A Gentle Introduction to Convex Optimization Methods

Gene Li
gene@ttic.edu

Naren Manoj
nsm@ttic.edu

February 2024

Contents

1 Introduction

[gl: Before starting course material, introduce ourselves.]

This course provides an introduction to the theory and practice of continuous optimization, with a focus on first-order (gradient-based) methods. The aim of the course is not to provide an extremely rigorous treatment of convex optimization, but instead introduce you to these concepts so that you can get a sense of how to apply it to problems that you are interested in. For the interested reader, there are a lot of excellent books and lecture notes.

[gl: Boyd and Vandenberghe book; Bubeck notes.]

The course is roughly split into two parts. Both parts will have a mix of both theory and programming.

1. In Part 1, we will cover the standard theoretical analyses of gradient-based methods for different problem settings, and implement the algorithms on real problems, mostly motivated by applications in machine learning.
2. In Part 2, we will focus on "advanced topics" which build on the material covered in Part 1. Topics include stochastic optimization, acceleration, and derivative-free optimization.

1.1 Why study optimization?

In the most general form, we consider solving the optimization problem:

$$\begin{aligned} \text{minimize}_x \quad & f(x) \\ \text{subject to} \quad & x \in S \subseteq \mathbb{R}^d. \end{aligned}$$

- Here, x denotes the *optimization variable*.
- We also have a constraint $x \in S$, and we call S the *constraint set*.
- Lastly, we call $f(\cdot)$ the *objective function*.

Different fields might use different names for the variables. For example in machine learning it is common to minimize a function L over a variable w or θ.

Many real-world problems in modern engineering, finance, and data science can be framed as optimization problems. Thus, it is useful to develop a common framework for studying such problems. Let's consider some examples. [gl: Here we can ask students about possible constraints that they might want to enforce for these examples.]

- Portfolio optimization: Suppose we have some money, and we would like to invest it into d different stocks. The variable x_i for $i \in [d]$ represents the investment into the ith stock, so $x \in \mathbb{R}^d$ represents the total portfolio allocation. However, we might have some constraints on our portfolio allocation. For example, we do not have infinite money, which corresponds to some bound on x, for example $\|x\|_1 \leq C$. We can also have a constraint that the investments are diversified, and this can be captured by assuming that $x_i > c > 0$. The objective function f can capture some desired cost or utility. It can be the total return for the portfolio, or it can be some measure of "risk" which represents how much downside there is to the investment strategy, or maybe some weighted combination.

- Device design: Imagine we have a bunch of devices we want to size in an electronic circuit. The optimization variable is the length and width of each chip, and f can be some measure of power consumption. However, we often have constraints on the size of the circuit components as determined by the manufacturer.
- Operations research: Suppose we are a major shipping company like UPS. Every day we have to determine how to move packages around the country. Suppose we have a fleet of trucks that we want to send from City A to City B, with several intermediate cities along the way. We can decide how many trucks we want to send along each route. But there are constraints: we cannot send more trucks on a route than what the capacity of the road allows (otherwise they end up in a traffic jam). What is the best way to allocate our fleet of vehicles? It turns out that this is a very old problem.

1.2 Our First Algorithm: Grid Search

Let's consider a simple example. Suppose $f : \mathbb{R} \to \mathbb{R}$ is a continuous function in 1D, and suppose $S = [0, 1]$. Our goal is to find the global minimizer $x^\star$. [gl: Draw a picture here.] Of course, since $x^\star$ is a real number, we might not be able to exactly find $x^\star$. However, we will usually be content with finding a near-optimal point $\widehat{x}$: that is, $\widehat{x}$ satisfies the property that $f(\widehat{x}) \leq f(x^\star) + \varepsilon$ for some very small value of $\varepsilon > 0$.

We will make the following assumption on f:

Assumption 1. *The function f is L-Lipschitz, i.e., for any $x, y \in [0, 1]$ we have $|f(x) - f(y)| \leq L|x - y|$.*

Roughly speaking Assumption 2 states that small changes in the input only change the function very slightly, up to a factor of L. This means that the slope cannot be too large for differentiable f. Also functions which are nondifferentiable can satisfy the Lipschitz assumption, for example consider $f(x) = |x - x^\star|$.

There is a very simple algorithm for optimization in 1D described in Algorithm 1.

Algorithm 1 Grid Search

1: **Require:** accuracy parameter ε, function f.
2: Construct the partition $S' = \{0, \varepsilon/L, 2\varepsilon/L, \cdots, 1\}$
3: For each $x' \in S'$, evaluate $f(x')$.
4: **Return** $\widehat{x} := \operatorname{argmin}_{x' \in S'} f(x')$.

Theorem 1. *Under Assumption 2, Algorithm 1 uses $O(L/\varepsilon)$ function evaluations and returns an ε-optimal point.*

Proof of Theorem 1. Let $\widetilde{x}^\star \in S'$ denote the point that $x^\star$ is closest to in S'. By Assumption 2, we know that

$$|f(\widetilde{x}^\star) - f(x^\star)| \leq L \cdot |\widetilde{x}^\star - x^\star| \leq L \cdot \frac{\varepsilon}{L} = \varepsilon.$$

Therefore we know that $f(\widetilde{x}^\star) \leq f(x^\star) + L\varepsilon$. Furthermore, by optimality of $\widehat{x}$ we have

$$f(\widehat{x}) \leq f(\widetilde{x}^\star) \leq f(x^\star) + \varepsilon,$$

completing the proof of Theorem 1. □

It's also clear that if we don't know the analytic form of f then we need some kind of assumption like Assumption 2. Consider the "needle-in-the-haystack" function. $f(x) = 1 - \mathbb{1}\{x = x^\star\}$. If we are just blinding evaluating points, there is no way we can actually find the minimum.

Grid Search in Higher Dimensions. It is straightforward to extend grid search to work in higher dimensions. We have an analogue of Assumption 2 for multivariate functions.

Assumption 2. *The function f is L-Lipschitz, i.e., for any $x, y \in [0,1]$ we have $|f(x) - f(y)| \le L\|x - y\|_2$.*

Let's say $S = [0,1]^d$. We can define the box partition:

$$S' \coloneqq \{0, \varepsilon/(L\sqrt{d}), 2\varepsilon/(L\sqrt{d}), \cdots, 1\}^d$$

Following the same analysis we know that the point closest to the optimum $\widetilde{x}^\star$ satisfies

$$f(\widetilde{x}^\star) \le f(x^\star) + L \cdot \|\widetilde{x}^\star - x^\star\|_2 \le L\sqrt{d} \cdot \|\widetilde{x}^\star - x^\star\|_\infty \le \varepsilon.$$

The second inequality uses the fact that for any d-dimensional vector v, we have $\|v\|_2 \le \sqrt{d}\|v\|_\infty$. So therefore we have that $f(\widehat{x}) \le f(\widetilde{x}^\star)$, so $\widehat{x}$ is ε-suboptimal.

Why is grid search bad? Observe that in higher dimensions, the grid search algorithm requires us to compute function evaluations at $\left(\frac{L\sqrt{d}}{\varepsilon}\right)^d$ points. Unfortunately, for many optimization problems such as the ones we have already mentioned before, d can be quite large! An exponential dependence on d is undesirable. In the worst case this dependence cannot be removed. However, the rest of this course will be concerned with further restrictions on the objective function f and the constraint set S that enable us to reduce such dependence on d.

Gradient Methods. Instead of grid search, we will mostly study gradient methods for optimization. Gradient descent is a popular technique for minimizing functions. At a high level, it is a form of *local search*, where we maintain some "guess" of the best point $x^\star$, and iteratively refine our guess. To see how to do this, consider minimizing $f : \mathbb{R} \to \mathbb{R}$. By Taylor's theorem, we know that for any δ sufficiently small, we have

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

If δ is very small, then the term $\frac{1}{2}\delta^2 f''(x)$ is negligible. This tells us that if we start at the point x and move to the nearby point $x - \delta$, then we would expect to decrease our function value by the quantity $\delta \cdot f'(x)$. In this course, we will generalize this idea to multivariate functions. In fact, we will show that algorithms based on this idea allow us to solve the optimization problem.

2 Convexity

In this section, we will introduce the concept of convexity (both for functions and sets). [gl: See this https://tjdiamandis.github.io/convex-short-course/notes/01-convex-sets-functions.pdf]

2.1 Convex Sets

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

Some examples.

1. Linear spaces $\{x \in \mathbb{R}^d : Ax = 0\}$ and halfspaces $\{x \in \mathbb{R}^d : \langle a, x\rangle \geq 0\}$.
2. Affine transformation of convex sets. If K is convex, then so is $\{Ax + b : x \in K\}$.
3. Norm balls $\{x \in \mathbb{R}^d : \|x\|_p \leq B\}$ for any $p \geq 1$.
4. Intersections of convex sets.
5. Positive Semidefinite Matrices: $S_+^d = \{A \in \mathbb{R}^{d\times d} : A \succeq 0\}$. By $A \succeq 0$ we mean that $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^d$. To see this let A be the set of all symmetric matrices.
6. Polyhedra: $\{x \in \mathbb{R}^d : Ax = b, Cx \leq d\}$.

2.2 Convex Functions

Definition 2. *A function $f : K \to \mathbb{R}$ is* convex *if for any $x, y \in K$ and scalars $\gamma \in [0,1]$, we have*

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

[gl: Draw a picture.]

There is a relationship between convex functions and sets.

Definition 3. *The* epigraph *of a function $f : K \to \mathbb{R}$ is defined as*

$$\text{epi}(f) = \{(x,t) : f(x) \leq t\}.$$

Proposition 1. *A function is convex if and only if its epigraph is convex.*

[gl: Draw a picture.]

Some examples.

- Affine functions $f(x) = ax + b$
- Exponential: $f(x) = e^{ax}$.
- Powers: $f(x) = |x|^p$ for any $p \geq 1$.
- Negative log $f(x) = -\log x$ and negative entropy $f(x) = -x \log x$.
- Norms: $\|x\|_p$ for any $p \geq 1$.
- max: $f(x) = \max\{x_1, \cdots, x_d\}$.
- Quadratic functions: $f(x) = \langle x, Ax\rangle + \langle b, x\rangle + c$ for any $A \succeq 0$.

Convex functions have the following nice property.

Proposition 2. *For any convex f, all local minima are also global minima.*

Proof. Let $x \in K$ be any local minima of f. We know that every point close to f must have larger function value. Now pick any $y \in K$. We can always pick γ small enough so that

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

The first inequality follows by the local minima property, and the second inequality follows by convexity. Therefore, rearranging this equation we get that $f(x) \leq f(y)$ for all $y \in K$, meaning that x is actually a global minima. □

2.3 First Order Characterization

Next we examine the first order condition that you might have seen in calculus class for univariate functions. For multivariate $f : K \to \mathbb{R}$ we can define the gradient at point $x \in K$ as the vector of partial derivatives:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_d} \right) \in \mathbb{R}^d.$$

At this point it might be good to review some basic material on matrix calculus. [gl: Go through parts of the Stanford 229 notes https://cs229.stanford.edu/section/cs229-linalg.pdf and derive the gradients for different functions.]

- Gradient of a linear function. $f(x) = \langle b, x \rangle$
- Gradient of a quadratic function. $f(x) = \langle x, Ax \rangle$.
- Gradient of norms.

We can relate linear functions of the gradients to 1D derivatives. For any $f : K \to \mathbb{R}$ that is differentiable, and any $x, y \in K$ we have

$$\langle \nabla f(x), y \rangle = \left. \frac{\partial f(x + \eta y)}{\partial \eta} \right|_{\eta = 0}.$$

For convex functions we have the following fact.

Proposition 3. *Assume $f : K \to \mathbb{R}$ is differentiable. Then f is convex if for all $x, y \in K$ we have*

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

Proof. First let us assume that f is convex. Then we have for any $x, y \in K$

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

Rearranging this implies that

$$\begin{aligned} f(y) &\geq \frac{f(\gamma y + (1-\gamma)x) - (1-\gamma)f(x)}{\gamma} \\ &= f(x) + \frac{f(x + \gamma(y-x)) - f(x)}{\gamma} \\ &\to f(x) + \langle \nabla f(x), y - x \rangle \quad \text{as } \gamma \to 0. \end{aligned}$$

Now we show the other direction. Let's fix two points x, y as well as γ. Define $z = \gamma x + (1-\gamma)y$ we can get that

$$f(x) \geq f(z) + \langle \nabla f(z), x - z \rangle, \quad \text{and} \quad f(y) \geq f(z) + \langle \nabla f(z), y - z \rangle.$$

If we add these two inequalities after scaling by γ and $1-\gamma$ respectively we get that

$$\gamma f(x) + (1-\gamma) f(y) \geq f(z) + \langle \nabla f(z), \gamma x + (1-\gamma)y - z \rangle = f(\gamma x + (1-\gamma)y).$$

This establishes convexity. □

A way to think about this is that for convex functions, the gradient of f at a given point x gives us a linear lower bound on the function at any other point. [gl: Include an illustration.] As a corollary, if K is taken to be the entire space $\mathbb{R}^d$ we know that x is a global minimum if and only if $\nabla f(x) = 0$.

3 Gradient Descent for Quadratics

Now we will introduce the gradient descent algorithm and analyze it for quadratic objective functions. Quadratic objective functions are already a very interesting class of functions to optimize, and they have many applications. [gl: Look at these notes https://yuxinchen2020.github.io/large_scale_optimization/lectures/grad_descent_unconstrained.pdf.]

3.1 Gradient Descent Algorithm

Let us first think about how we want to build an algorithm to solve optimization. For now, we will consider unconstrained optimization. Imagine we have some function f which we want to minimize. The basic idea of an iterative algorithm is that we start with some *initial point* x_0 and we construct a sequence of points $x_1, x_2, \cdots$ which satisfies the property that

$$f(x_{t+1}) < f(x_t), \quad t = 0, 1, \cdots$$

Thus, eventually we will reach the minimizer $x^\star \in \operatorname{argmin}_x f(x)$. Concretely, to construct such a sequence of points, in every iteration, we will define the next iterate by searching along a direction, as $x_{t+1} = x_t + \eta_t d_t$, where d_t is a *descent direction* at x_t and η_t is a step size.

The method of **gradient descent** is an example of this general template. It can be traced back to Augustin Louis Cauchy in 1847. The idea is to pick the descent direction to be the negative of the gradient of f:

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

In other words, this is the direction of *steepest descent*, since we know that:

$$\operatorname*{argmin}_{d:\|d\|_2 \leq 1} \frac{\partial f(x+\eta d)}{\partial \eta}\bigg|_{\eta=0} = \operatorname*{argmin}_{d:\|d\|_2 \leq 1} \langle \nabla f(x), d \rangle = -\frac{\nabla f(x)}{\|\nabla f(x)\|_2}.$$

The last equality is due to Cauchy-Schwarz inequality. In words, this says that the direction of descent which results in the greatest decrease in function value f is exactly $-\frac{\nabla f(x)}{\|\nabla f(x)\|_2}$.

Now that we have established the gradient descent methods, there are a few questions we would like to understand about it. The main question is one about iteration complexity: how many steps of gradient descent do we need to convergence to some fixed suboptimality? We did this kind of analysis already for the grid search algorithm.

3.2 Quadratic Minimization

Let's begin by optimizing quadratic objective functions:

$$\min_{x \in \mathbb{R}^d} f(x) \coloneqq \frac{1}{2}(x - x^\star)^\top Q(x - x^\star).$$

Here, we have $Q \succ 0$ is some $n \times n$ matrix. It is clear that the optimal value of this function is 0, and is obtained at the point $x = x^\star$.

We have already calculated the gradient of the function:

$$\nabla f(x) = Q(x - x^\star).$$

Convergence for constant stepsizes. We show this result for gradient descent with constant step size. We will use $\lambda_i(Q)$ to denote the ith largest eigenvalue of Q.

Theorem 2. *Suppose we pick $\eta_t = \eta = \frac{2}{\lambda_1(Q)+\lambda_n(Q)}$, then*

$$\|x_t - x^\star\|_2 \leq \left(\frac{\lambda_1(Q) - \lambda_n(Q)}{\lambda_1(Q) + \lambda_n(Q)}\right)^t \|x_0 - x^\star\|_2.$$

Observe that $\lambda_1, \cdots \lambda_n$ are all positive and in strictly decreasing order, so the RHS is further upper bounded by $(\lambda_1/\lambda_n)^t \|x_0 - x^\star\|_2$. We call λ_1/λ_n the *condition number*, and it plays an important role in optimization! This convergence rate is extremely fast; and it is often called linear convergence or geometric convergence. The name "linear" comes from the fact that if you plot the error on a log-linear plot vs. iteration count, it lies below a line.

Let's prove this result.

Proof. By the GD update rule we have

$$x_{t+1} - x^\star = x_t - x^\star - \eta_t \nabla f(x_t) = (I - \eta_t Q)(x_t - x^\star).$$

Taking norms of both sides we get that

$$\|x_{t+1} - x^\star\|_2 \leq \|I - \eta_t Q\| \cdot \|x_t - x^\star\|_2.$$

Now observe that

$$\|I - \eta_t Q\| = \max(|1 - \eta_t \lambda_1(Q)|, |1 - \eta_t \lambda_n(Q)|).$$

Recall that under our choice of η_t we get that

$$\|I - \eta_t Q\| = 1 - \frac{2\lambda_n(Q)}{\lambda_1(Q) + \lambda_n(Q)} = \frac{\lambda_1(Q) - \lambda_n(Q)}{\lambda_1(Q) + \lambda_n(Q)}.$$

Plugging this back into our bound and applying recursion completes the proof. □

We also have the following corollary.

Corollary 1. *Under the choice $\eta_t = \eta = \frac{2}{\lambda_1(Q)+\lambda_n(Q)}$, gradient descent achieves the suboptimality guarantee*

$$f(x_t) - f(x^\star) \leq \frac{\lambda_1(Q)}{2}\left(\frac{\lambda_1(Q) - \lambda_n(Q)}{\lambda_1(Q) + \lambda_n(Q)}\right)^{2t} \|x_0 - x^\star\|_2.$$

Proof. This follows by Cauchy Schwarz. □

Convergence for exact line search. A downside of the previous result is that we need to exactly know what $\lambda_1(Q)$ and $\lambda_n(Q)$ are for the matrix Q. In some cases, this might require some preliminary experimentation. Another strategy is the *exact line search rule*:

$$\eta_t = \operatorname*{argmin}_{\eta \geq 0} f(x_t - \eta \nabla f(x_t)).$$

That is, we will search in the direction $\nabla f(x_t)$ to find the step size that results in the largest function decrease. While exactly computing the step size is not possible, observe that this is actually a 1D optimization problem, so it can be reasonably solved with something like binary search!

Here is a convergence rate guarantee for exact line search.

Theorem 3. *Suppose we pick $\eta_t = \operatorname{argmin}_{\eta \geq 0} f(x_t - \eta \nabla f(x_t))$, then we have*

$$f(x_t) - f(x^\star) \leq \left(\frac{\lambda_1(Q) - \lambda_n(Q)}{\lambda_1(Q) + \lambda_n(Q)} \right)^{2t} (f(x_0) - f(x^\star)).$$

This theorem has a very similar convergence rate to what we proved before.

Proof. First we show a closed form for the exact line search step size. Let $g_t \coloneqq Q(x_t - x^\star)$. Then we can verify that

$$\eta_t = \operatorname*{argmin}_{\eta \geq 0} \frac{1}{2}(x_t - x^\star - \eta g_t)^\top Q (x_t - x^\star - \eta g_t) = \frac{\langle g_t, g_t \rangle}{\langle g_t, Q g_t \rangle}.$$

To solve the minimization problem, we took derivatives of the objective and set it to zero. Therefore we have

$$\begin{aligned}
f(x_{t+1}) &= \frac{1}{2}(x_t - x^\star - \eta_t g_t)^\top Q (x_t - x^\star - \eta_t g_t) \\
&= \frac{1}{2}(x_t - x^\star)^\top Q (x_t - x^\star) - \eta_t \|g_t\|_2^2 + \frac{\eta_t^2}{2} g_t^\top Q g_t \\
&= \frac{1}{2}(x_t - x^\star)^\top Q (x_t - x^\star) - \frac{\|g_t\|_2^4}{2 g_t^\top Q g_t} \\
&= \left(1 - \frac{\|g_t\|_2^4}{g_t^\top Q g_t \cdot g_t^\top Q^{-1} g_t} \right) f(x_t).
\end{aligned}$$

The last line used the fact that $f(x_t) = \frac{1}{2}(x_t - x^\star)^\top Q (x_t - x^\star) = \frac{1}{2} g_t^\top Q^{-1} g_t$.

Now we use Kantorovich's inequality to get that

$$\frac{\|g_t\|_2^4}{g_t^\top Q g_t \cdot g_t^\top Q^{-1} g_t} \geq \frac{4 \lambda_1(Q) \lambda_n(Q)}{(\lambda_1(Q) + \lambda_n(Q))^2}.$$

So therefore we have

$$f(x_{t+1}) \leq \left(1 - \frac{4 \lambda_1(Q) \lambda_n(Q)}{(\lambda_1(Q) + \lambda_n(Q))^2} \right) f(x_t) = \left(\frac{\lambda_1(Q) - \lambda_n(Q)}{\lambda_1(Q) + \lambda_n(Q)} \right)^2 f(x_t).$$

Since $f(x^\star) = 0$ this concludes the proof. □

[gl: Optional]

Proof of Kantorovich's inequality. Without loss of generality, we can assume that Q is a diagonal matrix with entries $\lambda_1, \cdots, \lambda_n$. This is because if Q can always be written as $V\Sigma V^\top$, so we can always reparameterize g_t as Vg_t' for some g_t'. Also suppose that $\|g_t\|_2 = 1$. Let α_i be the squared ith entry of g_t. The inequality can be rewritten as

$$\left(\sum_i \alpha_i \cdot \lambda_i\right)\left(\sum_i \frac{\alpha_i}{\lambda_i}\right) \leq \frac{(\lambda_1+\lambda_n)^2}{4\lambda_1\lambda_n}$$

Probabilistic Proof. We prove the following continuous statement: if X is a random variable in $[a, b]$ we have

$$\mathbb{E}[X]\,\mathbb{E}[X^{-1}] \leq \frac{(a+b)^2}{4ab}.$$

By Cauchy Schwarz we know that

$$\mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])] \leq \left(\mathbb{E}\left[(X-\mathbb{E}[X])^2\right]\right)^{1/2}\left(\mathbb{E}\left[(Y-\mathbb{E}[Y])^2\right]\right)^{1/2}$$

If we take $Y = -X^{-1}$ we get

$$\mathbb{E}[X]\,\mathbb{E}\left[X^{-1}\right] \leq 1 + \left(\mathbb{E}\left[(X-\mathbb{E}[X])^2\right]\right)^{1/2}\left(\mathbb{E}\left[(X^{-1}-\mathbb{E}[X^{-1}])^2\right]\right)^{1/2}$$

Now we use the fact that for any random variable Z we have $\mathbb{E}\left[(X-\mathbb{E}[X])^2\right] = \inf_a \mathbb{E}[(Z-a)^2]$. So we can get

$$\mathbb{E}[X]\,\mathbb{E}\left[X^{-1}\right] \leq 1 + \left(\frac{b-a}{2}\right)\cdot\left(\frac{b-a}{2ab}\right) = \frac{(a+b)^2}{4ab}.$$

[gl: https://math.stackexchange.com/questions/4004848/an-upper-bound-of-product-of-two-inner-pr is another proof.] □

4 Smooth and Strongly Convex Optimization

[gl: If time permits, cover backtracking line search?]

At this point, we have seen how gradient descent behaves on quadratic functions when we have a good handle on how close it looks like to the identity. But, a useful question to ask is – did we deeply use the fact that the function was a quadratic? Can we get a similar result by assuming something weaker?

For instance, based on the above, it seems plausible that a function that always looks close enough to a quadratic at each iterate will make GD behave similarly to the exactly quadratic case. In this section, we will see that this is indeed the case.

Let us first formalize what we mean by a function looking close enough to a quadratic at each iterate.

Definition 4 (Nice function (we couldn't think of a better name...))**.** *Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$ denote some sequence of points in $\mathbb{R}^d$. We say that f is a "nice function" with respect to $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$ and a reference $\boldsymbol{x}^\star$ if $f(\boldsymbol{x}^\star) \le f(\boldsymbol{x}_i)$ for all $i \in \{1, \dots, T\}$, and if for all $i \in \{1, \dots, T\}$, we have*

$$f(\boldsymbol{x}_{i+1}) \le f(\boldsymbol{x}_i) + \langle \nabla f(\boldsymbol{x}_i), \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \rangle + \frac{\beta}{2} \cdot \|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_2^2 \qquad \textit{quadratic upper bound}$$

$$f(\boldsymbol{x}_i) \le f(\boldsymbol{x}^\star) + \frac{1}{2\alpha} \|\nabla f(\boldsymbol{x})\|_2^2 \qquad \textit{suboptimality vs gradient}$$

The second condition is a bit less natural, but we will see how this relates to things we have seen previously.

As a sanity check, we will see that quadratics are nice functions for any sequence of points in $\mathbb{R}^d$. However, Definition 4 is a lot more general, as we might only care about f behaving well on a particular sequence of points, or a particular subspace, etc.

For example, when f is a "nice function" with respect to $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$ generated by running gradient descent, we have:

Theorem 4. *Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$ be generated by running gradient descent on f with fixed step size η. If f is a nice function with respect to $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$ with reference $\boldsymbol{x}^\star$, then we have for all i that*

$$f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}^\star) \le (f(\boldsymbol{x}_i) - f(\boldsymbol{x}^\star)) \left(1 - 2\alpha \left(\eta - \frac{\beta\eta^2}{2}\right)\right).$$

In particular, if we choose $\eta = 1/\beta$, then we improve our function distance from the reference $\boldsymbol{x}^\star$ significantly in each round.

$$f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}^\star) \le (f(\boldsymbol{x}_i) - f(\boldsymbol{x}^\star)) \left(1 - \frac{\alpha}{\beta}\right).$$

Proof of Theorem 4. Each difference has a convenient form – we write this below.

$$\boldsymbol{x}_{i+1} - \boldsymbol{x}_i = -\eta \nabla f(\boldsymbol{x}_i)$$

Thus, we get bounds on $f(\boldsymbol{x}_{i+1})$, which follow from plugging into the upper bound part of Definition 4.

$$f(\boldsymbol{x}_{i+1}) \le f(\boldsymbol{x}_i) - \eta \|\nabla f(\boldsymbol{x}_i)\|_2^2 + \frac{\beta\eta^2}{2} \|\nabla f(\boldsymbol{x}_i)\|_2^2 = f(\boldsymbol{x}_i) - \|\nabla f(\boldsymbol{x}_i)\|_2^2 \left(\eta - \frac{\beta\eta^2}{2}\right) \tag{1}$$

Next, we use

$$\|\nabla f(\boldsymbol{x}_i)\|_2^2 \ge 2\alpha \left(f(\boldsymbol{x}_i) - f(\boldsymbol{x}^\star)\right).$$

Let $\delta_i \coloneqq f(\boldsymbol{x}_i) - f(\boldsymbol{x}^\star)$, subtract $f(\boldsymbol{x}^\star)$ from both sides of (1), plug in (2), and we get

$$\delta_{i+1} \le \delta_i - 2\alpha\delta_i \left(\eta - \frac{\beta\eta^2}{2}\right) = \delta_i \left(1 - 2\alpha \left(\eta - \frac{\beta\eta^2}{2}\right)\right).$$

Now, let us optimize over the step size η. Clearly, it is enough to maximize the below function over η.

$$2\alpha\eta - \alpha\beta\eta^2$$

It is easy to check that the best value of $\eta = (2\alpha)/(2\alpha\beta) = 1/\beta$. Plugging this all the way through gives us

$$\delta_{i+1} \leq \delta_i \left(1 - \frac{\alpha}{\beta}\right),$$

thereby completing the proof of Theorem 4. □

From this, a quick calculation reveals that to get ε-close in function value to the reference $\boldsymbol{x}^\star$, it is enough to run about $\delta_0 \cdot \beta/\alpha \cdot \ln(1/\varepsilon)$ iterations of gradient descent if f was "nice" with respect to the iterates of gradient descent. Note that we did not explicitly assume that f is convex in the above analysis.

One problem with this is that it does not immediately seem that useful – we are left with showing that the iterates of gradient descent satisfy Definition 4. This might depend on several things, including our initialization, what the gradients look like at various parts of f, etc. For starters, for simplicity, there are global conditions we can impose on f that will imply these for free.

We first handle the quadratic upper bound assumption. We give a global condition that will ensure that all the iterates of gradient satisfy it, regardless of how we initialize.

Definition 5 (Smooth function)**.** *We say f is β-smooth if:*

$$\text{for all } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d: \quad |f(\boldsymbol{x}) - f(\boldsymbol{y}) - \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle| \leq \frac{\beta}{2} \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 .$$

[nsm: picture]

In fact, Definition 5 implies the quadratic upper bound condition imposed by Definition 4 for *any* sequence of points, not necessarily just those generated by gradient descent.

We now discuss the suboptimality vs gradient condition. Definition 6 presents a natural condition under which we get what we need for free.

Definition 6 (Polyak-Łojasiewicz function)**.** *We say that f is α-PŁ if for all global minimizers $\boldsymbol{x}^\star$, we have:*

$$\text{for all } \boldsymbol{x} \in \mathbb{R}^d: \quad f(\boldsymbol{x}) \leq f(\boldsymbol{x}^\star) + \frac{1}{2\alpha} \|\nabla f(\boldsymbol{x})\|_2^2 .$$

5 Smooth Convex Optimization

6 Lipschitz and Bounded Convex Optimization

[gl: subgradients]

7 Stochastic Optimization

[gl: Also see `https://stanford.edu/~jduchi/PCMIConvex/Duchi16.pdf` for more details.]

In today's lecture we will study the stochastic gradient descent, a widely used algorithm which is the backbone of the success of machine learning.

[gl: To avoid confusion, we will use w as our optimization variable!]

A typical machine learning problem can be written as the following:

$$\min_w F(w) = \mathbb{E}_z[f(w; z)].$$

Here, we are again faced with an optimization problem. The goal is to minimize some function $F(x)$, which we assume to be convex. However, the difference compared to previous settings is that in typical machine learning scenarios, we do not actually know the function $F(\cdot)$, nor can we evaluate its gradients. However, we often can get some estimates of the function values and gradients, and we want to understand how we can use these estimates to solve the problem.

Why does this come up? For now, let's take a brief detour. Consider the standard machine learning setup, described below. Actually, many examples we have already talked about in the course are examples of machine learning tasks.

- We have some instance space $\mathcal{X}$ and some label space $\mathcal{Y}$. For example, $\mathcal{X}$ can be the space of all images, and $\mathcal{Y}$ can represent whether there is a dog in the picture or not. Another example we can use is $\mathcal{X}$ is some information about a patient, and $\mathcal{Y}$ can be some prediction about the patient, like their blood sugar level. We can even have more complicated settings. Let $\mathcal{X}$ be some sequence of words, and $\mathcal{Y}$ be a new word. Then our task here is to predict the next word in a sequence.
- There is some distribution in nature $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ pairs.
- As a machine learning practitioner, you get examples (x, y) which are drawn from the distribution. We assume that the examples are identically and independently distributed.
- You are also given a hypothesis, or predictor class $\mathcal{F} \subseteq (\mathcal{X} \to \mathcal{Y})$, which is a collection of mappings from instances to labels. For example, $\mathcal{F}$ can be a linear function class, or even a deep neural network.
- Lastly, there is also a loss function $\ell(f, (x, y))$ which measures the predictive power or performance of f. Some examples are $\ell(f, (x, y)) = (f(x) - y)^2$ (the square loss), or $\ell(f, (x, y)) = \max\{0, 1 - f(x)y\}$ (hinge loss). Or we can consider the cross entropy loss $-y \cdot \log f(x) - (1 - y) \cdot \log(1 - f(x))$.
- The goal of machine learning is to minimize the expected loss:

  $$\min_{f \in \mathcal{F}} L(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f, (x, y))].$$

  It is easy to see that this is exactly the previous stochastic optimization problem, but with different notation. So it is beneficial to understand how to solve this general stochastic optimization problem.
- A popular strategy for solving this is empirical risk minimization, which solves the following optimization problem.

  $$\min_{f \in \mathcal{F}} \widehat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f, (x_i, y_i)).$$

The key idea in SGD is to use *estimates* of $F(\cdot)$ in order to approximately solve the optimization problem. In particular, we will adopt the following abstraction:

- In every iteration, we get *unbiased* estimates of the true gradient. That is, we will get a random estimate g_t such that $\mathbb{E}[g_t|w_t] = \nabla f(w_t)$.

In machine learning, this assumption is (roughly) satisfied. We usually get a dataset of i.i.d. pairs (x, y) and we can evaluate the loss or the gradient of the loss $\ell(f, (x, y))$. For empirical risk minimization, we can sample a random index $i \in [n]$ in every round and evaluate the gradient at that round.

However, if we wanted to do gradient descent on the full dataset, this would require us to iterate through all n datapoints before taking a step. This can be potentially wasteful, because we are waiting to evaluate n gradients before doing any optimization.

A subtle point is that if we iterate through the same dataset multiple times, as is common in practice called multi-pass SGD, it turns out that the estimates will not be independent of each other for the stochastic optimization problem! Usually, this is not that big of a problem, but it is something to keep in mind.

7.1 SGD for Convex Lipschitz Bounded

In stochastic gradient descent, we will instead make the update:

$$w_{t+1} = w_t - \eta_t g_t.$$

We provide some intuition on the benefits of SGD for ERM:

- Practical data usually has a lot of redundancy, so waiting to evaluate all the gradients before taking a step could be inefficient.
- In contrast, SGD will be extremely efficient at the beginning of optimization, as it gets fast initial performance with low cost-per-iteration.

We will analyze the performance of this algorithm. First we will study convex Lipschitz bounded functions.

Theorem 5. *Let F be a convex function with minimizer $w^\star = \operatorname{argmin}_{w \in B_2(B)} F(w)$. Suppose we run SGD for T iterations with constant step size η. Also assume that $\|g_t\|_2 \le G$ with probability 1. Then we have*

$$\mathbb{E}[F(\bar{w})] - F(w^\star) \le \frac{BG}{\sqrt{T}}.$$

Proof. We will use $g_{1:t}$ to denote the sequence of vectors $g_1, \cdots, g_t$.

By Jensen's inequality we have

$$\mathbb{E}_{g_{1:T}}[F(\bar{w}) - F(w^\star)] \le \mathbb{E}_{g_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T} F(w_t) - F(w^\star)\right].$$

Now we will prove a guarantee which holds for any $g_{1:T}$.

Lemma 1. *For any sequence of vectors $g_{1:T}$ and vector $w^\star$, the OGD algorithm with initialization $w_1 = 0$ satisfies*

$$\sum_{t=1}^{T} \langle w_t - w^\star, g_t \rangle \le \frac{\|w^\star\|}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|g_t\|^2.$$

In particular if $\|g_t\| \le G$ and $\|w^\star\| \le B$ by setting $\eta = \sqrt{\frac{B^2}{G^2 T}}$ we have

$$\sum_{t=1}^{T} \langle w_t - w^\star, g_t \rangle \le \frac{BG}{\sqrt{T}}.$$

Using the lemma, we get that

$$\mathbb{E}_{g_{1:T}}\left[\sum_{t=1}^{T} \langle w_t - w^\star, g_t \rangle\right] \le \frac{BG}{\sqrt{T}}$$

So it is left to show that

$$\mathbb{E}_{g_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T} F(w_t) - F(w^\star)\right] \le \mathbb{E}_{g_{1:T}}\left[\sum_{t=1}^{T} \langle w_t - w^\star, g_t \rangle\right].$$

Recall the law of total expectation: for every pair of random variables X, Y and function g, we have $\mathbb{E}_X[g(X)] = \mathbb{E}_Y \mathbb{E}_X[g(X)|Y]$. Setting $X = g_{1:t}$ and $Y = g_{1:t-1}$ we get that

$$\mathbb{E}_{g_{1:T}}[\langle w_t - w^\star, g_t \rangle] = \mathbb{E}_{g_{1:t}}[\langle w_t - w^\star, g_t \rangle] = \mathbb{E}_{g_{1:t-1}} \mathbb{E}_{g_{1:t}}[\langle w_t - w^\star, g_t \rangle \mid g_{1:t-1}]$$

Once $g_{1:t-1}$ are known, the value of w_t is not random so we have

$$\mathbb{E}_{g_{1:t-1}} \mathbb{E}_{g_{1:t}}[\langle w_t - w^\star, g_t \rangle \mid g_{1:t-1}] = \mathbb{E}_{g_{1:t-1}} \langle w_t - w^\star, \mathbb{E}_{g_t}[g_t \mid g_{1:t-1}] \rangle$$

We know that w_t only depends on $g_{1:t-1}$ and SGD requires that $\mathbb{E}_{g_t}[g_t \mid w_t] = \nabla f(w_t)$, so therefore we have $\mathbb{E}_{g_t}[g_t \mid g_{1:t-1}] = \nabla f(w_t)$ and by convexity.

$$\mathbb{E}_{g_{1:t-1}} \langle w_t - w^\star, \mathbb{E}_{g_t}[g_t \mid g_{1:t-1}] \rangle \ge \mathbb{E}_{g_{1:t-1}}[F(w_t) - F(w^\star)] = \mathbb{E}_{g_{1:T}}[F(w_t) - F(w^\star)].$$

Therefore averaging over t and using linearity of expectation we have shown the inequality. □

It is left to prove the lemma.

Proof. We can calculate that

$$\begin{aligned}
\langle w_t - w^\star, g_t \rangle &= \frac{1}{\eta} \langle w_t - w^\star, \eta g_t \rangle \\
&= \frac{1}{2\eta}\left(-\|w_t - w^\star - \eta g_t\|^2 + \|w_t - w^\star\|^2 + \eta^2 \|g_t\|^2\right) \\
&= \frac{1}{2\eta}\left(-\|w_{t+1} - w^\star\|^2 + \|w_t - w^\star\|^2\right) + \frac{\eta}{2}\|g_t\|^2.
\end{aligned}$$

Now we will sum over t for both sides to get

$$\begin{aligned}
\sum_{t=1}^{T} \langle w_t - w^\star, g_t \rangle &= \frac{1}{2\eta}\left(\|w_1 - w^\star\|^2 - \|w_{T+1} - w^\star\|^2\right) + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2 \\
&\le \frac{1}{2\eta}\|w^\star\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|^2.
\end{aligned}$$

For the second part of the lemma, we just need to apply the bounds on $\|w^\star\|$, $\|g_t\|$, and the value of η. □

This rate matches what we got in the deterministic case with exact gradients in each round. This means that SGD is an extremely robust algorithm; the convergence rate is barely affected by the amount of noise.

What happens if we also assume smoothness?Unfortunately, in stochastic optimization, smoothness does not really help at all. The intuitive reason is because the noise of the gradients can still be large near the optimum. This is in contrast for the deterministic case, where we can get $1/T$ rate for smooth optimization (or a $1/T^2$ rate with acceleration).

Theorem 6. *Suppose F is μ-smooth, and suppose the stochastic oracle is such that $\mathbb{E}\|\nabla f(x) - g\|_2^2 \leq \sigma^2$. Also let B denote the bound on the initial distance to the optimum, i.e., $\|w^\star\| \leq B$. Then stochastic gradient descent with $\eta = 1/\left(\beta + 1/\frac{B}{\sigma}\sqrt{\frac{2}{T}}\right)$ achieves a guarantee*

$$\mathbb{E}[F(\bar{w})] - F(w^\star) \leq B\sigma\sqrt{\frac{2}{T}} + \frac{\beta B^2}{T}.$$

This result allows us to interpolate between the deterministic setting and the noisy setting.

7.2 SGD for Smooth and Strongly Convex

Interestingly, once we use SGD, we cannot get the linear convergence anymore, even for smooth and strongly convex optimization. See e.g., https://arxiv.org/pdf/1109.5647.pdf. But strong convexity does help a little bit.

Theorem 7. *Suppose F is λ-strongly convex and μ-smooth, and that $\mathbb{E}\|g_t\|^2 \leq G^2$. If we pick $\eta_t = 1/(\lambda t)$ then it holds for any T that*

$$\mathbb{E}[F(w_T) - F(w^\star)] \leq \frac{2\mu G^2}{\lambda^2 T}.$$

To prove this theorem, we will prove this important lemma.

Lemma 2. *Suppose F is λ-strongly convex and $\mathbb{E}\|g_t\|^2 \leq G^2$. Then picking $\eta_t = 1/(\lambda t)$ then it holds for any T that*

$$\mathbb{E}\left[\|w_T - w^\star\|^2\right] \leq \frac{4G^2}{\lambda^2 T}.$$

The theorem is a straightforward corollary of this lemma by using μ-smoothness.

Proof. By strong convexity, we know that

$$\langle \nabla f(w_t), w_t, w^\star \rangle \geq F(w_t) - F(w^\star) + \frac{\lambda}{2}\|w_t - w^\star\|^2,$$

and also that

$$F(w_t) - F(w^\star) \geq \frac{\lambda}{2}\|w_t - w^\star\|^2.$$

Therefore we can compute that

$$\mathbb{E}\left[\|w_{t+1} - w^\star\|^2\right] = \mathbb{E}\left[\|w_t - \eta_t g_t - w^\star\|^2\right]$$

$$
\begin{aligned}
&= \mathbb{E}\big[\|w_t - w^\star\|^2\big] - 2\eta_t\,\mathbb{E}[\langle g_t, w_t - w^\star\rangle] + \eta_t^2\,\mathbb{E}\|g_t\|^2 \\
&= \mathbb{E}\big[\|w_t - w^\star\|^2\big] - 2\eta_t\,\mathbb{E}[\langle \nabla f(x_t), w_t - w^\star\rangle] + \eta_t^2\,\mathbb{E}\|g_t\|^2 \\
&\le \mathbb{E}\big[\|w_t - w^\star\|^2\big] - 2\eta_t\,\mathbb{E}\left[F(w_t) - F(w^\star) + \frac{\lambda}{2}\|w_t - w^\star\|^2\right] + \eta_t^2 G^2 \\
&\le (1 - 2\eta_t\lambda)\,\mathbb{E}\big[\|w_t - w^\star\|^2\big] + \eta_t^2 G^2.
\end{aligned}
$$

By choosing $\eta_t = 1/(\lambda t)$ we see that

$$
\mathbb{E}\big[\|w_{t+1} - w^\star\|^2\big] \le \left(1 - \frac{2}{t}\right)\mathbb{E}\big[\|w_t - w^\star\|^2\big] + \frac{G^2}{\lambda^2 t^2}.
$$

Now we apply an inductive argument. Observe that at $t = 1$ we have

$$
\frac{\lambda}{2}\|w_1 - w^\star\|^2 \le \langle \nabla f(w_1), w_1 - w^\star\rangle,
$$

which means that

$$
\|\nabla f(w_1)\| \ge \frac{\lambda}{2}\|w_1 - w^\star\|.
$$

In addition we know that

$$
\begin{aligned}
\mathbb{E}\|g_1\|^2 &= \mathbb{E}\|\nabla f(w_1) + (g_1 - \nabla f(w_1))\|^2 \\
&= \mathbb{E}\|\nabla f(w)\|^2 + \mathbb{E}\|g_1 - \nabla f(w_1)\|^2 + 2\,\mathbb{E}[\langle g_1 - \nabla f(w_1), \nabla f(w_1)\rangle] \\
&\ge \mathbb{E}\|\nabla f(w)\|^2.
\end{aligned}
$$

Therefore we get that

$$
\mathbb{E}\big[\|w_1 - w^\star\|^2\big] \le \frac{4}{\lambda^2}\,\mathbb{E}\big[\|\nabla f(w_1)\|^2\big] \le \frac{4}{\lambda^2}\,\mathbb{E}\big[\|g_1\|^2\big] \le \frac{4G^2}{\lambda^2}.
$$

At $t = 2$, we can also get that

$$
\mathbb{E}\big[\|w_2 - w^\star\|^2\big] \le \frac{G^2}{4\lambda^2} \le \frac{4G^2}{\lambda^2 \cdot 2}. \tag{2}
$$

Now we apply induction. Suppose that at iterate t we have $\mathbb{E}\big[\|w_2 - w^\star\|^2\big] \le \frac{4G^2}{\lambda^2 \cdot t}$. Then at iterate $t + 1$ we have

$$
\begin{aligned}
\mathbb{E}\big[\|w_{t+1} - w^\star\|^2\big] &\le \left(1 - \frac{2}{t}\right)\mathbb{E}\big[\|w_t - w^\star\|^2\big] + \frac{G^2}{\lambda^2 t^2} \\
&\le \left(1 - \frac{2}{t}\right)\cdot\frac{4G^2}{\lambda^2 t} + \frac{G^2}{\lambda^2 t^2} \\
&= \frac{4G^2}{\lambda^2 t} - \frac{7G^2}{\lambda^2 t^2} \le \frac{4G^2}{\lambda^2(t+1)}.
\end{aligned}
$$

So the induction step holds. □

Remark: a similar conclusion holds for the averaged iterate $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ (but with slightly worse constants).

[gl: Discuss comparison between batch GD and SGD for empirical risk minimization.]

8 Acceleration

9 Linear Programming

10 Derivative Free Optimization

In previous lectures, we have considered optimization problems where one gets access to function evaluations $f(x)$ as well as gradients $\nabla f(x)$. This is usually a reasonable setting in practice. An additional setting (which we do not consider here, but is an exciting research direction) is studying *higher order* methods for optimization, where in addition receiving function evaluations and gradients, one also gets to see higher order derivatives. For example, if one can readily calculate the Hessian $\nabla^2 f(x)$, there are many algorithms that can use such information. These are called *second-order* methods.

Derivative Free Convex Optimization. In this lecture, however, we go the other direction and study what is called *derivative-free optimization* (also called *black-box* or *zero-th order* optimization). Here the goal is same as before: find a solution to the problem

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

where $\Omega \subseteq \mathbb{R}^d$ is a constraint set. However, now we restrict ourselves to only receiving function values $f(x)$ and not getting access to gradients $\nabla f(x)$!

Before we study methods for solving this problem, we motivate why to study it. Often, the feedback we get comes from a system for which we cannot efficiently compute derivatives for. Here are several examples:

- Hyperparameter selection. Even for the first order methods we have studied so far in this course, we have had to make certain parameter choices. These are called "hyperparameters". Especially in the deep learning era, one can think of many tunable hyperparameters: width of the model, depth of the model, learning rate, how much acceleration to use, etc. If we want to select the best model, we can also optimize over these hyperparameters.
- Control Problems. Imagine trying to design a system to control a drone to hover at a certain height with the minimum amount of power (In fact later today we will program a strategy to do so). A common strategy is PID control, which has 3 parameters. We can try to select the optimal choice of parameters using derivative free optimization.
- Routing Problems. Imagine every day we want to go from home to school, and we pick a different path $x \in \Omega$. When we get to school, we can see how long it took (some measure of cost which we denote $f(x)$), but we do not get to see gradients.
- Other examples in protein-folding, circuit design, etc.

10.1 Random Search Algorithm

Let's introduce a simple algorithm to solve this problem. It is stated in Algorithm 2.

Algorithm 2 is mostly meant to illustrate the central ideas. Basically, whatever was true for gradient descent that we showed before is true for Algorithm 2, up to a multiplicative cost of d. This is what

Algorithm 2 Random Search

1: Pick an initial point $x_0 \in \Omega$.
2: **for** $t = 1, 2, \cdots, T$:
3: Let $u_t \in \mathbb{S}^{d-1}$ be a random direction.
4: Set step size $\eta_t = \operatorname{argmin}_{\eta > 0} f(x_t - \eta u_t)$.
5: Update $x_{t+1} = x_t - \eta_t u_t$.
6: **Return**

we would expect, because for gradient descent, in every step we get $O(d)$ information, but here we only get $O(1)$.

Analysis for Smooth Functions. Let's assume that f is a β-smooth convex differentiable function. Recall that this means that for any η, we have

$$\begin{aligned} f(x_t - \eta u_t) &\leq f(x_t) - \eta \langle u_t, \nabla f(x_t) \rangle + \frac{\eta^2 \beta}{2} \|u_t\|^2 \\ &= f(x_t) - \eta \langle u_t, \nabla f(x_t) \rangle + \frac{\eta^2 \beta}{2}, \quad \text{since } u_t \in \mathbb{S}^{d-1}. \end{aligned}$$

We can optimize over η now. For some choice of $\eta^\star$ we have

$$f(x_t - \eta^\star u_t) = f(x_t) - \frac{1}{\beta} \langle u_t, \nabla f(x_t) \rangle^2.$$

Moreover because of our choice for the step size η_t we have

$$f(x_{t+1}) = f(x_t - \eta_t u_t) \leq f(x_t - \eta^\star u_t) = f(x_t) - \frac{1}{\beta} \langle u_t, \nabla f(x_t) \rangle^2.$$

Taking expectations we have

$$\begin{aligned} \mathbb{E}[f(x_{t+1})] &\leq f(x_t) - \frac{1}{\beta} \mathbb{E}\big[\langle u_t, \nabla f(x_t) \rangle^2\big] \\ &\leq f(x_t) - \frac{1}{\beta d} \mathbb{E}\big[\|\nabla f(x_t)\|^2\big]. \end{aligned}$$

In expectation, this is similar to the type of results for gradient descent that we saw before, up to an additional factor of d. One can show similar convergence guarantees as before, but we will not do so now.

Line Search. Algorithm 2 is not that practical, since step 4 involves doing an exact line search. In practice however, we can approximate this with some kinds of heuristics. For 1D minimization, we can use something in Golden-Section Search (GSS), see, e.g., https://en.wikipedia.org/wiki/Golden-section_search. This will be implemented for you in code. It is also implemented in scipy: https://docs.scipy.org/doc/scipy/reference/optimize.minimize_scalar-golden.html.

Other Methods. There are other methods for solving derivative-free optimization. The wikipedia page has more details: https://en.wikipedia.org/wiki/Derivative-free_optimization. For example, there is a more sophisticated version of this due to Nelder and Mead https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method. Nelder-Mead works well in practice but does not have strong theoretical guarantees. However, if you are looking for a numerical method for zeroth-order optimization, it is implemented in packages like `scipy`.

10.2 Application: Optimal Control

We introduce the problem of optimal control. The optimal control problem has a few features:

- The states are represented by $x \in \mathbb{R}^{d_x}$.
- The actions are represented by $a \in \mathbb{R}^{d_a}$.
- There is noise or random disturbance which can be represented by some e_t.

The optimal control problem has a dynamical system model which is given by:

$$x_{t+1} = f(x_t, a_t, e_t).$$

The transition function f tells us for any input state x and action a, as well as noise e how the system transitions. For example, one popular model is the so-called linear dynamical system:

$$x_{t+1} = Ax_t + Ba_t + e_t.$$

However, we can imagine more complicated transition functions f. Associated with the optimal control problem is a cost function:

$$c_t(x_t, a_t).$$

This measures the amount of cost we incur. For example, if we want to control a robotic car, the cost can capture the distance from the desired trajectory as well as the amount of energy consumed.

The goal of the optimal control problem is to compute a good policy $\widehat{\pi} : \mathcal{X} \to \mathcal{A}$ that minimizes the total cost:

$$\min_{\widehat{\pi}} \sum_{t=1}^{T} \mathbb{E}_{e_t}[c_t(x_t, a_t)].$$

The policy tells us in every given state what action to take.

- So this is exactly a stochastic optimization problem! Often we will actually parameter the policy function with some parameter θ, so the problem can be equivalently written as

$$\min_{\theta} \sum_{t=1}^{T} \mathbb{E}_{e_t}[c_t(x_t, \pi_\theta(x_t))].$$

- We are working here with what is called Markovian policies, which only take as input the current state. One can also consider policies which are "history-dependent": they take into account the entire previous sequence of states and actions, i.e., the policy at time t is a function of $\tau_t = (x_1, a_1, x_2, a_2, \cdots, x_t)$.
- Here, we are using notation which is standard in RL. It is different than what we have been using before. The parameter to optimize here is θ.
- How can we solve this problem? In general, the dynamics function f may not be known to us. For example if f models the heat flow in a massive data center, then knowing f requires a deep understanding of physics and heat transfer. Even if we know f, computing the gradients of this cost function might be very difficult. Another approach is to learn an approximate model

of f: for example we can approximate it as a linear system. This is approach is called different names in different communities. In reinforcement learning, its often called model-based RL; another name from control theory is system identification. Once we have a simpler model $\widehat{f}$, then it might be possible to find a good policy. For linear dynamical systems, there is a rich literature on how to compute a good policy.

- We will go with a direct, model-free approach. In this case, our goal is to directly optimize the function without worrying about modeling the transition model. We can actually do so using the derivative-free optimization methods that we introduced before.

Solving Optimal Control with Random Search. We will illustrate this with the Flappy Bird game. Imagine we have a bird flying at a constant velocity, and it is being pulled down by gravity. The goal is to control the bird so that it doesn't hit the ground while minimizing the amount of acceleration.

- The bird's state can be written as two variables: $x = (x_h, x_v)$ where x_h represents the current height and x_v represents the velocity.
- The action that the bird can take is (upwards) accleration, represented by a scalar $a > 0$.
- The cost function is given by $c_t(x_t, a_t) = -10 \times \mathbb{1}\{x_t < 0\} + a_t$. That is, we get a large cost if our height is negative, and we get a smaller cost every time we accelerate.

the goal is for the bird to not hit the ground while minimizing the amount of acceleration that it uses.

[gl: Flappy Bird lab.]

Policy Gradient. We will introduce an alternative way to solve the RL problem, which can also be thought of as a zeroth order optimization problem for RL. This is called the policy gradient method, and it underlies a lot of more state-of-the-art methods in RL that have received a lot of attention recently (TRPO, PPO). [gl: See also https://arxiv.org/pdf/1806.09460.pdf and https://lilianweng.github.io/posts/2018-04-08-policy-gradient/]

10.3 Bibliographical Note

Some of the material in this section comes from the book [CSV09] as well as the lecture notes https://ee227c.github.io/code/lecture20.html.

References

[CSV09] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.