamma-1} \partial_i f(x_t) e_i = \frac{1}{\sum_{j=1}^n \beta_j^\gamma} \cdot \nabla f(x_t) \circ (\beta_i^{\gamma-1})_{i \in [n]} \tag{32}$$

which is only a scaled version of $\nabla f(x_t)$ when $\gamma = 1$. Still, we can prove the following theorem:

Theorem 12.2. *Define the weighted norms*

$$\|x\|^2_{[\gamma]} := \sum_{i=1}^n x_i^2 \beta_i^\gamma \text{ and } \|x\|^{*2}_{[\gamma]} := \sum_{i=1}^n x_i^2 \beta_i^{-\gamma} \tag{33}$$

and note that the norms are dual to one another. We then have that the rule RCD(γ) *produces iterates satisfying*

$$\mathbb{E}[f(x_t) - \arg\min_{x \in \mathbb{R}^n} f(x)] \leqslant \frac{2R^2_{1-\gamma} \cdot \sum_{i=1}^n \beta_i^\gamma}{t-1}, \tag{34}$$

where $R^2_{1-\gamma} = \sup_{x \in \mathbb{R}^n : f(x) \leqslant f(x_1)} \|x - x^*\|_{[1-\gamma]}$.

Proof. Recall the inequality that for a general β_g-smooth convex function g, one has that

$$g\left(u - \frac{1}{\beta_g} \nabla g(u)\right) - g(u) \leqslant -\frac{1}{2\beta_g} \|\nabla g\|^2 \tag{35}$$

Hence, considering the functions $g_i(u;x) = f(x + ue_i)$, we see that $\partial_i f(x) = g_i'(u;x)$, and thus g_i is β_i smooth. Hence, we have

$$f\left(x - \frac{1}{\beta_i}\nabla f(x)e_i\right) - f(x) = g_i(0 - \frac{1}{\beta_g}g_i'(0;x)) - g(0;x) \leqslant -\frac{g_i'(u;x)^2}{2\beta_i} = -\frac{\partial_i f(x)^2}{2\beta_i}\,. \quad (36)$$

Hence, if $i\ p^\gamma$, we have

$$\begin{aligned}
\mathbb{E}[f(x - \frac{1}{\beta_i}\partial_i f(x)e_i) - f(x)] &\leqslant \sum_{i=1}^n p_i^\gamma \cdot -\frac{\partial_i f(x)^2}{2\beta_i} && (37)\\
&= -\frac{1}{2\sum_{i=1}^n \beta_i^\gamma}\sum_{i=1}^n \beta^{\gamma-1}\partial_i f(x)^2 && (38)\\
&= -\frac{\|\nabla f(x)\|_{[1-\gamma]}^{*2})}{2\sum_{i=1}^n \beta_i^\gamma} && (39)
\end{aligned}$$

Hence, if we define $\delta_t = \mathbb{E}[f(x_t) - f(x^*)]$, we have that

$$\delta_{t+1} - \delta_t \leqslant -\frac{\|\nabla f(x_t)\|_{[1-\gamma]}^{*2}}{2\sum_{i=1}^n \beta_i^\gamma} \quad (40)$$

Moreover, with probability 1, one also has that $f(x_{t+1}) \leqslant f(x_t)$, by the above. We now continue with the regular proof of smooth gradient descent. Note that

$$\begin{aligned}
\delta_t &\leqslant \nabla f(x_t)^\top (x_t - x_*)\\
&\leqslant \|\nabla f(x_t)\|_{[1-\gamma]}^* \|x_t - x_*\|_{[1-\gamma]}\\
&\leqslant R_{1-\gamma}\|\nabla f(x_t)\|_{[1-\gamma]}^*\,.
\end{aligned}$$

Putting these things together implies that

$$\delta_{t+1} - \delta_t \leqslant -\frac{\delta_t^2}{2R_{1-\gamma}^2\sum_{i=1}^n \beta_i^\gamma} \quad (41)$$

Recall that this was the recursion we used to prove convergence in the non-stochastic case. ■

Theorem 12.3. *If f is in addition α-strongly convex w.r.t to $\|\cdot\|_{[1-\gamma]}$, then we get*

$$\mathbb{E}[f(x_{t+1}) - \arg\min_{x\in\mathbb{R}^n} f(x)] \leqslant \left(1 - \frac{\alpha}{\sum_{i=1}^n \beta_i^\gamma}\right)^t (f(x_1) - f(x^*))\,. \quad (42)$$

Proof. We need the following lemma:

Lemma 12.4. *Let f be an α-strongly convex function w.r.t to a norm $\|\cdot\|$. Then, $f(x) - f(x^*) \leqslant \frac{1}{2\alpha}\|\nabla f(x)\|_*^2$.*

Proof.

$$\begin{aligned} f(x) - f(y) &\leqslant \nabla f(x)^\top (x-y) - \frac{\alpha}{2}\|x-y\|_2^2 \\ &\leqslant \|\nabla f(x)\|_* \|x-y\|^2 - \frac{\alpha}{2}\|x-y\|_2^2 \\ &\leqslant \max_t \|\nabla f(x)\|_* t - \frac{\alpha}{2} t^2 \\ &= \frac{1}{2\alpha}\|\nabla f(x)\|_*^2 . \end{aligned}$$

■

Lemma 12.4 shows that

$$\|\nabla f(x_s)\|_{[1-\gamma]}^{*2} \geqslant 2\alpha\delta_s .$$

On the other hand, Theorem 12.2 showed that

$$\delta_{t+1} - \delta_t \leqslant -\frac{\|\nabla f(x_t)\|_{[1-\gamma]}^{*2}}{2\sum_{i=1}^n \beta_i^\gamma} \tag{43}$$

Combining these two, we get

$$\delta_{t+1} - \delta_t \leqslant -\frac{\alpha\delta_t}{\sum_{i=1}^n \beta_i^\gamma} \tag{44}$$

$$\delta_{t+1} \leqslant \delta_t \left(1 - \frac{\alpha}{\sum_{i=1}^n \beta_i^\gamma}\right) . \tag{45}$$

Applying the above inequality recursively and recalling that $\delta_t = \mathbb{E}[f(x_t) - f(x^*)]$ gives the result.

■

12.4 Random coordinate vs. stochastic gradient descent

What's surprising is that RCD(γ) is a descent method, despite being random. This is not true of normal SGD. But when does RCD(γ) actually do better? If $\gamma = 1$, the savings are proportional to the ratio of $\sum_{i=1} \beta_i/\beta \cdot (T_{coord}/T_{grad})$. When f is twice differentiable, this is the ratio of

$$\frac{\operatorname{tr}(\max_x \nabla^2 f(x))}{\|\max_x \nabla^2 f(x)\|_{\text{op}}} (T_{coord}/T_{grad}) \tag{46}$$

12.5 Other extensions to coordinate descent

1. Non-Stochastic, Cyclic SGD
2. Sampling with Replacement
3. Strongly Convex + Smooth!?
4. Strongly Convex (generalize SGD)
5. Acceleration? See [TVW+17]

Part IV
Dual methods

13 Duality theory

These notes are based on earlier lecture notes by Benjamin Recht and Ashia Wilson.

13.1 Optimality conditions for equality constrained optimization

Recall that $x_\star$ minimizes a smooth, convex function f over a closed convex set Ω if and only if

$$\langle \nabla f(x_\star), x - x_\star \rangle \geqslant 0 \quad \forall x \in \Omega\,. \tag{47}$$

Let's specialize this to the special case where Ω is an affine set. Let A be an $n \times d$ matrix with rank n such that $\Omega = \{x \ : \ Ax = b\}$ for some $b \in \mathbb{R}^n$. Note that we can always assume that $\operatorname{rank}(A) = n$ or else we would have redundant constraints. We could also parameterize Ω as $\Omega = \{x_0 + v : Av = 0\}$ for any $x_0 \in \Omega$. Then using (47), we have

$$\langle \nabla f(x_\star), x - x_\star \rangle \geqslant 0 \ \ \forall x \in \Omega \ \ \text{ if and only if } \ \ \langle \nabla f(x_\star), u \rangle \geqslant 0 \ \ \forall u \in \ker(A)\,.$$

But since $\ker A$ is a subspace, this can hold if and only if $\langle \nabla f(x_\star), u \rangle = 0$ for all $u \in \ker(A)$. In particular, this means, $\nabla f(x_\star)$ must lie in $\ker(A)^\perp$. Since we have that $\mathbb{R}^d = \ker(A) \oplus \operatorname{Im}(A^T)$, this means that $\nabla f(x_\star) = A^T\lambda$ for some $\lambda \in \mathbb{R}^n$.

To summarize, this means that $x_\star$ is optimal for f over Ω if and only if there $\exists \lambda_\star \in \mathbb{R}^m$ such that

$$\begin{cases} \nabla f(x_\star) + A^T\lambda_\star = 0 \\ Ax_\star = b \end{cases}$$

These optimality conditions are known as the *Karush-Kuhn-Tucker Conditions* or *KKT Conditions*.

As an example, consider the equality constrained quadratic optimization problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Q x + c^T x \\ \text{subject to} & Ax = b \end{array}$$

The KKT conditions can be expressed in matrix form

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix}.$$

13.2 Nonlinear constraints

Let Ω be a closed convex set. Let's define the *tangent cone* of Ω at x as

$$\mathcal{T}_\Omega(x) = \text{cone}\{z - x : z \in \Omega\}$$

The tangent cone is the set of all directions that can move from x and remain in Ω. We can also define the *normal cone* of Ω at x to be the set

$$\mathcal{N}_\Omega(x) = \mathcal{T}_\Omega(x)^\circ = \{u \ : \ \langle u, v\rangle \leqslant 0, \ \ \forall v \in \mathcal{T}_\Omega(x)\}$$

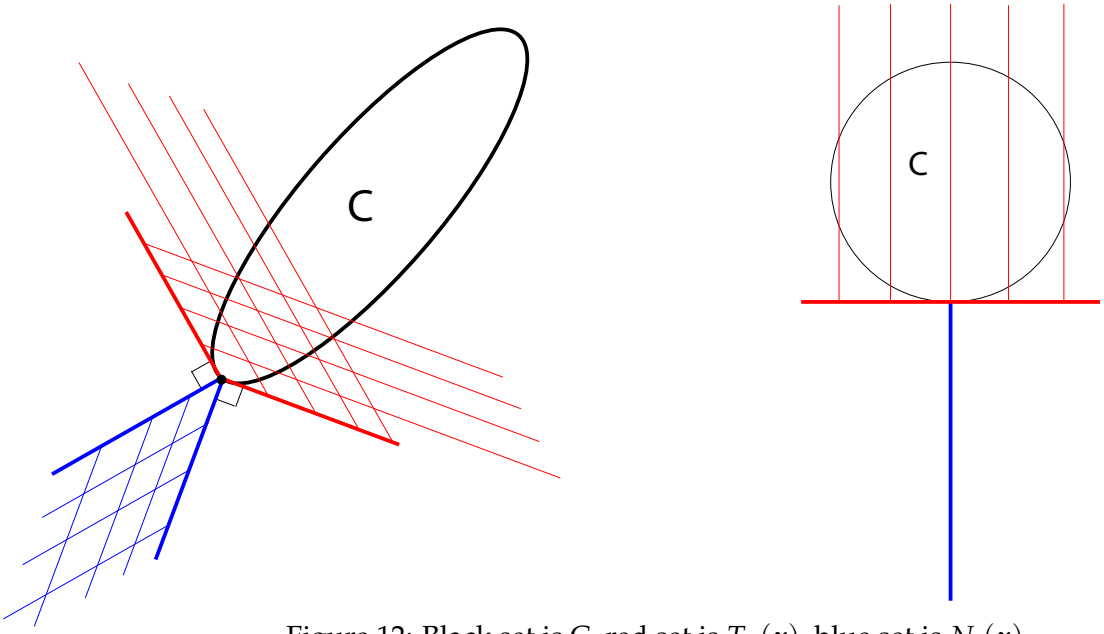


Figure 12: Black set is C, red set is $T_C(x)$, blue set is $N_c(x)$

Suppose we want to minimize a continuously differentiable function f over the intersection of a closed convex set Ω and an affine set $\mathcal{A} = \{x : Ax = b\}$

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega \\ & Ax = b \end{array} \tag{48}$$

where A is again a full rank $n \times d$ matrix. In this section, we will generalize (47) to show

Proposition 13.1. *$x_\star$ is optimal for (48) if and only if there exists $\lambda_\star \in \mathbb{R}^n$ such that*

$$\begin{cases} -\nabla[f(x_\star) + A^\top \lambda_\star] \in \mathcal{N}_\Omega(x_\star) \\ x_\star \in \Omega \cap \mathcal{A} \end{cases} .$$

The key to our analysis here will be to rely on convex analytic arguments. Note that when there is no equality constraint $Ax = b$, our constrained optimality condition is completely equivalent to the assertion

$$-\nabla f(x_\star) \in \mathcal{N}_\Omega(x_\star) . \tag{49}$$

Thus, to prove Proposition 13.1, it will suffice for us to understand the normal cone of the set $\Omega \cap \mathcal{A}$ at the point $x_\star$. To obtain a reasonable characterization, we begin by proving a general fact.

Proposition 13.2. *Let $\Omega \subseteq \mathbb{R}^d$ be a closed convex set. Let $\mathcal{A}$ denote the affine set $\{x \,:\, Ax = b\}$ for some $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$. Suppose that the set $\mathrm{ri}(\Omega) \cap \mathcal{A}$ is non-empty. Then for any $x \in \Omega \cap \mathcal{A}$,*

$$\mathcal{N}_{\Omega\cap\mathcal{A}}(x) = \mathcal{N}_\Omega(x) + \{A^T\lambda \,:\, \lambda \in \mathbb{R}^n\} .$$

Proof. The "$\supseteq$" assertion is straightforward. To see this, suppose $z \in \Omega \cap \mathcal{A}$ and note that $z - x \in \mathrm{null}(A)$ so that $(z-x)^\top A^\top \lambda = \lambda^\top A(z-x) = 0$ for all $\lambda \in \mathbb{R}^n$. If $u \in \mathcal{N}_\Omega(x)$, then $(z-x)^\top u \leqslant 0$, so for $\lambda \in \mathbb{R}^n$, we have

$$\langle z - x, u + A^\top \lambda\rangle = \langle z - x, u\rangle \leqslant 0$$

implying that $u + A^\top\lambda \in \mathcal{N}_{\Omega\cap\mathcal{A}}(x)$. For the reverse inclusion, let $v \in \mathcal{N}_{\Omega\cap\mathcal{A}}(x)$. Then we have

$$v^\top(z-x) \leqslant 0 \text{ for all } z \in \Omega \cap \mathcal{A}$$

Now define the sets

$$C_1 = \left\{(y,\mu) \in \mathbb{R}^{d+1} \,:\, y = z - x,\ z \in \Omega,\ \mu \leqslant v^\top y\right\}$$
$$C_2 = \left\{(y,\mu) \in \mathbb{R}^{d+1} \,:\, y \in \ker(A),\ \mu = 0\right\} .$$

Note that $\mathrm{ri}(C_1) \cap C_2 = \emptyset$ because otherwise there would exist a $(\widehat{y}, \widehat{\mu})$ such that

$$v^\top \widehat{y} > \widehat{\mu} = 0$$

and $\widehat{y} \in \mathcal{T}_{\Omega\cap\mathcal{A}}(x)$. This would contradict our assumption that $v \in \mathcal{N}_{\Omega\cap\mathcal{A}}(x)$. Since their intersection is empty, we can properly separate $\mathrm{ri}(C_1)$ from C_2. Indeed, since C_2 is a subspace and C_1 has nonempty relative interior, there must be a (w,γ) such that

$$\inf_{(y,\mu)\in C_1} \{w^\top y + \gamma\mu\} < \sup_{(y,\mu)\in C_1} \{w^\top y + \gamma\mu\} \leqslant 0$$

while

$$w^\top u = 0 \text{ for all } u \in \ker(A).$$

In particular, this means that $w = A^T\lambda$ for some $\lambda \in \mathbb{R}^n$. Now, γ must be nonnegative, as otherwise,

$$\sup_{(y,\mu)\in C_1} \{w^\top y + \gamma\mu\} = \infty$$

(which can be seen by letting μ tend to negative infinity). If $\gamma = 0$, then

$$\sup_{y\in C_1} w^\top y \leqslant 0$$

but the set $\{y \,:\, w^T y = 0\}$ does not contain the set $\{z - x \,:\, z \in \Omega\}$ as the infimum is strictly negative. This means that the relative interior of $\Omega - \{x\}$ cannot intersect the kernel of A which contradicts our assumptions. Thus, we can conclude that γ is strictly positive. By homogeneity, we may as well assume that $\gamma = 1$.

To complete the argument, note that we now have

$$(w + v)^\top (z - x) \leqslant 0 \text{ for all } z \in \Omega.$$

This means that $v + w \in \mathcal{N}_\Omega(x)$ and we have already shown that $w = A^T\lambda$. Thus,

$$v = (v + w) - w \in \mathcal{N}_\Omega(x) + \mathcal{N}_\mathcal{A}(x)\,.$$

■

Let's now translate the consequence of this proposition for our problem. Using (49) and Proposition 13.2, we have that $x_\star$ is optimal for

$$\min f(x) \quad s.t \quad x \in \Omega, \;\; Ax = b$$

if and only if $Ax_\star = b$ and there exists a $\lambda \in \mathbb{R}^n$ such that

$$-\nabla[f(x_*) + A^\top\lambda] \in \mathcal{N}_\Omega(x_\star) \quad \forall x \in \Omega\,.$$

This reduction is not immediately useful to us, as it doesn't provide an algorithmic approach to solving the constrained optimization problem. However, it will form the basis of our dive into duality.

13.3 Duality

Duality lets us associate to any constrained optimization problem, a concave maximization problem whose solutions lower bound the optimal value of the original problem. In particular, under mild assumptions, we will show that one can solve the primal problem by first solving the dual problem.

We'll continue to focus on the standard primal problem for an equality constrained optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega \\ & Ax = b \end{array} \tag{50}$$

Here, assume that Ω is a closed convex set, f is differentiable, and A is full rank.

The key behind duality (here, Lagrangian duality) is that problem (50) is equivalent to

$$\min_{x \in \Omega} \max_{\lambda \in \mathbb{R}^n} f(x) + \lambda^T (Ax - b)$$

To see this, just note that if $Ax \neq b$, then the max with respect to λ is infinite. On the other hand, if $Ax = b$ is feasible, then the max with respect to λ is equal to $f(x)$.

The *dual problem* associated with (50) is

$$\max_{\lambda \in \mathbb{R}^n} \min_{x \in \Omega} f(x) + \lambda^T (Ax - b)$$

Note that the function

$$q(\lambda) := \min_{x \in \Omega} f(x) + \lambda^T (Ax - b)$$

is always a concave function as it is a minimum of linear functions. Hence the dual problem is a concave maximization problem, regardless of what form f and Ω take. We now show that it always lower bounds the primal problem.

13.4 Weak duality

Proposition 13.3. *For any function $\varphi(x, z)$,*

$$\min_x \max_z \varphi(x, z) \geqslant \max_z \min_x \varphi(x, z)\,.$$

Proof. The proof is essentially tautological. Note that we always have

$$\varphi(x, z) \geqslant \min_x \varphi(x, z)$$

Taking the maximization with respect to the second argument verifies that

$$\max_z \varphi(x, z) \geqslant \max_z \min_x \varphi(x, z) \quad \forall x\,.$$

Now, minimizing the left hand side with respect to x proves

$$\min_x \max_z \varphi(x, z) \geqslant \max_z \min_x \varphi(x, z)$$

which is precisely our assertion. ■

13.5 Strong duality

For convex optimization problems, we can prove a considerably stronger result. Namely, that the primal and dual problems attain the same optimal value. And, moreover, that if we know a dual optimal solution, we can extract a primal optimal solution from a simpler optimization problem.

Theorem 13.4 (Strong Duality)**.**

1. *If* $\exists z \in \text{relint}(\Omega)$ *that also satisfies our equality constraint, and the primal problem has an optimal solution, then the dual has an optimal solution and the primal and dual values are equal*
2. *In order for* $x_\star$ *to be optimal for the primal and* $\lambda_\star$ *optimal for the dual, it is necessary and sufficient that* $Ax_\star = b$, $x_\star \in \Omega$ *and*

$$x_\star \in \arg\min_{x \in \Omega} \ \mathcal{L}(x, \lambda_\star) = f(x) + {\lambda_\star}^T (Ax - b)$$

Proof. For all λ and all feasible x

$$q(\lambda) \leqslant f(x) + \lambda(Ax - b) = f(x)$$

where the second equality holds because $Ax = b$.

Now by Proposition 13.1, $x_\star$ is optimal if and only if there exists a $\lambda_\star$ such that

$$\langle \nabla f(x_\star) + A^T \lambda_\star, x - x_\star \rangle \geqslant 0 \ \ \forall x \in \Omega$$

and $Ax_\star = b$. Note that this condition means that $x_\star$ minimizes $\mathcal{L}(x, \lambda_\star)$ over Ω.

By preceding argument, it now follows that

$$\begin{aligned} q(\lambda_\star) &= \inf_{x \in \Omega} \mathcal{L}(x, \lambda_\star) \\ &= \mathcal{L}(x_\star, \lambda_\star) \\ &= f(x_\star) + {\lambda_\star}^T (Ax_\star - b) = f(x_\star) \end{aligned}$$

which completes the proof. ■

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.

14.1 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 14.1 (Concave functions)**.** A function f is concave $\iff$ $-f$ is convex.

Fact 14.2. *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. ■

14.2 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.

14.3 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 14.3 (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

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

14.4 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 14.4 (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 14.5 (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.

14.5 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 multipliers, 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 14.6. *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

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

15 Fenchel duality and algorithms

In this section, we introduce the Fenchel conjugate. First, recall that for a real-valued convex function of a single variable $f(x)$, we call $f^*(p) := \sup_x px - f(x)$ its Legendre transform. This is illustrated in figure 13.

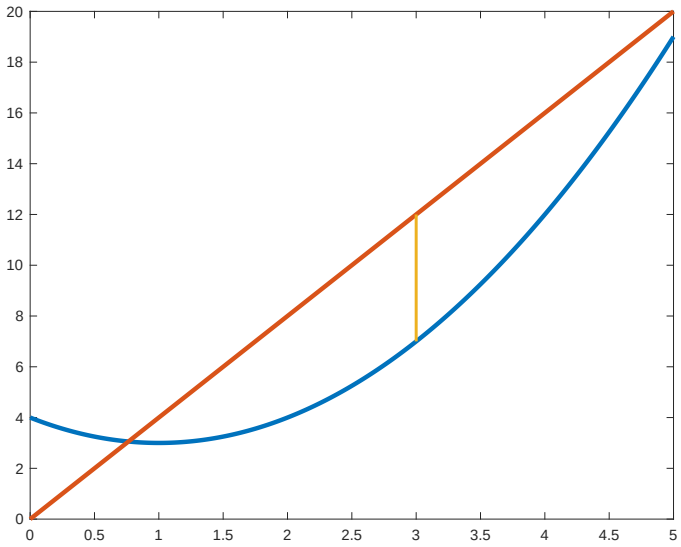


Figure 13: Legendre Transform. The function $f^*(p)$ is how much we need to vertically shift the epigraph of f so that the linear function px is tangent to f at x.

The generalization of the Legendre transform to (possibly nonconvex) functions of multiple variables is the Fenchel conjugate.

Definition 15.1 (Fenchel conjugate)**.** The Fenchel conjugate of $f : \mathbb{R}^n \to \mathbb{R}$ is

$$f^*(p) = \sup_x \langle p, x \rangle - f(x)$$

We now present several useful facts about the Fenchel conjugate. The proofs are left as an exercise.

Fact 15.2. *f^* is convex.*

Indeed, f^* is the supremum of affine functions and therefore convex. Thus, the Fenchel conjugate of f is also known as its convex conjugate.

Fact 15.3. *$f^*(f^*(x)) = f$ if f is convex.*

In other words, the Fenchel conjugate is its own inverse for convex functions. Now, we can also relate the subdifferential of a function to that of its Fenchel conjugate. Intuitively, observe that $0 \in \partial f^*(p) \iff 0 \in p - \partial f(x) \iff p \in \partial f(x)$. This is summarized more generally in the following fact.

Fact 15.4. *The subdifferential of f^* at p is $\partial f^*(p) = \{x : p \in \partial f(x)\}$.*

Indeed, $\partial f^*(0)$ is the set of minimizers of f.

In the following theorem, we introduce Fenchel duality.

Theorem 15.5 (Fenchel duality)**.** *Suppose we have f proper convex, and g proper concave. Then*

$$\min_x f(x) - g(x) = \max_p g^*(p) - f^*(p)$$

where g^* is the concave conjugate of g, defined as $\inf_x \langle p, x\rangle - g(x)$.

In the one-dimensional case, we can illustrate Fenchel duality with Figure 14.

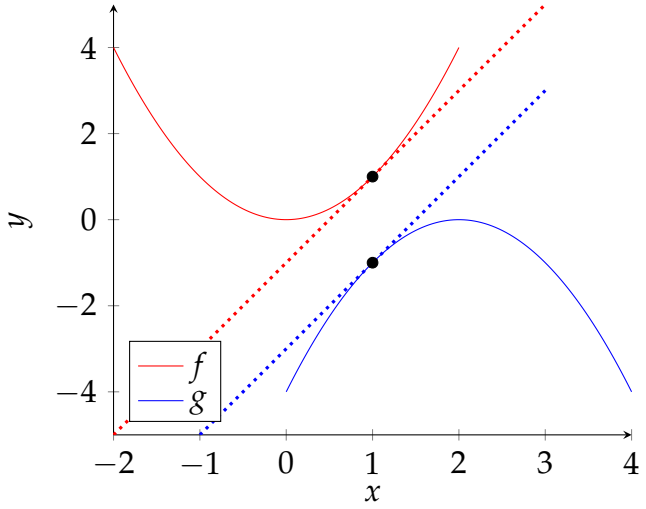


Figure 14: Fenchel duality in one dimension

In the minimization problem, we want to find x such that the vertical distance between f and g at x is as small as possible. In the (dual) maximization problem, we draw tangents to the graphs of f and g such that the tangent lines have the same slope p, and we want to find p such that the vertical distance between the tangent lines is as large as possible. The duality theorem above states that strong duality holds, that is, the two problems have the same solution.

We can recover Fenchel duality from Lagrangian duality, which we have already studied. To do so, we need to introduce a constraint to our minimization problem in Theorem 15.5. A natural reformulation of the problem with a constraint is as follows.

$$\min_{x,z} f(x) - g(z) \text{ subject to } x = z \tag{51}$$

15.1 Deriving the dual problem for empirical risk minimization

In empirical risk minimization, we often want to minimize a function of the following form:

$$P(w) = \sum_{i=1}^{m} \phi_i(\langle w, x_i \rangle) + R(w) \tag{52}$$

We can think of $w \in \mathbb{R}^n$ as the model parameter that we want to optimize over (in this case it corresponds to picking a hyperplane), and x_i as the features of the i-th example in the training set. $\phi_i(\cdot, x_i)$ is the loss function for the i-th training example and may depend on its label. $R(w)$ is the regularizer, and we typically choose it to be of the form $R(w) = \frac{\lambda}{2}\|w\|^2$.

The primal problem, $\min_{w \in \mathbb{R}^n} P(w)$, can be equivalently written as follows:

$$\min_{w,z} \sum_{i=1}^{m} \phi_i(z_i) + R(w) \text{ subject to } X^\top w = z \tag{53}$$

By Lagrangian duality, we know that the dual problem is the following:

$$\begin{aligned}
&\max_{\alpha \in \mathbb{R}^m} \min_{z,w} \sum_{i=1}^{m} \phi_i(z_i) + R(w) - \alpha^\top (X^\top w - z) \\
&= \max_{\alpha} \min_{w,z} \sum_{i=1}^{m} \phi_i(z_i) + \alpha_i z_i + R(w) - \alpha^\top X^\top w \\
&= \max_{\alpha} \left(- \min_{w,z} - \left(\sum_{i=1}^{m} \phi_i(z_i) + \alpha_i z_i \right) + (X\alpha)^\top w - R(w) \right) \\
&= \max_{\alpha} - \left(\sum_{i=1}^{m} \max_{z_i} \left(-\phi_i(z_i) - \alpha_i z_i \right) + \max_{w} (X\alpha)^\top w - R(w) \right) \\
&= \max_{\alpha} - \sum_{i=1}^{n} \phi_i^*(-\alpha_i) - R^*(X\alpha)
\end{aligned}$$

where ϕ_i^* and R^* are the Fenchel conjugates of ϕ_i and R^* respectively. Let us denote the dual objective as $D(\alpha) = \sum_{i=1}^{m} \phi_i^*(-\alpha_i) - R^*(X\alpha)$. By weak duality, $D(\alpha) \leqslant P(w)$.

For $R(w) = \frac{\lambda}{2}\|w\|^2$, $R^*(p) = \frac{\lambda}{2}\|\frac{1}{\lambda}p\|^2$. So R is its own convex conjugate (up to correction by a constant factor). In this case the dual becomes:

$$\max_{\alpha} \sum_{i=1}^{m} \phi_i^*(-\alpha_i) - \frac{\lambda}{2}\|\frac{1}{\lambda}\sum_{i=1}^{m} \alpha_i x_i\|^2$$

We can relate the primal and dual variables by the map $w(\alpha) = \frac{1}{\lambda}\sum_{i=1}^{m} \alpha_i x_i$. In particular, this shows that the optimal hyperplane is in the span of the data. Here are some examples of models we can use under this framework.

Example 15.6 (Linear SVM)**.** We would use the hinge loss for ϕ_i. This corresponds to

$$\phi_i(w) = \max(0, 1 - y_i x_i^\top w)\,, \quad -\phi_i^*(-\alpha_i) = \alpha_i y_i$$

Example 15.7 (Least-squares linear regression)**.** We would use the squared loss for ϕ_i. This corresponds to

$$\phi_i(w) = (w^\top x_i - y_i)^2\,, \quad -\phi_i^*(-\alpha_i) = \alpha_i y_i + \alpha^2/4$$

We end with a fact that relates the smoothness of ϕ_i to the strong convexity of ϕ_i^*.

Fact 15.8. *If ϕ_i is $\frac{1}{\gamma}$-smooth, then ϕ_i^* is γ-strongly convex.*

15.2 Stochastic dual coordinate ascent (SDCA)

In this section we discuss a particular algorithm for empirical risk minimization which makes use of Fenchel duality. Its main idea is picking an index $i \in [m]$ at random, and then solving the dual problem at coordinate i, while keeping the other coordinates fixed.

More precisely, the algorithm performs the following steps:

1. Start from $w^0 := w(\alpha^0)$
2. For $t = 1, \ldots, T$:
   (a) Randomly pick $i \in [m]$
   (b) Find $\Delta\alpha_i$ which maximizes

$$-\Phi_i\left(-(\alpha_i^{t-1} + \Delta\alpha_i)\right) - \frac{\lambda}{2m}\left\|w^{t-1} + \frac{1}{\lambda}\Delta\alpha_i x_i\right\|^2$$

3. Update the dual and primal solution
   (a) $\alpha^t = \alpha^{t-1} + \Delta\alpha_i$
   (b) $w^t = w^{t-1} + \frac{1}{\lambda}\Delta\alpha_i x_i$

For certain loss functions, the maximizer $\Delta\alpha_i$ is given in closed form. For example, for hinge loss it is given explicitly by:

$$\Delta\alpha_i = y_i \max\left(0, \min(1, \frac{1 - x_i^T w^{t-1} y_i}{\|x_i\|^2/\lambda m} + \alpha_i^{t-1} y_i)\right) - \alpha_i^{t-1},$$

and for squared loss it is given by:

$$\Delta\alpha_i = \frac{y_i - x_i^T w^{t-1} - 0.5\alpha_i^{t-1}}{0.5 + \|x\|^2/\lambda m}.$$

Note that these updates require both the primal and dual solutions to perform the update.

Now we state a lemma given in [SSZ13], which implies linear convergence of SDCA. In what follows, assume that $\|x_i\| \leqslant 1$, $\Phi_i(x) \geqslant 0$ for all x, and $\Phi_i(0) \leqslant 1$.

Lemma 15.9. *Assume Φ_i^* is γ-strongly convex, where $\gamma > 0$. Then:*

$$\mathbb{E}[D(\alpha^t) - D(\alpha^{t-1})] \geqslant \frac{s}{m}\mathbb{E}[P(w^{t-1}) - D(\alpha^{t-1})],$$

where $s = \frac{\lambda m \gamma}{1+\lambda m \gamma}$.

We leave out the proof of this result, however give a short argument that proves linear convergence of SDCA using this lemma. Denote $\epsilon_D^t := D(\alpha^*) - D(\alpha^t)$. Because the dual solution provides a lower bound on the primal solution, it follows that:

$$\epsilon_D^t \leqslant P(w^t) - D(\alpha^t).$$

Further, we can write:

$$D(\alpha^t) - D(\alpha^{t-1}) = \epsilon_D^{t-1} - \epsilon_D^t.$$

By taking expectations on both sides of this equality and applying Lemma 15.9, we obtain:

$$\begin{aligned}\mathbb{E}[\epsilon_D^{t-1} - \epsilon_D^t] &= \mathbb{E}[D(\alpha^t) - D(\alpha^{t-1})] \\ &\geqslant \frac{s}{m}\mathbb{E}[P(w^{t-1}) - D(\alpha^{t-1})] \\ &\geqslant \frac{s}{m}\mathbb{E}[\epsilon_D^{t-1}].\end{aligned}$$

Rearranging terms and recursively applying the previous argument yields:

$$\mathbb{E}[\epsilon_D^t] \leqslant (1 - \frac{s}{m})\mathbb{E}[\epsilon_D^{t-1}] \leqslant (1 - \frac{s}{m})^t \epsilon_D^0.$$

From this inequality we can conclude that we need $O(m + \frac{1}{\lambda\gamma}\log(1/\epsilon))$ steps to achieve ϵ dual error.

Using Lemma 15.9, we can also bound the primal error. Again using the fact that the dual solution underestimates the primal solution, we provide the bound in the following way:

$$\begin{aligned}\mathbb{E}[P(w^t) - P(w^*)] &\leqslant \mathbb{E}[P(w^t) - D(\alpha^t)] \\ &\leqslant \frac{m}{s}\mathbb{E}[D(\alpha^{t+1}) - D(\alpha^t)] \\ &\leqslant \frac{m}{s}\mathbb{E}[\epsilon_D^t],\end{aligned}$$

where the last inequality ignores the negative term $-\mathbb{E}[\epsilon_D^{t-1}]$.

16 Backpropagation and adjoints

From now onward, we give up the luxuries afforded by convexity and move to the realm of non-convex functions. In the problems we have seen so far, deriving a closed-form expression for the gradients was fairly straightforward. But, doing so can be a challenging task for general non-convex functions. In this class, we are going to focus our attention on functions that can be expressed as compositions of multiple functions. In what follows, we will introduce *backpropagation* - a popular technique that exploits the composite nature of the functions to compute gradients incrementally.

16.1 Warming up

A common view of backpropagation is that "*it's just chain rule*". This view is not particular helpful, however, and we will see that there is more to it. As a warm-up example, let us look at the following optimization problem with $g : \mathbb{R}^n \to \mathbb{R}$, $f : \mathbb{R} \to \mathbb{R}$:

$$\min_x f(g(x)). \tag{54}$$

Using a similar trick to the one we saw in the context of ADMM, we can rewrite this problem as

$$\begin{aligned} &\min_x f(z) \\ &\text{s.t. } z = g(x) \end{aligned} \tag{55}$$

Note that we have converted our original unconstrained problem in x into a constrained optimization problem in x and z with the following Lagrangian,

$$\mathcal{L}(x, z, \lambda) = f(z) + \lambda(g(x) - z). \tag{56}$$

Setting $\nabla\mathcal{L} = 0$, we have the following optimality conditions,

$$0 = \nabla_x \mathcal{L} = \lambda g'(x) \Leftrightarrow 0 = \lambda g'(x) \tag{57a}$$
$$0 = \nabla_z \mathcal{L} = f'(z) - \lambda \Leftrightarrow \lambda = f'(z) \tag{57b}$$
$$0 = \nabla_\lambda \mathcal{L} = g(x) - z \Leftrightarrow z = g(x) \tag{57c}$$

which implies

$$\begin{aligned} 0 &= f'(g(x))g'(x) \\ &= \nabla_x f(g(x)) \qquad \text{(by chain rule)} \end{aligned}$$

Hence, solving the Lagrangian equations gave us an incremental way of computing gradients. As we will see shortly, this holds at great generality. It is important to notice that we did *not* use the chain rule when solving the equations in (57). The chain rule only showed up in the proof of correctness.

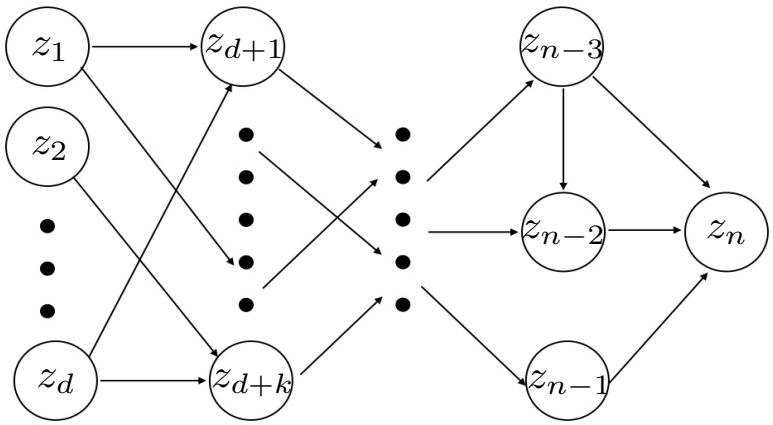


Figure 15: Computation graph

16.2 General formulation

Any composite function can be described in terms of its computation graph. As long as the elementary functions of the computation graph are differentiable, we can perform the same procedure as above. Before moving ahead, let us introduce some notation:

- Directed, acyclic computation graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Number of vertices: $|V| = n$
- Set of ancestors of i^{th} node: $\alpha(i) = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$
- Set of successors of i^{th} node: $\beta(i) = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$
- Computation at the i^{th} node: $f_i(z_{\alpha(i)}), \quad f_i : \mathbb{R}^{|\alpha(i)|} \to \mathbb{R}^{|\beta(i)|}$
- Nodes:
  - Input - $z_1, \ldots, z_d$
  - Intermediate - $z_{d+1}, \ldots, z_{n-1}$
  - Output - z_n

Then, the general formulation is given by

$$\begin{aligned} &\min \;\; z_n \\ &\text{s.t.} \;\; z_i = f_i(z_{\alpha(i)}). \end{aligned} \tag{58}$$

with the following Lagrangian,

$$\mathcal{L}(x, z, \lambda) = z_n - \sum_i \lambda_i (z_i - f_i(z_{\alpha(i)})). \tag{59}$$

As in the warm-up example, we set $\nabla \mathcal{L} = 0$. This can be viewed as an algorithm comprising two separate steps:

Backpropagation algorithm

- *Step 1*: Set $\nabla_\lambda \mathcal{L} = 0$, i.e.,

$$\nabla_{\lambda_i} \mathcal{L} = z_i - f_i(z_{\alpha(i)}) = 0 \Leftrightarrow z_i = f_i(z_{\alpha(i)}) \tag{60}$$

  Observation: This is known as *forward pass or forward propagation* as the values at the nodes (z_i) are computed using the values of the ancestors.

- *Step 2*: Set $\nabla_{z_j} \mathcal{L} = 0$,
  - for $j = n$,

$$\begin{aligned} 0 = \nabla_{z_j} \mathcal{L} &= 1 - \lambda_n \\ \Leftrightarrow \lambda_n &= 1 \end{aligned}$$

  - for $j < n$,

$$\begin{aligned} 0 &= \nabla_{z_j} \mathcal{L} \\ &= \nabla_{z_j} (z_n - \sum_i \lambda_i (z_i - f_i(z_{\alpha(i)}))) \\ &= -\sum_i \lambda_i (\nabla_{z_j}[z_i] - \nabla_{z_j} f_i(z_{\alpha(i)})) \\ &= -\lambda_j + \sum_i \lambda_i \nabla_{z_j} f_i(z_{\alpha(i)}) \\ &= -\lambda_j + \sum_{i \in \beta(j)} \lambda_i \frac{\partial f_i(z_{\alpha(i)})}{\partial z_j} \\ \Leftrightarrow \quad \lambda_j &= \sum_{i \in \beta(j)} \lambda_i \frac{\partial f_i(z_{\alpha(i)})}{\partial z_j} \end{aligned}$$

  Observation: This is known as *backward pass or backpropagation* as $\lambda_i's$ are computed using the gradients and values of λ at successor nodes in the computation graph.

16.3 Connection to chain rule

In this section, we will prove a theorem that explains why backpropagation allows us to calculate gradients incrementally.

Theorem 16.1. *For all $1 \leqslant j \leqslant n$, we have*

$$\lambda_j = \frac{\partial f(x)}{\partial z_j},$$

i.e., the partial derivative of the function f at x w.r.t to the j^{th} node in the graph.

Proof. We assume for simplicity that the computation graph has L layers and edges exist only between consecutive layers, i.e., $f = f_L \circ \cdots \circ f_1$. The proof is by induction over layers (starting from the output layer).
Base case: $\lambda_n = 1 = \frac{\partial f_n(x)}{\partial z_n} = \frac{\partial z_n}{z_n}$.

Induction: Fix p^{th} layer and assume claim holds for nodes in all subsequent layers $l > p$.

Then, for node z_j in layer p,

$$\begin{aligned}
\lambda_j &= \sum_{i \in \beta(j)} \lambda_i \frac{\partial f_i(z_{\alpha(i)})}{\partial z_j} \\
&= \sum_{i \in \beta(j)} \frac{\partial f(x)}{\partial z_i} \frac{\partial z_i}{z_j} \qquad (z_{\beta(j)} \text{ belong to layer } p+1) \\
&= \frac{\partial f(x)}{\partial z_j} \qquad \text{(by multivariate chain rule).}
\end{aligned}$$

■

($*$) Note that the proof for arbitrary computation graphs is by induction over the partial order induced by the reverse computation graph.

Remarks

1. Assuming elementary node operations cost constant time, cost of both the forward and backward pass is $O(|\mathcal{V}| + |\mathcal{E}|) \Rightarrow$ Linear time!
2. Notice that the algorithm itself does not use the chain rule, only its correctness guarantee uses it.
3. Algorithm is equivalent to the "*method of adjoints*" from control theory introduced in the 60's. It was rediscovered by Baur and Strassen in '83 for computing partial derivatives [BS83]. More recently, it has received a lot of attention due to its adoption by the Deep Learning community since the 90's.
4. This algorithm is also known as *automatic differentiation*, not to be confused with
   (a) Symbolic differentiation
   (b) Numerical differentiation

16.4 Working out an example

Suppose we have a batch of data $X \in \mathbb{R}^{n \times d}$ with labels $y \in \mathbb{R}^n$. Consider a two-layer neural net given by weights $W_1, W_2 \in \mathbb{R}^{d \times d}$:

$$f(W_1, W_2) = \|\sigma(XW_1)W_2 - y\|^2$$

To compute gradients, we only need to implement forward/backward pass for the elementary operations:

- Squared norm
- Subtraction/addition
- Component-wise non-linearity σ
- Matrix multiplication

Observe that the partial derivatives for the first three operations are easy to compute. Hence, it suffices to focus on matrix multiplication.

Back-propagation for matrix completion

The two steps of the backpropagation algorithm in this context are:

Forward Pass:

- Input: $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times d}$
- Output: $C = AB \in \mathbb{R}^{m \times d}$

Backward Pass:

- Input: Partial derivatives $\Lambda \in \mathbb{R}^{m \times d}$ (also A, B, C from forward pass)
- Output:
  - $\Lambda_1 \in \mathbb{R}^{m \times n}$ (partial derivatives for left input)
  - $\Lambda_2 \in \mathbb{R}^{n \times d}$ (partial derivatives for right input)

Claim 16.2. $\Lambda_1 = \Lambda B^T, \Lambda_2 = A^T \Lambda$

Proof.

$$f = \sum_{i,j} \lambda_{ij} C_{ij} = \sum_{i,j} (AB)_{ij} = \sum_{i,j} \lambda_{ij} \Sigma_k a_{ik} b_{kj}.$$

So, by Lagrange update rule,

$$(\Lambda_1)_{pq} = \frac{\partial f}{\partial a_{pq}} = \sum_{i,j,k} \lambda_{ij} \frac{\partial a_{ik}}{\partial a_{pq}} b_{kj} = \sum_j \lambda_{pj} b_{qj} = (\Lambda B^T)_{pq}.$$

Using the same approach for partial derivative w.r.t. B, we get

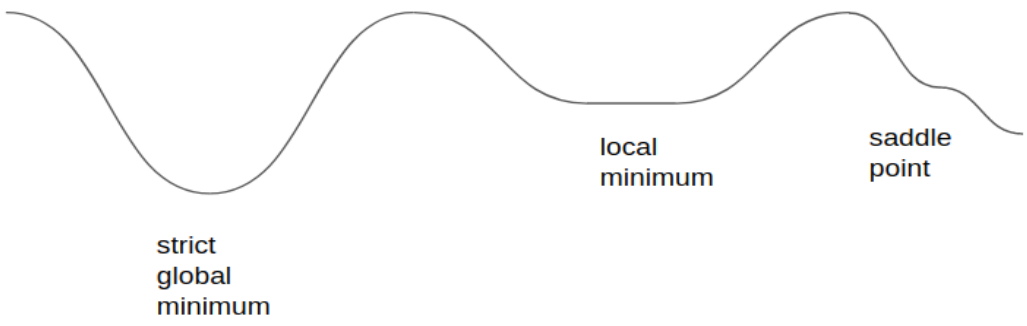
$$(\Lambda_2)_{pq} = (A^T \Lambda)_{pq}$$

■

Part V
Non-convex problems

17 Non-convex problems

This lecture provides the important information on how non-convex problems differ from convex problems. The major issue in non-convex problems is that it can be difficult to find the global minimum because algorithms can easily get stuck in the possibly numerous local minima and saddle points.



17.1 Local minima

Definition 17.1 (Local minimum)**.** A point x^* is an unconstrained *local minimum* if there exist $\epsilon > 0$ such that $f(x^*) \leqslant f(x)$ for all x with $\|x - x^*\| < \epsilon$.

Definition 17.2 (Global minimum)**.** A point x^* is an unconstrained *global minimum* if $f(x^*) \leqslant f(x)$ for all x.

For both definitions, we say "strict" if these inequalities are strict.

Proposition 17.3 (Necessary Conditions for local minimum)**.** *Let x^* be an unconstrained local minimum of $f : \mathbb{R}^n \to \mathbb{R}$ and assume f is continuously differentiable (C^1) in an open set containing x^*. Then*

1. $\nabla f(x^*) = 0$ *(First-Order Necessary Condition)*
2. *If in addition f is twice continuously differentiable in an open set around x^*, then $\nabla^2 f(x^*) \succeq 0$. (Second Order Necessary Condition)*

Proof of Proposition 17.3. Fix any direction $d \in \mathbb{R}^n$.

1. $g(\alpha) := f(x^* + \alpha d)$. Then

$$\begin{aligned} 0 &\leqslant \lim_{\alpha \to 0} \frac{f(x^* + \alpha d) - f(x^*)}{\alpha} \\ &= \frac{\partial g(0)}{\partial \alpha} \\ &= d^\top \nabla f(x^*) \end{aligned} \tag{61}$$

   Inequality 61 follows because x^* is a local minimum, $0 \leqslant f(x^* + \alpha d) - f(x^*)$ for sufficiently small alpha. So, we can construct a sequence with only positive α that converges to x^* such that each element $0 \leqslant \frac{f(x^* + \alpha_n d) - f(x^*)}{\alpha_n}$ which implies that statement given that f is locally differentiable.

   Since d is arbitrary, this implies that $\nabla f(x^*) = 0$.

2. First we represent $f(x^* + \alpha d) - f(x^*)$ using the 2nd order Taylor expansion.

$$\begin{aligned} f(x^* + \alpha d) - f(x^*) &= \alpha \nabla f(x^*)^\top d + \frac{\alpha^2}{2} d^\top \nabla^2 f(x^*) d + O(\alpha^2) \\ &= \frac{\alpha^2}{2} d^\top \nabla^2 f(x^*) d + O(\alpha^2) \end{aligned}$$

   Now we do the following

$$\begin{aligned} 0 &\leqslant \lim_{\alpha \to 0} \frac{f(x^* + \alpha d) - f(x^*)}{\alpha^2} \\ &= \lim_{\alpha \to 0} \frac{1}{2} d^\top \nabla^2 f(x^*) d + \frac{O(\alpha^2)}{\alpha^2} \\ &= \frac{1}{2} d^\top \nabla^2 f(x^*) d \end{aligned}$$

   Because d is arbitrary, this implies that $\nabla^2 f(x^*) \succeq 0$ (Positive semidefinite).

■

Note that $\nabla f(x^*) = 0$ alone does not imply x^* is a local minimum. Even the necessary conditions $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succeq 0$ does not imply x^* is a local minimum. This is because it could be that $\nabla^2 f(x^*) = 0$, but the 3rd order is not 0. For example in the 1d case, $x^* = 0$ for $f(x) = x^3$ satisfies these conditions, but is not a local minimum. Now, we will look at the actual sufficient conditions for a local minimum, but these conditions can only detect strict local minima.

Proposition 17.4 (Sufficient conditions for strict local minimum)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable (C^2) over an open set S. Suppose $x \in S$ such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x) \succ 0$ (positive definite). Then, x^* is a strict unconstrained local minimum.*

Proof of Proposition 17.4. Fix $d \in \mathbb{R}^n$. Note that $d^\top \nabla^2 f(x^*) d \geqslant \lambda_{\min} \|d\|^2$, where $\lambda_{\min}$ is the smallest eigenvalue of $\nabla^2 f(x^*)$.

$$
\begin{aligned}
f(x^* + d) - f(x^*) &= \nabla f(x^*)^\top d + \frac{1}{2} d^\top \nabla^2 f(x^*) d + O(\|d\|^2) && (62)\\
&\geqslant \frac{\lambda_{\min}}{2} \|d\|^2 + O(\|d\|^2) \\
&= \left(\frac{\lambda_{\min}}{2} + \frac{O(\|d\|^2)}{\|d\|^2}\right) \|d\|^2 \\
&> 0 && (63)
\end{aligned}
$$

Equality 62 follows from using the 2nd Order Taylor expansion.
Inequality 63 follows for sufficiently small $\|d\|$.

Therefore, x^* must be a strict local minimum. ■

17.2 Stationary points

For non-convex problems we must accept that gradient descent cannot always find the global minimum, but can it at least find a nearby local minimum? It turns out that for many problems we care about it can usually, but not always!

Definition 17.5 (Stationary point)**.** We say a point $x \in \mathbb{R}^n$ is a stationary point of $f\colon \mathbb{R}^n \to \mathbb{R}$ if $\nabla f(x) = 0$.

Proposition 17.6. *Gradient Descent converges to a stationary point.*

Proof of Proposition 17.6. Suppose $x' = x - \eta \nabla f(x)$. From Taylor expansion we get the following

$$
\begin{aligned}
f(x') &= f(x) + \nabla f(x)^\top (x' - x) + O(\|x' - x\|) \\
&= f(x) - \eta \|\nabla f(x)\|^2 + O(\eta \|\nabla f(x)\|) \\
&= f(x) - \eta \|\nabla f(x)\|^2 + O(\eta) && (64)
\end{aligned}
$$

Equality 64 is justified because we control η, and $\|\nabla f(x)\|$ is a constant with respect to η.

Now we need to worry about selecting step sizes.

17.2.1 Minimization Rule / Line Search

Given a descent direction d (example $d = -\nabla f(x)$), let our step rate η be as follows

$$\eta \in \underset{\eta \geqslant 0}{\operatorname{argmin}} f(x + \eta d)$$

Using this procedure is called **Line Search** because we search for the best step size along the direction d. However, exact line search can be expensive due to the argmin.

Instead, we can approximate this minimization by using the **Armijo Rule**. Fix

$$\gamma, s, \sigma < 1$$

Put $\eta = \gamma^m s$ where m is the smallest non-negative integer such that

$$f(x) - f(x + \gamma^m s d) \geqslant -\sigma \gamma^m s \nabla f(x)^\top d$$

Think of s as an initial learning rate. If s causes sufficient decrease then stop, otherwise keep multiplying by γ until you do. Typical choices for parameters are

$$\gamma = \frac{1}{2}, \sigma = \frac{1}{100}, s = 1$$

Notice that as long as d satisfies $-\nabla f(x)^T d > 0$ that the inequality ensures that our function sequence will decrease.

Proposition 17.7. *Assume that f if continuous and differentiable (C^1), and let $\{x_t\}$ be a sequence generated by $x_{t+1} = x_t - \eta_t \nabla f(x_t)$ where η_t is selected by the Armijo rule. Then, every limit point of $\{x_t\}$ is a stationary point.*

Proof of Proposition 17.7. Let $\bar{x}$ be a limit point. By continuity $\{f(x_t)\}$ converges to $f(\bar{x})$ and therefore:

$$f(x_t) - f(x_{t+1}) \to 0$$

By definition of Armijo rule:

$$f(x_t) - f(x_{t+1}) \geqslant -\sigma \eta_t \|\nabla f(x_t)\|^2 \tag{65}$$

Suppose for the sake of contradiction that $\bar{x}$ is not a stationary point of f. Then,

$$\limsup_{t \to \infty} -\|\nabla f(x_t)\|^2 < 0$$

By inequality 65, this must mean that $\eta_t \to 0$. This implies $\exists t_0$ such that $\forall t \geqslant t_0$

$$f(x_t) - f(x_t - \frac{\eta_t}{\gamma} \nabla f(x_t)) < \frac{\sigma \eta_t}{\gamma} \|\nabla f(x_t)\|^2$$

Because $\eta_t \to 0$, we know that after some t_0 all step sizes are chosen with a $m \geqslant 1$. Therefore, going back one iteration of Armijo rule was not good enough to satisfy the inequality or else some previous step size would have been chosen.

Now let $\tilde{\eta}_t = \frac{\eta_t}{\gamma}$ and we can continue as follows

$$\begin{aligned} \frac{f(x_t) - f(x_t - \tilde{\eta}_t \nabla f(x_t))}{\tilde{\eta}_t} &< \sigma \|\nabla f(x)\|^2 && \Rightarrow \\ \nabla f(x_t - \tilde{\eta}_t \nabla f(x_t))^T \nabla f(x_t) &< \sigma \|\nabla f(x)\|^2 && \Rightarrow \quad (66) \\ \|\nabla f(x_t)\|^2 &\leqslant \sigma \|\nabla f(x_t)\|^2 && (67) \end{aligned}$$

Inequality 66 follows from using Mean Value Theorem (MVT)
Inequality 67 follows by taking the limit as $\eta_t \to 0 \Rightarrow \tilde{\eta}_t \to 0$

This is a contradiction because $0 < \sigma < 1$. Therefore, the limit point $\bar{x}$ is a stationary point of f. ■

Therefore, if we can use the Armijo rule to determine step sizes that guarantee that gradient descent will converge to a stationary point. ■

17.3 Saddle points

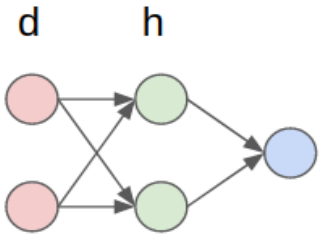
Now that we have found that gradient descent will converge to a stationary point, how concerned should we be that the stationary point is not a local minimum?

Definition 17.8 (Saddle points)**.** Saddle points are stationary points that are not local optima.

This means that if f is twice differentiable then $\nabla^2 f(x)$ has both positive and negative eigenvalues.

17.3.1 How do saddle points arise?

In most non-convex problems there exists several local minima. This is clear to see in problems that have natural symmetry such as in a two layer fully connected neural networks.



Notice that any permutation of the units of the hidden layer would preserve the same function, so we have at least $h!$ local minima. Typically a convex combination of two distinct local minima in a non-convex problem is not a local minimum. In the case where $\nabla f(x)$ is differentiable, then by Mean Value Theorem we know that there must exist another stationary point between any two local minima. So, there often exists at least one saddle point between any two distinct local minima. Hence, many local minima tends to lead to many saddle points.

However, recent work has demonstrated that saddle points are usually not a problem.

1. Gradient descent does not converge to strict saddle points from a random initialization. [GHJY15]
2. Saddle points can be avoided with noise addition. [LPP$^+$17]

18 Escaping saddle points

This lecture formalizes and shows the following intuitive statement for nonconvex optimization:

Gradient descent almost never converges to (strict) saddle points.

The result was shown in [LSJR16]. Let's start with some definitions.

Definition 18.1 (Stationary point)**.** We call x^* a stationary point if the gradient vanishes at x^*, i.e., $\nabla f(x^*) = 0$.

We can further classify stationary points into different categories. One important category are saddle points.

Definition 18.2 (Saddle point)**.** A stationary point x^* is a *saddle point* if for all $\epsilon > 0$, there are points $x, y \in B(x^*; \epsilon)$ s.t. $f(x) \leqslant f(x^*) \leqslant f(y)$.

Definition 18.3 (Strict saddle point)**.** For a twice continuously differentiable function $f \in C^2$, a saddle point x^* is a *strict saddle point* if the Hessian at that point is not positive semidefinite, i.e. $\lambda_{\min}(\nabla^2 f(x^*)) < 0$, where $\lambda_{\min}$ denotes the smallest eigenvalue.

18.1 Dynamical systems perspective

It'll be helpful to think of the trajectory defined by gradient descent as a dynamical system. To do so, we view each gradient descent update as a operator. For a fixed step size η, let

$$g(x) = x - \eta \nabla f(x)$$

so the notation for the result of iteration t from our previous discussion of gradient descent carries over as $x_t = g^t(x_0) = g(g(...g(x_0)))$, where g is applied t times on the initial point x_0. We call g the gradient map. Note that x^* is stationary iff. it is a fixed point of the gradient map i.e. $g(x^*) = x^*$. Also note that $Dg(x) = I - \eta \nabla^2 f(x)$ (Jacobian of g) , a fact that will become important later. Now we formalize a notion of the set of "attractors" of x^*.

Definition 18.4. The global stable set of x^*, is defined as

$$W^S(x^*) = \{x \in \mathbb{R}^n : \lim_t g^t(x) = x^*\}$$

In words, this is the set of points that will eventually converge to x^*.

With this definition out of the way, we can state the main claim formally as follows.

Theorem 18.5. *Assume $f \in C^2$ and is β-smooth. Also assume that the step size $\eta < 1/\beta$. Then for all strict saddle points x^*, its set of attractors $W^S(x^*)$ has Lebesgue measure 0.*

Remark 18.6. *In fact, it could be proven with additional technicalities that the Lebesgue measure of $\bigcup_{\text{strict saddle points } x^*} W^S(x^*)$ is also 0. This is just another way to say that gradient descent almost surely converges to local minima.*

Remark 18.7. *By definition, this also holds true to any probability measure absolutely continuous w.r.t. the Lebesgue measure (e.g. any continuous probability distribution). That is,*

$$\mathbb{P}(\lim_t x_t = x^*) = 0$$

However, the theorem above is only an asymptotic statement. Non-asymptotically, even with fairly natural random initialization schemes and non-pathological functions, gradient descent can be significantly slowed down by saddle points. The most recent result [DJL+17] is that gradient descent takes exponential time to escape saddle points (even though the theorem above says that they do escape eventually). We won't prove this result in this lecture.

18.2 The case of quadratics

Before the proof, let's go over two examples that will make the proof more intuitive:

Example 18.8. $f(x) = \frac{1}{2}x^T Hx$ where H is an n-by-n matrix, symmetric but not positive semidefinite. For convenience, assume 0 is not an eigenvalue of H. So 0 is the only stationary point and the only strict saddle point for this problem.

We can calculate $g(x) = x - \eta Hx = (I - \eta H)x$ and $g^t(x) = (I - \eta H)^t x$. And we know that $\lambda_i(I - \eta H) = 1 - \eta\lambda_i(H)$, where λ_i for $i = 1...n$ could denote any one of the eigenvalues. So in order for $\lim_t g^t(x) = \lim_t (1 - \eta\lambda_i(H))^t x$ to converge to 0, we just need $\lim_t (1 - \eta\lambda_i(H))^t$ to converge to 0, that is, $|1 - \eta\lambda_i(H)| < 1$. This implies that

$$W^S(0) = \text{span}\left\{u | Hu = \lambda u, 0 < \lambda < \frac{\eta}{2}\right\}$$

i.e. the set of eigenvectors for the positive eigenvalues smaller than $\frac{\eta}{2}$. Since η can be arbitrarily large, we just consider the larger set of eigenvectors for the positive eigenvalues. By our assumption on H, this set has dimension lower than n, thus has measure 0.

Example 18.9. Consider the function $f(x, y) = \frac{1}{2}x^2 + \frac{1}{4}y^4 - \frac{1}{2}y^2$ with corresponding gradient update

$$g(x, y) = \begin{bmatrix} (1 - \eta)x \\ (1 + \eta)y - \eta y^3 \end{bmatrix},$$

and Hessian

$$\nabla^2 f(x, y) = \begin{bmatrix} 1 & 0 \\ 0 & 3y^2 - 1 \end{bmatrix}.$$

We can see that $(0, -1)$ and $(0, 1)$ are the local minima, and $(0, 0)$ is the only strict saddle point. Similar to in the previous example, $W^S(0)$ is a low-dimensional subspace.

18.3 The general case

We conclude this lecture with a proof of the main theorem.

Proof of Theorem 18.5. First define the local stable set of x^* as

$$W^S_\epsilon(x^*) = \{x \in B(x^*; \epsilon) : g^t(x) \in B(x^*; \epsilon)\ \forall t\}$$

Intuitively, this describes the subset of $B(x^*; \epsilon)$ that stays in $B(x^*; \epsilon)$ under arbitrarily many gradient maps. This establishes a notion of locality that matters for gradient descent convergence, instead of $B(x^*; \epsilon)$ which has positive measure.

Now we state a simplified version of the stable manifold theorem without proof: For a diffeomorphism $g : \mathbb{R}^n \to \mathbb{R}^n$, if x^* is a fixed point of g, then for all ϵ small enough, $W^S_\epsilon(x^*)$ is a submanifold of dimension equal to the number of eigenvalues of the $Dg(x^*)$ that are $\leqslant 1$. A diffeomorphism, roughly speaking, is a differentiable isomorphism. In fact, since differentiability is assumed for g, we will focus on the isomorphism.

Let x^* be a strict saddle point. Once we have proven the fact that g is a diffeomorphism (using the assumption that $\eta < 1/\beta$), we can apply the stable manifold theorem since x^* is a fixed point of g. Because $\nabla^f(x^*)$ must have an eigenvalue < 0, Dg must have an eigenvalue > 1, so the dimension of $W^S_\epsilon(x^*)$ is less than n and $W^S_\epsilon(x^*)$ has measure 0.

If $g^t(x)$ converges x^*, there must $\exists T$ large enough s.t. $g^T(x) \in W^S_\epsilon(x^*)$. So $W^S(x^*) \subseteq \bigcup_{t\geqslant 0} g^{-t}(W^S_\epsilon(x^*))$. For each t, g^t is in particular an isomorphism (as a composition of isomorphisms), and so it g^{-t}. Therefore $g^{-t}(W^S_\epsilon(x^*))$ has the same cardinality as $W^S_\epsilon(x^*)$ and has measure 0. Because the union is over a countable set, the union also has measure 0, thus its subset $W^S(x^*)$ ends up with measure 0 and we have the desired result.

Finally we show that g is bijective to establish the isomorphism (since it is assumed to be smooth). It is injective because, assuming $g(x) = g(y)$, by smoothness,

$$\|x - y\| = \|g(x) + \eta\nabla f(x) - g(y) - \eta\nabla f(x)\| = \eta\|\nabla f(x) - \nabla f(y)\| \leqslant \eta\beta\|x - y\|$$

Because $\eta\beta < 1$, we must have $\|x - y\| = 0$. To prove that g is surjective, we construct an inverse function

$$h(y) = \operatorname*{argmin}_x \frac{1}{2}\|x - y\|^2 - \eta f(x)$$

a.k.a. the proximal update. For $\eta < 1/\beta$, h is strongly convex, and by the KKT condition, $y = h(y) - \nabla f(h(y)) = g(h(y))$. This completes the proof. ■

19 Alternating minimization and EM

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

Lecture 19
(opens in your browser)

20 Derivative-free optimization, policy gradient, controls

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

Lecture 20
(opens in your browser)

21 Non-convex constraints I

Recall that convex minimization refers to minimizing convex functions over convex constraints. Today we will begin to explore minimizing convex functions with non-convex constraints. It is difficult to analyze the impact of "non-convexity" in general, since that can refer to anything that is not convex, which is a very broad class of problems. So instead, we will focus on solving least squares with sparsity constraints:

$$\min_{\|x\|_0 \leqslant s} \|Ax - y\|_2^2$$

for $y \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times d}$, and $x \in \mathbb{R}^d$ where $d < n$. We will show that in general even this problem is hard to solve but that for a restricted class of problems there is an efficient convex relaxation.

Least squares with sparsity constraints can be applied to solving compressed sensing and sparse linear regression, which are important in a variety of domains. In compressed sensing, A is a measurement model and y are the measurements of some sparse signal x. Compressed sensing is applied to reduce the number of measurements needed for, say, an MRI because by including a sparsity constraint on x we are able to recover the signal x in fewer measurements.

In sparse linear regression, A is the data matrix and y is some outcome variable. The goal of sparse linear regression is to recover a weights x on a sparse set of features that are responsible for the outcome variable. In genetics, A could be the genes of a patient, and y is whether they have a particular disease. Then the goal is to recover a weights x on a sparse set of genes that are predictive of having the disease or not.

When there is no noise in the linear equations, we can simplify the problem to

$$\begin{aligned} &\min \|x\|_0 \\ &Ax = y \end{aligned}$$

21.1 Hardness

Even this simplification is NP-hard, as we will show with a reduction to exact 3-cover, which is NP-complete. Our proof is from [FR13].

Definition 21.1. The *exact cover by 3-sets* problem is given a collection $\{T_i\}$ of 3-element subsets of $[n]$, does there exist an exact cover of $[n]$, a set $z \subseteq [d]$ such that $\cup_{j \in z} T_j = [n]$ and $T_i \cap T_j = \emptyset$ for $j \neq j' \in z$?

Definition 21.2. The support of a vector x is defined as

$$\text{supp}(x) = \{i \mid x_i \neq 0\}.$$

Theorem 21.3. *l_0-minimization for general A and y is NP-hard.*

Proof. Define matrix A as

$$A_{ij} = \begin{cases} 1 & \text{if } i \in T_j \\ 0 & \text{o.w} \end{cases}$$

and y as the all ones vector. Note that from our construction we have $\|Ax\|_0 \leqslant 3\|x\|_0$, since each column of A has 3 non-zeros. If x satisfies $Ax = y$, we thus have $\|x\|_0 \geqslant \frac{\|y\|_0}{3} = \frac{n}{3}$. Let us now run l_0-minimization on A, y and let $\widehat{x}$ be the output. There are two cases

1. If $\|\widehat{x}\|_0 = \frac{n}{3}$, then $y = \text{supp}(\widehat{x})$ is an exact 3-cover.
2. If $\|\widehat{x}\|_0 > \frac{n}{3}$, then no exact 3-cover can exist because it would achieve $\|\widehat{x}\|_0 = \frac{n}{3}$ and hence violate optimality of the solution derived through l_0 minimization.

Thus, since we can solve exact 3-cover through l_0 minimization, l_0 minimization must also be NP-hard. ■

21.2 Convex relaxation

Although l_0-minimization is NP-hard in general, we will prove that for a restricted class of A, we can relax l_0-minimization to l_1-minimization. First, define the set of approximately sparse vectors with support S as those whose l_1 mass is dominated by S. Formally,

Definition 21.4. The set of approximately sparse vectors with support S is

$$C(S) = \{\Delta \in \mathbb{R}^d \mid \|\Delta_{\bar{S}}\|_1 \leqslant \|\Delta_S\|_1\}$$

where $\bar{S} = [d]/S$ and Δ_s is Δ restricted to S,

$$(\Delta_S)_i = \begin{cases} \Delta_i & \text{if } i \in S \\ 0 & \text{o.w} \end{cases}$$

Recall that the nullspace of matrix A is the set $\text{null}(A) = \{\Delta \in \mathbb{R}^d \mid A\Delta = 0\}$. The nullspace is the set of "bad" vectors in our estimation problem. Consider a solution $Ax = y$. If $\Delta \in$ null(A), then $x + \Delta$ is also a solution since $A(x + \Delta) = Ax + A\Delta = Ax = b$. Thus, we focus on matrices whose nullspace only contains zero on the set of sparse vectors that we care about.

Definition 21.5. The matrix A satisfies the restricted nullspace property (RNP) with respect to the support S if $C(S) \cup \text{null}(A) = \{0\}$.

With these definitions in place, we can now state our main theorem.

Theorem 21.6. *Given $A \in \mathbb{R}^{n\times d}$ and $y \in \mathbb{R}^n$ we consider the solution to the l_0-minimization problem $x^* = \text{argmin}_{Ax=y} \|x\|_0$. Assume x^* has support S and let the matrix A satisfy the restricted nullspace property with respect to S. Then given the solutions of the l_1-minimization problem $\widehat{x} = \text{argmin}_{Ax=y} \|x\|_1$ we have $\widehat{x} = x^*$.*

Proof. We first note that by definition both x^* and $\widehat{x}$ satisfy our feasibility constraint $Ax = y$. Letting $\Delta = \widehat{x} - x^*$ be the error vector we have $A\Delta = A\widehat{x} - Ax^* = 0$, which implies that $\Delta \in null(A)$.

Our goal now is to show that $\Delta \in C(S)$ then we would have $\Delta = 0$ from the restricted nullspace property. First, since $\widehat{x}$ is optimal in l_1 it follows that $\|\widehat{x}\|_1 \leqslant \|x^*\|_1$. We then have

$$\begin{aligned}
\|x^*_S\|_1 = \|x^*\|_1 &\geqslant \|\widehat{x}\|_1 \\
&= \|x^* + \Delta\|_1 \\
&= \|x^*_S + \Delta_S\|_1 + \|x^*_{\overline{S}} + \Delta_{\overline{S}}\|_1 && \text{by splitting the } l_1 \text{ norm,} \\
&= \|x^*_S + \Delta_S\|_1 + \|\Delta_{\overline{S}}\|_1 && \text{by the support assumption of } \|x^*\|_1, \\
&\geqslant \|x^*_S\|_1 - \|\Delta_S\|_1 + \|\Delta_{\overline{S}}\|_1.
\end{aligned}$$

Hence $\|\Delta_S\|_1 \geqslant \|\Delta_{\overline{S}}\|_1$, which implies $\Delta \in C(S)$. ■

So far, so good. We have shown that the l_1-relaxation works for certain matrices. A natural question however is what kinds of matrices satisfy the restricted nullspace property. In order to get a handle on this, we will study yet another nice property of matrices, the so called restricted isometry property (RIP). Later, we will then see that specific matrix ensembles satisfy RIP with high probability.

Definition 21.7. A matrix A satisfies the (s, δ)-RIP if for all s-sparse vectors x ($\|x\|_0 \leqslant s$), we have

$$(1 - \delta)\|x\|_2^2 \leqslant \|Ax\|_2^2 \leqslant (1 + \delta)\|x\|_2^2.$$

The intuition is that A acts like an isometry on sparse vectors (a true isometry would have $\delta = 0$). The RIP is useful since it implies that the difference between two s-sparse vectors cannot be mapped to 0. By looking at the singular values of A we can derive the following lemma.

Lemma 21.8. *If A has (s, δ)-RIP, then*

$$\|A_S^\top A_S - I_S\|_2 \leqslant \delta$$

for all subsets S of size s. Where

$$(A_S)_{ij} = \begin{cases} A_{ij} & \text{if } j \in S \\ 0 & \text{o.w.} \end{cases}$$

We now show that the RIP implies the restricted nullspace property.

Theorem 21.9. *If the matrix A has the $(2s, \delta)$-RIP, then it also has the restricted nullspace property for all subsets S of cardinality $|S| \leqslant s$.*

Proof. Let $x \in null(A)$ be arbitrary but not equal to 0. Then we have to show that $x \notin C(S)$ for any S with $|S| \leqslant s$. In particular, let S_0 be the set indices of the s largest coefficients in x. It suffices to show that $\|x_{S_0}\|_1 < \|x_{\overline{S}_0}\|_1$ since it would then hold for any other subset.

We write

$$\overline{S}_0 = \bigcup_{j=1}^{\lceil \frac{d}{s} \rceil - 1} S_j$$

where

- S_1 is the subset of indices corresponding to the s largest entries in $\overline{S}_0$
- S_2 is the subset of indices corresponding to the s largest entries in $\overline{S}_0 \setminus S_1$
- S_3 is the subset of indices corresponding to the s largest entries in $\overline{S}_0 \setminus S_1 \setminus S_2$
- etc...

So we have $x = x_{S_0} + \sum_j x_{S_j}$. We have decomposed x into blocks of size s. This is sometimes called shelling. From RIP, we have

$$\|x_{S_0}\|_2^2 \leqslant \frac{1}{1-\delta} \|Ax_{S_0}\|_2^2.$$

Since $x \in null(A)$ by assumption we have

$$\begin{aligned} A(x_{S_0} + \sum_{j \geqslant 1} x_{S_j}) &= 0 \\ \implies Ax_{S_0} &= -\sum_{j \geqslant 1} Ax_{S_j}. \end{aligned}$$

Hence

$$\begin{aligned}
\|x_{S_0}\|_2^2 &\leqslant \frac{1}{1-\delta}\|Ax_{S_0}\|_2^2 \\
&= \frac{1}{1-\delta}\langle Ax_{S_0}, Ax_{S_0}\rangle \\
&= \frac{1}{1-\delta}\sum_{j\geqslant 1}\langle Ax_{S_0}, Ax_{S_j}\rangle \\
&= \frac{1}{1-\delta}\sum_{j\geqslant 1}\langle Ax_{S_0}, -Ax_{S_j}\rangle \\
&= \frac{1}{1-\delta}\sum_{j\geqslant 1}(\langle Ax_{S_0}, -Ax_{S_j}\rangle - \langle x_{S_0}, x_{S_j}\rangle) && \text{since } \langle x_{S_0}, x_{S_j}\rangle = 0 \\
&= \frac{1}{1-\delta}\sum_{j\geqslant 1}\langle x_{S_0}, (I - A^\top A)x_{S_j}\rangle \\
&\leqslant \frac{1}{1-\delta}\sum_{j\geqslant 1}\|x_{S_0}\|_2\delta\|x_{S_j}\|_2 && \text{from Lemma 21.8.}
\end{aligned}$$

So we have

$$\|x_{S_0}\|_2 \leqslant \frac{\delta}{1-\delta}\sum_{j\geqslant 1}\|x_{S_j}\|_2. \tag{68}$$

By construction, for each $j \geqslant 1$, we have

$$\|x_{s_j}\|_\infty \leqslant \frac{1}{S}\|X_{S_{j-1}}\|_1$$

and hence

$$\|x_{s_j}\|_2 \leqslant \frac{1}{\sqrt{S}}\|X_{S_{j-1}}\|_1.$$

Plugging into Equation 68, we get

$$\begin{aligned}
\|x_{S_0}\|_1 &\leqslant \sqrt{S}\|x_{S_0}\|_2 \\
&\leqslant \frac{\sqrt{S}\delta}{1-\delta}\sum_{j\geqslant 1}\|x_{S_j}\|_2 \\
&\leqslant \frac{\delta}{1-\delta}\sum_{j\geqslant 1}\|x_{S_{j-1}}\|_1 \\
&\leqslant \frac{\delta}{1-\delta}(\|x_{s_0}\|_1 + \sum_{j\geqslant 1}\|x_{S_{j-1}}\|_1)
\end{aligned}$$

which is equivalent to

$$\|x_{s_0}\|_1 \leqslant \frac{\delta}{1-\delta}(\|x_{s_0}\|_1 + \|x_{\bar{s}_0}\|_1).$$

Simple algebra gives us $\|x_{s_0}\|_1 \leqslant \|x_{\bar{s}_0}\|_1$ as long as $\delta < \frac{1}{3}$. ■

Now that we've shown that if A has the RIP then l_1-relaxation will work, we look at a few examples of naturally occurring matrices with this property.

Theorem 21.10. *Let $A \in \mathbb{R}^{n\times d}$ be defined as $a_{ij} \sim \mathcal{N}(0,1)$ iid. Then the matrix $\frac{1}{\sqrt{n}}A$ has (s,δ)-RIP for n at least $\mathcal{O}\left(\frac{1}{\delta^2}s\log\frac{d}{s}\right)$.*

The same holds for sub-Gaussians. We have similar results for more structured matrices such as subsampled Fourier matrices.

Theorem 21.11. *Let $A \in \mathbb{R}^{n\times d}$ be a subsampled Fourier matrix. Then A has (s,δ)-RIP for n at least $\mathcal{O}\left(\frac{1}{\delta^2}s\log^2 s\log d\right)$.*

This result is from [HR15] using work from [RV07, Bou14, CT06]. $\mathcal{O}\left(\frac{1}{\delta^2}s\log d\right)$ is conjectured but open.

There is a lot more work on convex relaxations. For sparsity alone people have studied many variations e.g.

- **Basic pursuit denoising (BPDN)** $\min \|x\|_1$ such that $\|Ax-y\|_2 \leqslant \epsilon$
- **Constrained LASSO** $\min \|Ax-y\|_2^2$ such that $\|x\|_1 \leqslant \lambda$
- **Lagrangian LASSO** $\min \|Ax-y\|_2^2 + \lambda\|x\|_1$

There are also convex relaxations for other constraints. For example $\min rank(X)$ such that $A(X) = Y$ is hard, a simpler problem is to solve the nuclear norm minimization instead: $\min \|X\|_*$ such that $A(X) = Y$. This can be applied to low-rank estimation for images or matrix completion.

Part VI
Higher-order and interior point methods

22 Newton's method

Up until now, we have only considered first order methods to optimize functions. Now, we will utilize second order information to achieve a faster rate of convergence.

As always, our objective is to minimize a function $f\colon \mathbb{R}^n \to \mathbb{R}$. The basic idea of Newton's Method is to set the first-order Taylor expansion of the gradient to zero: $F(x) = \nabla f(x) = 0$. This leads to an iterative update step that will (under certain conditions) lead to a significantly faster convergence rate than gradient descent methods.

To illustrate the point, consider a single variable function $\varphi\colon \mathbb{R} \to \mathbb{R}$. Our goal is to solve the non-linear equation $\varphi(x) = 0$. From Taylor's theorem, we can express the first-order form of $\varphi(x)$ as

$$\varphi(x) = \varphi(x_0) + \varphi'(x) \cdot (x - x_0) + o(|x - x_0|)$$

given $\delta = x - x_0$ we equivalently have that

$$\varphi(x_0 + \delta) = \varphi(x_0) + \varphi'(x) \cdot \delta + o(|\delta|)$$

Disregarding the $o(|\delta|)$ term, we solve (over δ) the following objective:

$$\varphi(x_0) + \varphi'(x_0)\delta = 0$$

Then, $\delta = -\frac{\varphi(x_0)}{\varphi'(x_0)}$, leading to the iteration $x_{t+1} = x_t - \frac{\varphi(x_t)}{\varphi'(x_t)}$.

We can similarly make an argument for a multi variable function $F : \mathbb{R}^d \to \mathbb{R}$. Our goal is to solve $F(x) = 0$. Again, from Taylor's theorem we have that

$$F(x + \Delta) = F(x) + J_F(x)\Delta + o(\|\Delta\|)$$

where J_F is the Jacobian. This gives us $\Delta = -J_F^{-1}(x)F(x)$, and the iteration

$$x_{t+1} = x_t - J_F^{-1}(x_t)F(x_t)$$

Given $f\colon \mathbb{R} \to \mathbb{R}$, Newton's method applies this update to $F(x) = \nabla f(x) = 0$. It uses the update rule

$$x_{t+1} = x_t - \nabla^2 f(x_t)^{-1} \nabla f(x_t)$$

A Newton step minimizes the second order Taylor approximation

$$f(x) \approx f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2}(x - x_t)^\top \nabla^2 f(x_t)(x - x_t)$$

Now, we will show that Newton's method converges to a local minimum, given a starting point that is within a neighborhood of that point.

Theorem 22.1. *Given $f : \mathbb{R}^n \to \mathbb{R}$ and assuming that*

1. *f is twice continuously differentiable*
2. *$\nabla^2 f(x)$ is Lipschitz: $\|\nabla^2 f(x) - \nabla^2 f(x')\| \leqslant \|x - x'\|$*
3. *$\exists x^*$ s.t. $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succeq \alpha I$ and $\|x^0 - x^*\| \leqslant \frac{\alpha}{2}$*

Then, $\|x_{t+1} - x^*\| \leqslant \frac{1}{\alpha}\|x_t - x^*\|^2$

Proof. Given that $\nabla f(x^*) = 0$, we have that

$$\begin{aligned} x_{t+1} - x^* &= x_t - x^* - \nabla^2 f(x_t)^{-1} \nabla f(x_t) \\ &= \nabla^2 f(x_t)^{-1} [\nabla^2 f(x_t)(x_t - x^*) - (\nabla f(x_t) - \nabla f(x^*))] \end{aligned}$$

This implies that

$$\|x_{t+1} - x^*\| \leqslant \|\nabla^2 f(x_t)^{-1}\| \cdot \|\nabla^2 f(x_t)(x_t - x^*) - (\nabla f(x_t) - \nabla f(x^*))\|$$

Claim 22.2. $\|\nabla^2 f(x_t)(x_t - x^*) - (\nabla f(x_t) - \nabla f(x^*))\| \leqslant \frac{1}{2}\|x^t - x^*\|^2$

Proof. Applying the integral remainder form of Taylor's theorem to $\nabla f(x_t)$ we have that

$$\nabla f(x_t) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x_t + \gamma(x^* - x_t)) \cdot (x_t - x^*) d\gamma$$

We therefore have that

$$\begin{aligned} &\|\nabla^2 f(x_t)(x_t - x^*) - (\nabla f(x_t) - \nabla f(x^*))\| \\ &= \| \int_0^1 [\nabla^2 f(x_t) - \nabla^2 f(x_t + \gamma(x^* - x_t))](x_t - x^*) d\gamma \| \\ &\leqslant \int_0^1 \|\nabla^2 f(x_t) - \nabla^2 f(x_t + \gamma(x^* - x_t))\| \cdot \|x_t - x^*\| d\gamma \\ &\leqslant \left(\int_0^1 \gamma d\gamma \right) \|x_t - x^*\|^2 && (\nabla^2 f(x_t) \text{ is Lipschitz}) \\ &= \frac{1}{2}\|x_t - x^*\|^2 \end{aligned}$$

■

Claim 22.3. $\|\nabla^2 f(x_t)^{-1}\| \leqslant \frac{2}{\alpha}$

Proof. By the Wielandt-Hoffman Theorem,

$$\begin{aligned} |\lambda_{min}(\nabla^2 f(x_t)) - \lambda_{min}(\nabla^2 f(x^*))| &\leqslant \|\nabla^2 f(x_t) - \nabla^2 f(x^*)\| \\ &\leqslant \|x_t - x^*\| && (\nabla^2 f(x_t) \text{ is Lipschitz}) \end{aligned}$$

Thus, for $\|x_t - x^*\| \leqslant \frac{\alpha}{2}$ and given that $\nabla^2 f(x^*) \succeq \alpha I$, this implies that $\lambda_{min}(\nabla^2 f(x_t)) \geqslant \frac{\alpha}{2}$. Hence, $\|\nabla^2 f(x_t)^{-1}\| \leqslant \frac{2}{\alpha}$. ■

Putting the two claims together, we have that

$$\|x_{t+1} - x^*\| \leqslant \frac{2}{\alpha} \cdot \frac{1}{2}\|x_t - x^*\|^2 = \frac{1}{\alpha}\|x_t - x^*\|^2$$

■

Note that we did not need convexity in the proof. Given that we are within a neighborhood of the local minimum x^*, then we can achieve ϵ error in just $O(\log\log\frac{1}{\epsilon})$ iterations. (this is called *quadratic convergence*.)

22.1 Damped update

In general, Newton's method can be quite unpredictable. For example, consider the function

$$f(x) = \sqrt{x^2 + 1}$$

essentially a smoothed version of the absolute value $|x|$. Clearly, the function is minimized at $x^* = 0$. Calculating the necessary derivatives for Newton's method, we find

$$\begin{aligned} f'(x) &= \frac{x}{\sqrt{x^2+1}} \\ f''(x) &= (1+x^2)^{-3/2}. \end{aligned}$$

Note that $f(x)$ is strongly convex since its second derivative strictly positive and 1-smooth ($|f'(x)| < 1$). The Newton step for minimizing $f(x)$ is

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x)} = -x_t^3.$$

The behavior of this algorithm depends on the magnitude of x_t. In particular, we have the following three regimes

$$\begin{cases} |x_t| < 1 & \text{Algorithm converges } \textit{cubically} \\ |x_t| = 1 & \text{Algorithm oscillates between } -1 \text{ and } 1 \\ |x_t| > 1 & \text{Algorithm diverges} \end{cases}$$

This example shows that even for strongly convex functions with Lipschitz gradients that Newton's method is only guaranteed to converge locally. To avoid divergence, a popular technique is to use a *damped* step–size:

$$x_{t+1} = x_t - \eta_t \nabla^2 f(x_t)^{-1} \nabla f(x_t)$$

η_t can be chosen by backtracking line search. Usually though $\eta = 1$ is a good first choice since, if you are in a region of convergence, you are guaranteed quadratic convergence.

22.2 Quasi-Newton methods

Let's compare gradient descent and Newton's method side by side.

$$\begin{aligned} x_{t+1} &= x_t - \eta_t \nabla f(x_t) && \text{(Gradient descent)} \\ x_{t+1} &= x_t - \nabla^2 f(x_t)^{-1} \nabla f(x_t) && \text{(Newton's method)} \end{aligned}$$

We can think of gradient descent as a Newton update in which we approximate $\nabla^2 f(x_t)^{-1}$ by a scaled version of the identity. That is, gradient descent is equivalent to Newton's method when $\nabla^2 f(x_t)^{-1} = \eta_t I$ where I is the identity matrix.

Quasi-Newton methods take the analogy a step further by approximating the Hessian by some other matrix. The idea in doing so is to avoid an expensive matrix inversion at each step. What we want is an approximation

$$\widehat{f}_{B_t}(x) \approx f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2}(x - x_t) B_t^{-1} (x - x_t)$$

such that:

1. $\nabla \widehat{f}_{B_t}(x_t) = \nabla f(x_t)$.
   It seems reasonable that our approximation should be the same up to first order.
2. $\nabla \widehat{f}_{B_t}(x_{t-1}) = \nabla f(x_{t-1})$
   This condition states that the gradient should still be correct at the previous iterate.

If the two last gradients are correct, we can expect our Hessian approximation to be reasonable along the direction $x_t - x_{t-1}$. This is called a *secant approximation* which can be written as

$$\nabla \widehat{f}_{B_t}(x_{t+1}) = \nabla f(x_t) - B_t^{-1}(x_{t+1} - x_t)$$

If we let

$$\begin{aligned} s_t &= x_{t+1} - x_t \\ y_t &= \nabla \widehat{f}_{B_t}(x_{t+1}) - \nabla f(x_t) \end{aligned}$$

Then we arrive at the *Secant Equation*

$$s_t = B_t y_t$$

There could be multiple B_t that satisfy this condition. We can enforce other constraints to help narrow down on a particular choice. Some popular requirements are requiring B_t to be positive definite, making sure B_t is as close to B_{t-1} as possible for some appropriate metric, or requiring B_t to be a low–rank update of previous iterates where the update can be done via the Sherman–Morrison formula. One of the most successful implementations of this is called BFGS named after Broyden, Fletcher, Goldfarb, Shanno and its limited–memory counterpart, L–BFGS.

23 Experimenting with second-order methods

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

Lecture 24
(opens in your browser)

24 Enter interior point methods

In the last lecture, we discussed Newton's method. Although it enjoys a fast local convergence guarantee, global convergence of Newton's method is not guaranteed. In this lecture, we'll introduce interior point methods, which can be thought of as an extension of Newton's method to ensure global convergence. We will first introduce the main idea of *barrier methods* at great generality, before we specialize our analysis to linear programming.

24.1 Barrier methods

Barrier methods replace inequality constraints with a so-called *barrier* function that is added to objective function of the optimization problem. Consider the following optimization problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in \Omega, \\ & g_j(x) \leqslant 0\,, \qquad j = 1, 2, \cdots, r\,, \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g_j : \mathbb{R}^n \to \mathbb{R}$ are given functions. The function f is continuous, and Ω is a closed set. For the rest of the lecture, we assume convex g_j and $\Omega = \mathbb{R}^n$. And we denote x^* as the optimal solution of the problem.

Definition 24.1 (Interior of the constraint region)**.** The interior (relative to Ω) of the constraint region is defined as $S = \{x \in \Omega : g_j(x) < 0,\ j = 1, 2, \cdots, r\}$.

Assuming nonempty and convex S, we define a so-called barrier function $B(x)$ defined on S, such that $B(x)$ is continuous the function blows up as we approach the boundary of the constraint region. More formally, $\lim_{g_j(x) \to 0_-} B(x) = \infty$. Two most common examples are logarithmic barrier function and inverse barrier function:

$$\text{Logarithmic:} \qquad B(x) = -\sum_{j=1}^{r} \ln\{-g_j(x)\} \tag{69}$$

$$\text{Inverse:} \qquad B(x) = -\sum_{j=1}^{r} \frac{1}{g_j(x)}. \tag{70}$$

Both of them are convex if all $g_j(x)$ are convex.

Given a barrier function $B(x)$, define a new cost function $f_\epsilon(x) = f(x) + \epsilon B(x)$, where ϵ is a positive real number. Then we can eliminate the inequality constraints in the original problem and obtain the following problem:

$$\begin{aligned} \min_x \quad & f_\epsilon(x) \\ \text{s.t.} \quad & x \in \Omega \end{aligned} \tag{71}$$

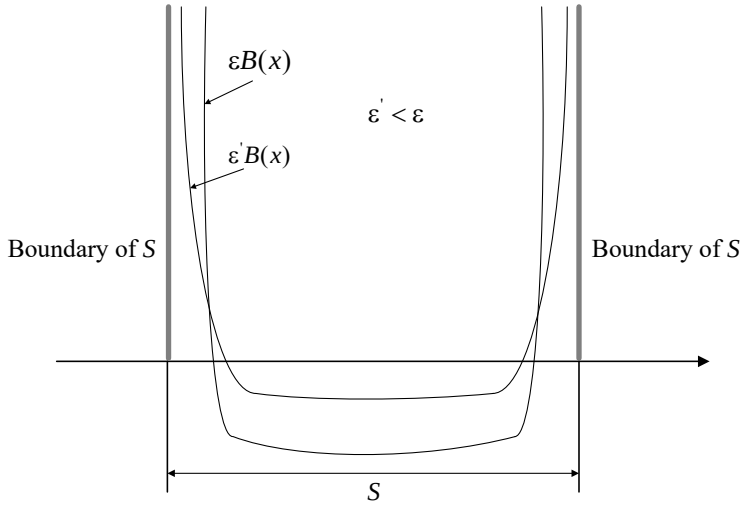


Figure 16: Form of a barrier term

The form of the barrier term $\epsilon B(x)$ is illustrated in Figure 16.

The barrier method is defined by introducing a sequence $\{\epsilon_t\}$ such that $0 < \epsilon_{t+1} < \epsilon_t$ for $t = 0, 1, 2, ...$ and $\epsilon_t \to 0$. Then we find a sequence $\{x_t\}$ such that $x_t \in \arg\min_{x \in S} f_{\epsilon_t}(x)$. Note that the barrier term $\epsilon_t B(x)$ goes to zero for all interior points $x \in S$ as $\epsilon_t \to 0$, allowing x_t to get increasingly closer to the boundary. Therefore, intuitively, x_t should approach x^* no matter x^* is in the interior or on the boundary of S. Its convergence is formalized in the following proposition.

Proposition 24.2. *Every limit point of a sequence $\{x_t\}$ generated by a barrier method is a global minimum of the original constrained problem.*

Proof. See Proposition 5.1.1 of [Ber16]. ■

The previous proposition shows that the global optima of our barrier problems converge to the global constrained optimum. But how do we solve this sequence of optimization problems. The key intuition is this. An initial interior point can often be obtained easily for some large enough ϵ_0. Then in each iteration, we can use x_t as an initialization to find x_{t+1} by Newton's method. If ϵ_t is close to ϵ_{t+1}, we expect that x_t is also close to x_{t+1}. Therefore, we have reason to hope that x_t is in the local convergence region for Newton's method. In this manner we can extend the local convergence guarantee of Newton to a more global property.

24.2 Linear programming

After sketching the basic idea in full generality, we will now tailor the logarithmic barrier method to the linear programming (LP) problem defined as follows:

$$\text{LP}: \quad \min_{x} \quad c^\top x \quad \text{s.t.} \quad Ax \geqslant b \tag{72}$$

Here, $A \in \mathbb{R}^{m \times n}$ with $m \geqslant n$ and $\text{rank}(A) = n$. Denote x^* as the optimal point.

First, we write out the augmented cost function by the logarithmic barrier method, i.e.,

$$f_\epsilon(x) = c^\top x - \epsilon \sum_{j=1}^{m} \ln\left(A_j^\top x - b\right). \tag{73}$$

where $A_j^\top$ is the j-th row of A. Define $x_\epsilon^* = \text{argmin}_x f_\epsilon(x)$.

Fact 24.3. *The optimal point x_ϵ^* exists and is unique for any $\epsilon > 0$.*

Proof. We can easily check that $f_\epsilon(x)$ is convex (as a sum of two convex functions). Therefore, the minimizer x_ϵ^* must exist and is unique.

To show the convexity of f_ϵ, we can check the second-order derivative, which is positive definite as shown in (75) later. ■

24.2.1 Central path

The central path of the LP problem in 72 is depicted by the set of $\{x_\epsilon^* | \epsilon > 0\}$, as shown in Figure 17.

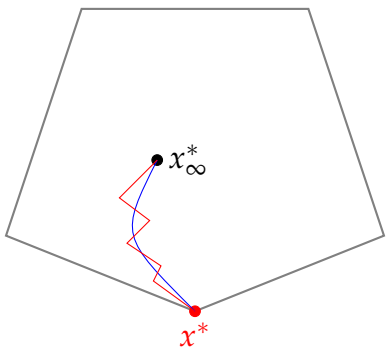


Figure 17: The central path

Our goal is to design an algorithm that will approximately follow the central path. Assume that we already have a "good enough" initial point, then at every step, we apply one step of Newton's method. To guarantee that the algorithm converges, we need to answer the following two questions:

- Under what conditions does the single-step Newton method work?
- How should we update ϵ?

24.2.2 Newton decrement

To apply Newton's method, first we need to find out the first-order and second-order derivatives of f_ϵ. Note that

$$\nabla f_\epsilon(x) = c - \epsilon \sum_{j=1}^{m} \frac{A_j}{A_j^\top x - b} \triangleq c - \epsilon A^\top S^{-1} \mathbb{1} \tag{74}$$

$$\nabla^2 f_\epsilon(x) = \epsilon A^\top S^{-2} A = \epsilon \sum_{j=1}^{m} \frac{A_j A_j^\top}{s_j^2} \tag{75}$$

where $\mathbb{1} = [1, 1, \cdots, 1]^\top \in \mathbb{R}^{m \times 1}$, and $S = \text{diag}\{s_1, \ldots, s_m\}$ is the diagonal matrix of slack quantities $s_j = A_j^\top x - b$.

Recall the Newton update

$$\bar{x} = x - [\nabla^2 f_\epsilon(x)]^{-1} \nabla f_\epsilon(x) = x - [\epsilon A^\top S^{-2} A]^{-1} \left(c - \epsilon A^\top S^{-1} \mathbb{1} \right) . \tag{76}$$

Recall that Newton's method finds the solution by making the first-order condition zero. To measure how much the Newton update will decrease the first-order approximation, we introduce the concept of Newton decrement.

Define the *Newton decrement* $q(x, \epsilon)$ as

$$q^2(x, \epsilon) = \nabla f_\epsilon(x)^\top [\nabla^2 f_\epsilon(x)]^{-1} \nabla f_\epsilon(x) . \tag{77}$$

Equivalently,

$$\begin{aligned} q(x, \epsilon) &= \left\| [\nabla^2 f_\epsilon(x)]^{-1/2} \nabla f_\epsilon(x) \right\|_2 \\ &= \left\| \nabla^2 f_\epsilon(x)^{-1} \nabla f_\epsilon \right\|_{\nabla^2 f_\epsilon(x)} , \end{aligned}$$

where $\|x\|_H = \sqrt{x^\top H x}$. The last identity reveals that we can think of the Newton decrement as the magnitude of the Newton step measured in the *local norm* of the Hessian.

Note that the Newton decrement also relates to the difference between $f_\epsilon(x)$ and the minimum of its second-order approximation:

$$\begin{aligned} & f_\epsilon(x) - \min_{\bar{x}} \left(f_\epsilon(x) + \nabla f_\epsilon(x)^\top (\bar{x} - x) + (\bar{x} - x)^\top \nabla^2 f_\epsilon(x) (\bar{x} - x) \right) \\ = \; & f_\epsilon(x) - \left(f_\epsilon(x) - \frac{1}{2} \nabla f_\epsilon(x)^\top [\nabla^2 f_\epsilon(x)]^{-1} \nabla f_\epsilon(x) \right) \\ = \; & \frac{1}{2} \nabla f_\epsilon(x)^\top [\nabla^2 f_\epsilon(x)]^{-1} \nabla f_\epsilon(x) \triangleq \frac{1}{2} q^2(x, \epsilon). \end{aligned} \tag{78}$$

We'll use the Newton decrement to find out the conditions for the convergence guarantee of the algorithm.

24.2.3 An update rule and its convergence guarantee

We'll now come up with an update rule that can guarantee convergence if some initial conditions are satisfied. To develop the update rule, we first introduce the following propositions.

Proposition 24.4. *Assume $Ax > b$ and $q(x, \epsilon) < 1$, then we have*

$$c^\top x - c^\top x^* \leqslant 2\epsilon n. \tag{79}$$

In particular, if we maintain that x_t is interior point satisfying $Ax_t > b$, and $q(x_t, \epsilon_t) < 1$, then $c^\top x_t$ converges to $c^\top x^*$ as ϵ_t goes to 0, i.e., x_t converges to global optimum. However, the condition $q(x_t, \epsilon_t) < 1$ is not trivial.

Proposition 24.5. *If $Ax > b$, and $q(x, \epsilon) < 1$, then the pure Newton iterate step $\bar{x}$ satisfies,*

$$q(\bar{x}, \epsilon) \leqslant q(x, \epsilon)^2 \tag{80}$$

It ensures that $q(\bar{x}, \epsilon) < 1$ given $q(x, \epsilon) < 1$ and x is interior point. But we also want that $q(\bar{x}, \bar{\epsilon}) < 1$ for some $\bar{\epsilon} < \epsilon$.

Proposition 24.6. *Assume $q(x, \epsilon) \leqslant \frac{1}{2}$, interior point $Ax > b$, put*

$$\bar{\epsilon} = \left(1 - \frac{1}{6\sqrt{n}}\right)\epsilon, \tag{81}$$

then we have

$$q(\bar{x}, \bar{\epsilon}) \leqslant \frac{1}{2} \tag{82}$$

These propositions suggest the following update rule,

$$x_{t+1} = x_t - \nabla^2 f_{\epsilon_t}(x)^{-1} \nabla f_{\epsilon_t}(x_t) \tag{83}$$

$$\epsilon_t = \left(1 - \frac{1}{6\sqrt{n}}\right)\epsilon \tag{84}$$

Theorem 24.7. *Suppose (x_0, ϵ_0) satisfies $Ax_0 > b$ and $q(x_0, \epsilon_0) \leqslant \frac{1}{2}$, then the algorithm converges in $\mathcal{O}(\sqrt{n}\log(n/\eta))$ iterations to η error, i.e., we have $c^\top x_t \leqslant c^\top x^* + \eta$ after $\mathcal{O}(\sqrt{n}\log(n/\eta))$ iterations.*

Proof. As Newton step maintains x_{t+1} in the interior, by using the three propositions above, we have

$$\begin{aligned} c^\top x_t &\leqslant c^\top x^* + 2\epsilon_t n \\ &= c^\top x^* + 2\left(1 - \frac{1}{6\sqrt{n}}\right)^t \epsilon_0 \\ &\leqslant c^\top x^* + 2\exp\left(-\frac{t}{6\sqrt{n}}\right)\epsilon_0 \end{aligned} \tag{85}$$

Therefore, to have a error of η, $t \geqslant \frac{6\sqrt{n}}{\epsilon_0}\log\frac{2n}{\eta}$. We can then conclude that the algorithm converges in $\mathcal{O}(\sqrt{n}\log(n/\eta))$ iterations to η error. ■

The algorithm stated above is the so-called short-step method. Although theoretical convergence rate is guaranteed, the combination of small decrease in ϵ and a single Newton step is slow in practice. Instead, a more practical method is the so-called long-step method, where ϵ is reduced in faster rate and several Newton steps are taken per iteration.

25 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 9: John Miller, Vlad Feinburg

Lecture 12: Erin Grant

Lecture 14: Feynman Liang

Lecture 15: Lydia Liu, Tijana Zrnic

Lecture 17: Adam Villaflor

Lecture 18: Yu Sun

Lecture 21: Smitha Milli, Karl Krauth

Lecture 22: Mehran Mirramezani and Serena Yuan

Lecture 23: Soroush Nasiriany, Armin Askari

lecture 25: Chen Tang, Liting Sun, Xinlei Pan

26 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.

[BE02] Olivier Bousquet and André Elisseeff. Stability and generalization. *JMLR*, 2:499–526, 2002.

[Ber16] D.P. Bertsekas. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 2016.

[Bou14] Jean Bourgain. *An Improved Estimate in the Restricted Isometry Problem*, pages 65–70. Springer International Publishing, Cham, 2014.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical computer science*, 22(3):317–330, 1983.

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

[CT06] Emmanuel J. Candès and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Information Theory*, 52(12):5406–5425, 2006.

[DGN14] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[DJL+17] Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pages 1067–1077, 2017.

[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.

[FR13] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. 2013.

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

[GHJY15] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points - online stochastic gradient for tensor decomposition. *CoRR*, abs/1503.02101, 2015.

[HR15] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. *CoRR*, abs/1507.01768, 2015.

[HRS15] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *CoRR*, abs/1509.01240, 2015.

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

[LPP+17] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. First-order methods almost always avoid saddle points. *CoRR*, abs/1710.07406, 2017.

[LSJR16] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

[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.

[RV07] Mark Rudelson and Roman Vershynin. On sparse reconstruction from fourier and gaussian measurements. 2007.

[SSSSS10] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11:2635–2670, 2010.

[SSZ13] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

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

[TVW+17] Stephen Tu, Shivaram Venkataraman, Ashia C Wilson, Alex Gittens, Michael I Jordan, and Benjamin Recht. Breaking locality accelerates block gauss-seidel. In *Proc. 34th ICML*, 2017.