Optimization for Data Science

Lecture Notes, FS 23

Bernd Gärtner and Niao He, ETH
Martin Jaggi, EPFL

February 17, 2023

Contents

Chapter 1

Introduction

Contents

1.1 About this lecture

These are lecture notes for the ETH course **261-5110-00L Optimization for Data Science**. This course has partially been co-developed with the EPFL course **CS-439 Optimization for machine learnbiaseding**; the two courses share some of their content but also differ in some material.

This course provides an in-depth theoretical treatment of classical and modern optimization methods that are relevant in data science. The emphasis is on the motivations and design principles behind the algorithms, on provable performance bounds, and on the mathematical tools and techniques to prove them. The goal is to equip students with a fundamental understanding about why optimization algorithms work, and what their limits are. This understanding will be of help in selecting suitable algorithms in a given application, but providing concrete practical guidance is not our focus.

In this first chapter, we discuss the role of optimization *for Data Science*. This turns out to be quite different from the classical role where an optimization algorithm (for example Kruskal's) solves a well-defined optimization problem (find the minimum spanning tree). In Data Science, we have *learning* problems, and they are not always well-defined. Optimization typically happens on training data, but even a perfect result may fail to solve the underlying learning problem. This is not a failure of the optimization algorithm but of the model in which it was applied. In Data Science, optimization is merely one ingredient in solving a learning problem. While this course focuses on optimization, it does not turn you into a holistic data scientist. But it will allow you to better understand and explore the (after all not so small) optimization corner in the Data Science landscape.

In his arcticle *50 years of Data Science*, David Donoho outlines the six divisions of what he calls *Greater Data Science* [Don17]:

1. Data Gathering, Preparation, and Exploration
2. Data Representation and Transformation
3. Computing with Data
4. Data Modeling
5. Data Visualization and Presentation

6. Science about Data Science

Even Computing with Data (which some may think that Data Science is mainly about) is only one of six divisions, and optimization is a subdivision of that. But optimization is important also in Data Modeling: towards being able to actually optimize a given model, the model designer must already be aware of the computational methods that are available, what they can in principle achieve, and how efficiently they can do this. The classical worst-case complexity of an optimization algorithm is rarely the right measure of applicability in practice. In Data Modeling, one also needs to make sure that the result of the optimization is meaningful for the learning problem at hand.

To summarize the state of affairs at this point: optimization is an *enabling technology* in Data Science, but it's not what Data Science is about. Optimization algorithms should not be used blindly, by confusing the optimization problem being solved with the ground truth. The optimization problem helps to solve a learning problem, and the latter is the ground truth. The same learning problem can in principle be modeled with many different optimization problems. What counts in the end is whether the result of the optimization is useful towards learning.

In the remainder of this chapter, we elaborate on this high-level summary. We start by framing the minimum spanning tree problem as a learning problem and present some surprising implications that this has for the classical minimum spanning tree algorithms. Then we introduce the predominant application of optimization in Data Science, namely empirical risk minimization as a way to learn from training data. We concisely explain overfitting, underfitting, generalization, regularization and early stopping. We then present two classical theorems as showcases for distribution-independent and distribution-dependent techniques to guarantee that emprirical risk minimization entails learning. Finally, we talk about the role of computational complexity in Data Science, by discussing the evolution from classical worst-case complexity over average-case complexity to smoothed complexity. The simplex method for linear programming serves as a prime example to document this evolution.

1.2 Algorithms in Data Science

The classical picture of an algorithm, as portrayed in many textbooks, is that of a computational procedure that transforms input into output. Here is an example of how Cormen, Leiserson, Rivest, and Stein describe such a transformation in their *Introduction to Algorithms* [CLRS09, Section 23.1]:

> Assume that we have a connected, undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}$ and wish to find a minimum spanning tree for G.

Then they present two algorithms for solving this problem, *Kruskal's algorithm*, and *Prim's algorithm*. Kruskal's algorithm maintains a forest (initially empty), and in each step adds an edge of minimum weight that connects two trees in the forest. Prim's algorithm maintains a tree (initially consisting of an arbitrary vertex), and in each step adds an edge of minimum weight that connects the tree to its complement.

Both algorithms are analyzed in terms of their worst case performance. Kruskal's algorithm with a suitable union-find data structure needs time $\mathcal{O}(|E| \log |E|)$. Prim's algorithm takes time $\mathcal{O}(|E| \log |V|)$ which can be improved to $\mathcal{O}(|E| + |V| \log |V|)$ using Fibonacci Heaps. So Prim's algorithm is faster for dense graphs. Case closed.

In Data Science, the viewpoint is very different. Here, the starting point are data which we want to explain in some (not always prespecified) way. An algorithm is a tool to help us in finding such an explanation. The data typically come from measurements or experiments, and there is not a single input, but a typically unknown input *distribution* from which a measurement or an experiment draws a (noisy) sample. In such a situation, we are not interested in explaining a concrete sample, but we want to explain the nature of the distribution. And this changes the way in which we should evaluate an algorithm.

The prevalent quality measure is how good an explanation is in expectation (over the input distribution). If we don't know the distribution, we are stuck with the empirical approach: sample a finite number of inputs (these are the training data), and based on them, construct an explanation for the whole input distribution. An algorithm is evaluated according to how good this explanation is. Classical criteria such as runtime or space are considered only to ensure that we can actually run the algorithm efficiently on possibly huge data.

We can summarize the situation as follows: in Data Science, an algorithm solves a *learning problem* (finding an explanation for data), not a classical computational problem in which every input must be mapped to a specified output.

1.2.1 Case Study: Learning the minimum spanning tree

For the minimum spanning tree problem, the above Data Science viewpoint changes the picture quite radically as shown in a case study by Gronskiy [Gro18, Chapter 4]. The setup is that there are ground truth weights which are all roughly the same, but they are subject to random noise which has the potential to produce different orderings of edges by weight.

As a consequence, we may see different minimum spanning trees for different samples drawn from the induced weight distribution. The desired explanation of the data is the unknown ground truth minimum spanning tree. So we want an algorithm that produces a spanning tree as close to the ground truth as possible (measured in the number of common edges, say).

If the noise is negligible, we can solve the problem perfectly from one sample, using any algorithm that computes the minimum spanning tree of the sample. But as the noise becomes more and more significant, this algorithm inevitably produces worse and worse explanations. Whether Kruskal's or Prim's algorithm is used doesn't make any difference.

But the picture becomes more interesting once we have *two* samples X', X'' (formally, these are weighted graphs with the same underlying undirected graph). Consider running algorithm A (think of Kruskal's or Prim's) in parallel on these two different noisy versions of the ground truth. In each step, one edge is added to each of the two trees. After step t, let $A_t(X')$ and $A_t(X'')$ be the sets of spanning trees that are still possible for X' and X'', taking the edges already processed into account. We have $A_0(X') = A_0(X'')$, the set of all spaning trees, but as t grows, the sets $A_t(X')$ and $A_t(X'')$ (and also their intersection) are shrinking.

There is a sweet spot (a value of t) that maximizes the *joint information*, defined as the probability that two spanning trees, chosen uniformly at random from $A_t(X')$ and $A_t(X'')$, respectively, are equal. At this point, we stop the algorithm and return a spanning tree *randomly chosen* from $A_t(X') \cap A_t(X'')$ as our explanation.

It can be shown experimentally that this *early stopping* works quite well (compared to standard attempts to learn the ground truth minimum spanning tree from two samples). More interestingly, it now matters which algorithm is chosen, not because of runtime as in the classical setting, but because of explanation quality. As Gronskiy shows, Kruskal's algorithm with early stopping leads in expectation to a much better explanation than Prim's algorithm with early stopping.

In fact, Gronskiy shows that a *third* algorithm (not discussed by Cormen, Leiserson, Rivest and Stein at all, due to its abysmal worst-case complexity) is even better than Kruskal's algorithm. This is the *Reverse Delete* algorithm. It starts with the full graph, and in every step it deletes the edge of *maximum* weight that does not disconnect the graph.

The takeway message at this point is that classical algorithms and Data Science algorithms have quite different design goals, and this in particular holds in optimization. In the following sections, we will elaborate on this in some more detail, with a focus on optimization.

1.3 Expected risk minimization

Optimization is a key ingredient in the process of learning from data. To make this point, we start by introducing a general framework that encompasses many concrete learning problems, and we frame the learning task as an idealized optimization problem that typically cannot be solved due to lack of knowledge.

We have a data source $\mathcal{X}$ (a probability space, equipped with a probability distribution that we in general do not know). But we can tap the source by drawing independent *samples* $X_1, X_2, \ldots \sim \mathcal{X}$, as many as we want (or can afford). The notation $X \sim \mathcal{X}$ stands for "X randomly chosen from $\mathcal{X}$, according to the probability distribution over $\mathcal{X}$". Our goal is to explain $\mathcal{X}$ through these samples (we deliberately leave this vague, as it can mean many things).

Formally, a probability space is a triple consisting of a ground set, a sigma algebra (set of events), and a probability distribution function. It is common to abuse notation by identifying $\mathcal{X}$ with the ground set. We will also do this in the following.

As a concrete example, consider the situation where $\mathcal{X} = \{0, 1\}$ models a biased coin flip ($1 =$ head, $0 =$ tail), meaning that head comes up with

(unknown) probability $p^\star$ for $X \sim \mathcal{X}$. The desired explanation of $\mathcal{X}$ is the bias $p^\star$. By the law of large numbers, we have $p^\star = \lim_{n\to\infty} |\{i : X_i = 1\}|/n$, but we will not be able to compute this limit via finitely many samples. When sampling $X_1, X_2, \ldots, X_n \sim \mathcal{X}$, we can only hope to approximate $p^\star$. Concretely, the Chernoff bound tells us how large n needs to be in order for the empirical estimate $|\{i : X_i = 1\}|/n$ to yield a good approximation of $p^\star$ with high probability.

In our abstract framework, we have a class $\mathcal{H}$ of *hypotheses* that we can think of as possible explanations of $\mathcal{X}$, and our goal is to select the hypothesis that best explains $\mathcal{X}$. To quantify this, we use a *risk* or *loss* function $\ell : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$ that says by how much we believe that a given hypothesis $H \in \mathcal{H}$ fails to explain given data $X \in \mathcal{X}$. Then we consider the *expected risk*

$$\ell(H) := \mathbb{E}_{\mathcal{X}}[\ell(H, X)] \tag{1.1}$$

as the measure of quality of H for the whole data source (other quality measures are conceivable, but expected risk is the standard one). Formally, our goal is therefore to find a hypothesis with smallest expected risk, i.e.

$$H^\star = \operatorname*{argmin}_{H\in\mathcal{H}} \ell(H), \tag{1.2}$$

assuming that such an optimal hypothesis exists. To put the biased coin example into this framework, we choose hypothesis class $\mathcal{H} = [0, 1]$ (the candidates for the bias $p^\star$) and loss function $\ell(H, X) = (X - H)^2$. In Exercise 1, you will analyze this case and in particular prove that a unique optimal hypothesis $H^\star$ as in (1.2) exists and equals $p^\star$, the bias of the coin. So *mathematically*, we may be able to argue about expected risk.

But *computationally*, we are typically at a loss (no pun intended). Not knowing the probability distribution over $\mathcal{X}$, we cannot compute the expected risk $\ell(H)$ of a hypothesis H as in (1.1), let alone find $H^\star$ as in (1.2). Having access to $\mathcal{X}$ only through finitely many samples, we generally cannot expect to perfectly explain $\mathcal{X}$.

Alternatively, and more realistically, we can try to be *probably approximately correct* (PAC). This means the following: given any tolerances $\delta, \varepsilon > 0$, we can produce with probability at least $1 - \delta$ a hypothesis $\tilde{H} \in \mathcal{H}$ such that

$$\ell(\tilde{H}) \leq \inf_{H\in\mathcal{H}} \ell(H) + \varepsilon. \tag{1.3}$$

This means that with high probability, we approximately solve the optimization problem (1.2). Here, the probability is over the joint probability distribution of the individual samples. So formally $\tilde{H}$ is a (hypothesis-valued) random variable. If the algorithm producing $\tilde{H}$ is randomized, the random variable additionally depends on the random choices made by the algorithm.

1.3.1 Running example: Learning a halfplane

Since the biased coin flip is a toy example not worth an abstract framework, we also want to introduce a "real" learning problem here. Let $\mathcal{X} = \mathbb{R}^2 \times \{0, 1\}$. The source $\mathcal{X}$ comes with an unknown halfplane $H^\star$. A halfplane is a set of the form $H = \{(x_1, x_2) \in \mathbb{R}^2 : ax_1 + bx_2 \leq c\}$ for given parameters $a, b, c \in \mathbb{R}$. We can think of a halfplane as a classifier for points in $\mathbb{R}^2$. One class consists of the points in the halfplane, the other class of the points in the halfplane's complement; see Figure 1.1. Our goal is to learn the unknown classifier $H^\star$, or at least get close to it. The hypothesis class $\mathcal{H}$ is the set of all halfplanes.

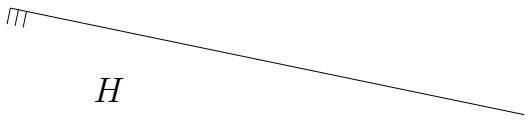


Figure 1.1: A halfplane in $\mathbb{R}^2$

Every sample $X = (\mathbf{x}, y) \in \mathcal{X}$ tells us whether the sampled point $\mathbf{x}$ is in the unknown halfplane or not. Concretely, $y = \mathbb{I}(\{\mathbf{x} \in H^\star\})$, where $\mathbb{I}$ denotes the indicator function of an event.

The probability distribution of the source $\mathcal{X}$ is therefore in fact a distribution over $\mathbb{R}^2$ from which we can sample points $\mathbf{x}$, and the *label* $y = \mathbb{I}(\{\mathbf{x} \in H^\star\}) \in \{0, 1\}$ is a dependent variable. This is the scenario of *supervised learning* where we obtain labeled training samples from the data source.

Our loss function ℓ is very simple in this example: For a given halfplane H and given sample $X = (\mathbf{x}, y)$, the value of $\ell(H, X)$ tells us whether $\mathbf{x}$ is misclassified by H (loss 1) or not (loss 0). This is known as the 0-1-loss and

is formally defined as

$$\ell(H, X) = \mathbb{I}(\{\mathbb{I}(\{\mathbf{x} \in H\}) \neq y\}) = \mathbb{I}(\mathbf{x} \in H \oplus H^\star), \tag{1.4}$$

where $\oplus$ denotes symmetric difference.

The expected risk $\ell(H)$ of a halfplane is then the probability that H misclassifies a point $\mathbf{x}$ randomly sampled from $\mathcal{X}$:

$$\ell(H) = \mathrm{prob}(H \oplus H^\star),$$

Halfplane $H^\star$ solves the expected risk minimization problem (1.1), with value $\ell(H^\star) = 0$. Through finitely many samples, we cannot expect to fully nail down $H^\star$, but we can hope to find a PAC halfplane $\tilde{H}$ as in (1.3), meaning that with high probability, the measure of misclassified points is small; see Figure 1.2.

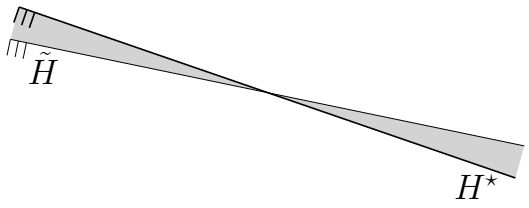


Figure 1.2: An almost optimal halfplane in $\mathbb{R}^2$; the gray region $\tilde{H} \oplus H^\star$ of misclasified points has small measure.

1.4 Empirical risk minimization

Empirical risk minimization is arguably the most prominent "customer" of optimization in Data Science. As outlined in the previous Section, we would like to solve the expected risk minimization problem (1.1) or its approximate version (1.3). But the crucial data we need for that, namely the expected risks $\ell(H), H \in \mathcal{H}$ are not available to us. But what we can do is draw n independent samples $X_1, X_2, \ldots, X_n$ from $\mathcal{X}$ which form our *training data*. Given these data, we compute the *empirical risk*

$$\ell_n(H) = \frac{1}{n} \sum_{i=1}^{n} \ell(H, X_i) \tag{1.5}$$

of hypothesis H. Note that $\ell_n(H)$ is a random variable. In probability, the sequence $(\ell_n(H))_{n\in\mathbb{N}}$ converges to $\ell(H)$. We know this from rolling a dice n times: as $n \to \infty$, the average number of pips converges to its expected value 3.5.

Formally, convergence in probability means the following.

Lemma 1.1 ((Weak) law of large numbers)**.** *Let $H \in \mathcal{H}$ be a hypothesis. For any $\delta, \epsilon > 0$, there exists $n_0 \in \mathbb{N}$ such that for $n \geq n_0$,*

$$|\ell_n(H) - \ell(H)| \leq \varepsilon \tag{1.6}$$

with probability at least $1 - \delta$.

In the language of the previous section, the empirical risk of a hypothesis is a PAC estimate of its expected risk. This motivates *empirical risk minimization* as a proxy for expected risk minimization (1.3). For $n \in \mathbb{N}$ and given training data $X_1, X_2, \ldots, X_n$ from $\mathcal{X}$, empiricial risk minimization attempts to produce a hypothesis $\tilde{H}_n$ such that

$$\ell_n(\tilde{H}_n) \leq \inf_{H\in\mathcal{H}} \ell_n(H) + \varepsilon. \tag{1.7}$$

This may still be a hard problem, but at least, we have all the data that we need in order to solve it. This suggests an algorithmic view of empirical risk minimization as a mapping from input to output.

Input:	training data $X_1, X_2, \ldots, X_n$
Output:	an almost optimal hypothesis $\tilde{H}_n$ as in (1.7)

The ouput $\tilde{H}_n$ is a (hypothesis-valued) random variable (possibly also depending on random choices made by the algorithm that computes $\tilde{H}_n$, but this does not matter for our discussion).

In the coin flip example where a hypothesis $H \in [0, 1]$ is a candidate for the bias $p^\star$, and the $X_i \in \{0, 1\}$ are biased coin flips, we have used loss function $\ell(H, X) = (X - H)^2$. Hence

$$\ell_n(H) = \frac{1}{n}\sum_{i=1}^{n}(X_i - H)^2.$$

Highschool calculus shows that this is (exactly and unsurprisingly) minimized by

$$\tilde{H}_n = \frac{1}{n} \sum_{i=1}^{n} X_i,$$

the relative frequency of heads in a sequence of n biased coin flips.

In our running example from Section 1.3.1, $X_1, X_2, \ldots, X_n$ are points in $\mathbb{R}^2$, labeled with either 1 or 0, signifying whether the point is in the unknown halfplane or not. For $n = 9$, we may therefore see a picture as in Figure 1.3 (left).

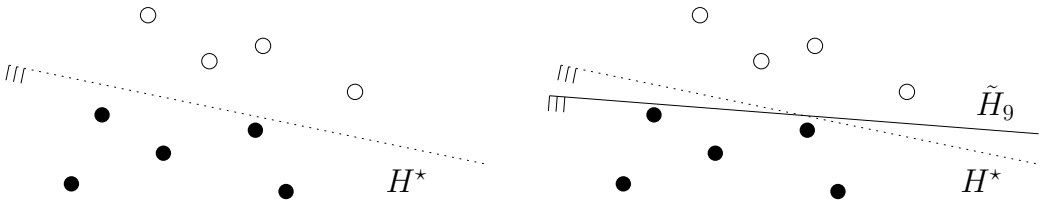


Figure 1.3: Left: training data (filled circles are the ones in the unknown halfplane $H^\star$); Right: a halfplane $\tilde{H}_9$ with minimum empirical risk $\ell_9(\tilde{H}_9) = 0$.

In this case, empiricial risk minimization is easy to do. We know that the training samples with label 1 are separated from the ones with label 0 by the line bounding the unknown halfplane $H^\star$. Hence, we can select *any* separating line and achieve optimal empirical risk 0. Such a separating line can efficiently be found through linear programming. With loss functions other than the 0-1-loss, separating hyperplanes may differ in empirical risk and it may matter which one we choose; here, they are all the same.

1.5 Empirical versus expected risk

Empirical risk minimization produces an (approximate) empirical risk minimizer $\tilde{H}_n$ according to (1.7). In an ideal world, $\tilde{H}_n$ is at the same time PAC for the minimum expected risk (as $n \to \infty$). Let us make it crystal-clear again what this formally means: $\tilde{H}_n$ is a random variable (depending on the training data $X_1, X_2, \ldots, X_n$), and we want that this random variable

satisfies (1.3):

$$\ell(\tilde{H}_n) \leq \inf_{H \in \mathcal{H}} \ell(H) + \varepsilon$$

with probability at least $1 - \delta$.

In the coin flip example, we have seen that $\tilde{H}_n$ is the relative frequency of heads in a sequence $X_1, X_2, \ldots, X_n$ of n biased coin flips, and using the formula for $\ell(H)$ in Exercise 1 (i), we compute

$$\ell(\tilde{H}_n) = \ell(p^\star) + \left(p^\star - \frac{1}{n} \sum_{i=1}^{n} X_i \right)^2 .$$

Now the Chernoff bound ensures that for n sufficiently large, the expected risk of $\tilde{H}_n$ is a good approximation of the minimum expected risk, with high probability.

In general, the law of large numbers (1.6) seems to ensure that we are living in an ideal world, just with adapted tolerances. Indeed, let $\tilde{H}$ be some approximately optimal hypothesis w.r.t. expected risk, meaning that $\ell(\tilde{H}) \leq \inf_{H \in \mathcal{H}} \ell(H) + \varepsilon$. Then we get

$$\begin{aligned} \ell(\tilde{H}_n) &\overset{(1.6)}{\leq} \ell_n(\tilde{H}_n) + \varepsilon \\ &\overset{(1.7)}{\leq} \inf_{H \in \mathcal{H}} \ell_n(H) + 2\varepsilon \leq \ell_n(\tilde{H}) + 2\varepsilon \\ &\overset{(1.6)}{\leq} \ell(\tilde{H}) + 3\varepsilon \leq \inf_{H \in \mathcal{H}} \ell(H) + 4\varepsilon. \end{aligned} \tag{1.8}$$

Here, the inequalites using (1.6) hold with probability at least $1 - \delta$ each, while the other ones are certain, so the whole chain of inequalities holds with probability at least $1 - 2\delta$.

If you think that this derivation was complete nonsense, you are right.

But where exactly is the problem? It's in the first inequality, all the other ones are correct.

The problem is that we cannot apply the law of large numbers to a *data-dependent* hypothesis $\tilde{H}_n$. In (1.6), we need to fix a hypothesis and can then argue that its empirical risk converges in probability to its expected risk. This is what we do with $\tilde{H}$ in the second inequality using (1.6).

But in the first inequality, $\tilde{H}_n$ depends on the data and was deliberately chosen such that the empirical risk is mimimized for the given data. As

we show in Section 1.5.1 below, this may lead to the empiricial risk being much smaller than the expected risk, so that the crucial first inequality does not hold.

Before doing so, let us establish a sufficient condition (a *uniform* version of the law of large numbers) under which the above derivation is sound. It does *not* hold in general, but if it holds, we are indeed in an ideal world: minimizing the empirical risk also minimizes the expected risk.

Theorem 1.2. *Suppose that for any $\delta, \epsilon > 0$, there exists $n_0 \in \mathbb{N}$ such that for $n \geq n_0$,*

$$\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon \tag{1.9}$$

with probability at least $1 - \delta$. Then, for $n \geq n_0$, an approximate empirical risk minimizer $\tilde{H}_n$ as in (1.7) is PAC for expected risk minimization, meaning that it satisfies

$$\ell(\tilde{H}_n) \leq \inf_{H \in \mathcal{H}} \ell(H) + 3\varepsilon$$

with probability at least $1 - \delta$.

Proof. By (1.9),

$$|\ell_n(\tilde{H}_n) - \ell(\tilde{H}_n)| \leq \sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon, \tag{1.10}$$

with probability at least $1 - \delta$. Then we get

$$\begin{aligned} \ell(\tilde{H}_n) &\overset{(1.10)}{\leq} \ell_n(\tilde{H}_n) + \varepsilon \\ &\overset{(1.7)}{\leq} \inf_{H \in \mathcal{H}} \ell_n(H) + 2\varepsilon \\ &\overset{(1.10)}{\leq} \inf_{H \in \mathcal{H}} \ell(H) + 3\varepsilon \end{aligned} \tag{1.11}$$

with probability at least $1 - \delta$. □

1.5.1 A counterexample

We provide a simple (artificial) example to show that the inequality $\ell(\tilde{H}_n) \leq \ell_n(\tilde{H}_n) + \varepsilon$ in our false derivation (1.8) can in general not be achieved. As a consequence, the uniform law of large numbers (1.9) also fails in this example.

Let $\mathcal{X} = [0,1]$, equipped with the uniform distribution. The set of hypotheses $\mathcal{H}$ consists of certain subsets of $\mathcal{X}$, and for $H \in \mathcal{H}, X \in \mathcal{X}$, we let $\ell(H, X) = \mathbb{I}(\{X \notin H\})$.

Concretely, we choose $\mathcal{H}$ such that it satisfies two properties:

(i) every $H \in \mathcal{H}$ has length (Lebesgue measure) $1/2$, so that $\ell(H) = 1/2$ for all $H \in \mathcal{H}$;

(ii) for all training data $X_1, X_2, \ldots, X_n \in \mathcal{X}$, there exists $\tilde{H}_n \in \mathcal{H}$ with $\tilde{H}_n \supseteq \{X_1, X_2, \ldots, X_n\}$, so that $\ell_n(\tilde{H}_n) = 0$.

Hence, $\ell(\tilde{H}_n) \leq \ell_n(\tilde{H}_n) + \varepsilon$ fails for small ε.

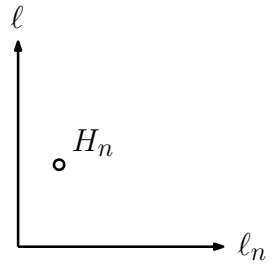
We could choose $\mathcal{H}$ as the collection of all subsets of $[0,1]$ of length $1/2$, but there is also a way of making $\mathcal{H}$ countable; see Exercise 2. $\mathcal{H}$ being infinite is necessary; for every finite set of hypotheses, a simple union bound deduces the uniform law of large numbers (1.9) from the version (1.6) with probability at least $1 - \delta|\mathcal{H}|$, and then Theorem 1.2 implies that the minimum expected risk converges to the minimum empirical risk.

In this example, empirical risk minimization still manages to compute an optimal hypothesis $H^\star$, one that minimizes the expected risk as in (1.2): Since $\ell(H) = 1/2$ for all $H \in \mathcal{H}$, *every* hypothesis is optimal, so there is simply no chance to make a mistake.

But we can easily tweak the example such that an empirical risk minimizer may not be PAC for minimum expected risk. For this, we add to $\mathcal{H}$ sets of lengths $1/4$, say, such that property (ii) above also holds for them. Then there is always a hypothesis $\tilde{H}_n$ such that $\ell_n(\tilde{H}_n) = 0$ and $\ell(\tilde{H}_n) = 3/4$ which is not PAC, since $\inf_{H \in \mathcal{H}} \ell(H) = 1/2$ still.

1.6 The map of learning

Whenever we select a data-dependent hypothesis H_n (by an algorithm that maps the training data $X_1, X_2, \ldots, X_n$ to a hypothesis), we can visualize it as a point in the (ℓ_n, ℓ)-plane:



This is to some extent a cartoon picture. H_n is a random variable, so it does not have fixed empirical and expected risk and therefore not a fixed location in the (ℓ_n, ℓ)-plane. The way we should view this picture is that it tells us where H_n ends up for $n \to \infty$, under the reasonable assumption that the algorithm with high probability "homes in" on some hypothesis as it sees more and more training data. If we adopt this view, the map that emerges is detailed in Figure 1.4. We discuss its different "countries" and "roads" in turn.

Algorithm. This is the computational procedure according to which a hypothesis H_n is obtained from training data $X_1, X_2, \ldots, X_n$. One possible algorithm is (approximate) empirical risk minimization, but there may be other algorithms. The algorithm controls the empirical risk and therefore locates H_n in the ℓ_n-dimension.

Validation. Once a hypothesis H_n has been obtained through training, one also needs to assess its expected risk, i.e. locate H_n in the ℓ-dimension. This is important in order to understand whether the hypothesis actually solves the learning task at hand. Using the weak law of large numbers, $\ell(H_n)$ can be estimated via its empirical risk on *test data*—fresh samples from $\mathcal{X}$ that the algorithm has not seen. This process is called validation. Using fresh samples ensures that H_n is a fixed hypothesis w.r.t. the test data, so the weak law of large numbers actually applies.

Empirical risk minimization. The gray area above the main diagonal (and slightly extending below) in Figure 1.4 is where the empirical risk minimizer $\tilde{H}_n$ ends up. In words, the empirical risk $\ell_n(\tilde{H}_n)$ always underestimates the expected risk $\ell(\tilde{H}_n)$, up to an error that we can make arbitrarily small. We have implicitly shown this before, through the correct

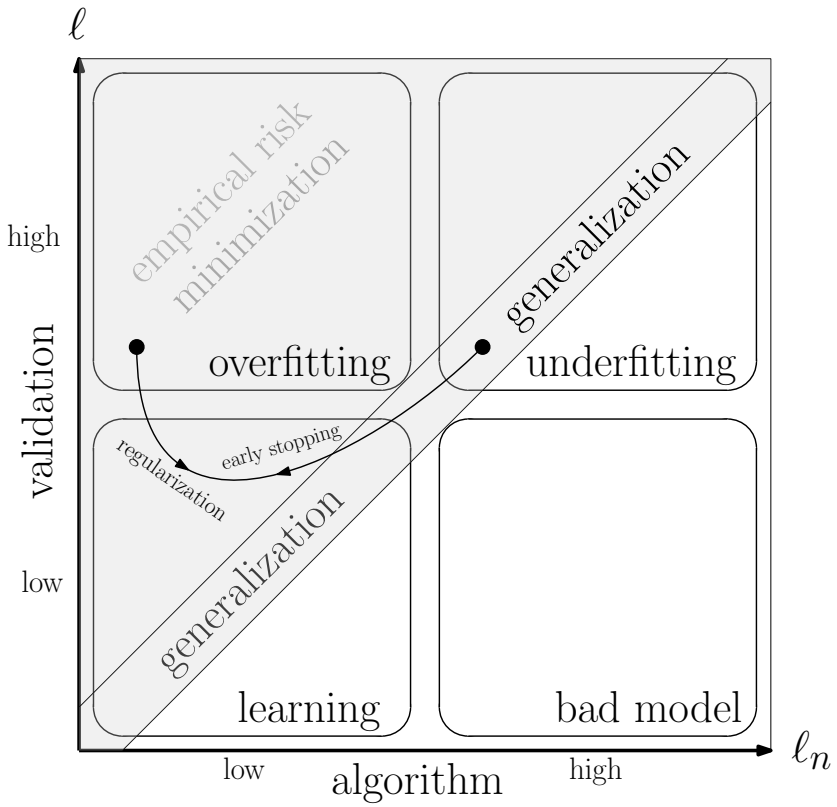


Figure 1.4: The map of learning

part of the chain of inequalities (1.8): we simply need to observe that the last term in this chain can further be bounded by $\ell(\tilde{H}_n) + 3\varepsilon$.

Overfitting. If our learning algorithm returns a hypothesis with low empirical risk but high expected risk, we have a case of overfitting. The explanation quality on the data source is much worse than on the training data. The main cause of overfitting is that our theory (hypothesis class $\mathcal{H}$ and loss function ℓ) is so complex that it allows us to almost perfectly explain any training data. But such a tailored explanation is unlikely to explain unseen data from $\mathcal{X}$. This is like trying to explain when your favorite sports team wins. From observing a few games (training data), one can always find "perfect" explanations of the form "player X wore black socks in exactly the won games." Unless you are superstitious, you may

find this entertaining, but you don't seriously expect such explanations to last for more games.

Underfitting. If the learning algorithm returns a hypothesis with high empirical risk, we cannot even explain the training data. In this case, there is no justified hope to be able to explain unseen data, and we have a case of underfitting. The main cause of underfitttting is that our theory is too simple to capture the nature of the data. In the sports team analogy, if you only have the two hypotheses "the team always wins", or "the team always loses", you will not be able to explain the results of a typical team.

It can in principle happen that a hypothesis with high empirical risk has low expected risk, but then we are in the **bad model** region where we have used a questionable theory.

Learning. If both empirical and expected risk are low, we can make a case that we have learned something. We have invested some effort into explaining the training data, and this explanation is also good for unseen data (maybe not spectacularly so, but sufficient to gain some insights). If you are able to predict the future outcomes of your favorite team's matches more reliably than a biased coin based on the current ranking, say, this is already an achievement.

Generalization. Ideally, the expected risk is close to the empirical risk, and if this happens, we have generalization. This means that the hypothesis explains unseen data equally well as the training data, so there are no surprises in using the explanation for the whole data source. But it does *not* mean that the explanation is good. For example, an easy way to get hold of a generalizing hypothesis is to simply ignore the training data and always return a fixed hypothesis. By the law of large numbers, this generalizes very well, but is not informative at all. In this situation, we are in the upper right generalization region. On the other hand, if a hypothesis has low empirical risk *and* generalizes, learning is implied, and this scenario is the one that learning algorithms are shooting for.

Regularization. In the case that overfitting is observed, a possible remedy is to add a regularization term r to the loss function ℓ with the goal

of "punishing" complex hypotheses. Typically, r has a unique minimizer and nothing to do with the learning problem at hand.

Empirically minimizing $\ell' = \ell + \lambda r$ for a real number $\lambda > 0$ therefore has the effect that we introdue a *bias*, meaning that we deviate more and more from our theory, with the effect that the empirical risk increases. But as the intended consequence, the *variance* (sensitivity to the training data) decreases, and this may reduce the expected risk. For large λ, the new loss function ℓ' is dominated by r, so the empirical risk minimizer eventually becomes uninformative and ends up in the underfitting region. The art is to optimize this *bias-variance tradeoff*: find the sweet spot $\lambda^\star$ that minimizes the expected risk. The curve in Figure 1.4 symbolically depicts the development of the expected risk as we increase λ to move away from the original overfitting hypothesis. This also visualizes the *bias-variance tradeoff* saying that in order to get smaller variance (less dependence on the training data), we need to introduce a higher bias (a worse loss function).

Early stopping. Another way to deal with overfitting is *early stopping*. Typically we are searching for the empirical risk minimizer using an iterative method such as *gradient descent* (Chapter 3). In each step, we decrease the empirical risk of our candidate hypothesis until we (hopefully) converge to an approximately optimal hypothesis $\tilde{H}_n$. We have seen that if $\tilde{H}_n$ overfits, moving away from it through regularization may be beneficial. This can also be achieved by not even getting to $\tilde{H}_n$ in the first place, via early stopping of the stepwise optimization algorithm. How early we stop is a choice we have to make, similar to choosing the parameter λ in regularization. Symbolically, you can think of regularization and early stopping as exploring the same bias-variance tradeoff curve in Figure 1.4, but in opposite directions.

1.6.1 What are the guarantees?

The important takeaway at this point is the following: Although optimization (in the form of empirical risk minimization) is widely and successfully used in practice, it should never be used blindly.

In almost every concrete application, there are potentially many sensible theories (hypothesis classes $\mathcal{H}$ and loss functions ℓ). Which one to pick is often more an art than a science. There are conflicting goals that we have

already touched upon in discussing regularization.

On the one hand, the theory should be informative, meaning that the way in which hypotheses are chosen and evaluated as good or bad is meaningful in the application at hand. For example, the constant loss function $\ell \equiv 0$ ensures perfect generalization and learning according to the map in Figure 1.4 but is obviously useless. On the other hand, the theory should be robust, meaning that empirical risk minimization is a good way of learning about the expected risk.

Finally, we should also be able to perform empirical risk minimization efficiently. This computational aspect favors convex loss functions (see Chapter 2), although these might not be the most informative ones in the given application.

In this course, we do not cover the art of choosing the "right" theory. In subsequent chapters, we focus on how to efficiently solve the optimization problems that arise *after* this artwork has been done.

That said, there *is* a large body of results supporting this artwork. Over the last fifty years, and starting long before machine learning has become mainstream, the field of statistical learning theory has paved the way towards understanding why machine learning works.

In the next section, we sketch a historical cornerstone that started what is now known as the VC (Vapnik-Chervonenkis) theory. Originally developed in the Soviet Union in the late 1960s, it became known to the rest of the world through English translations of the original Russian articles from 1968 and the later one from 1971, which can be considered as the "full paper" containing all proofs [VC71].

The paper establishes a notion of complexity of the hypothesis class $\mathcal{H}$ that is directly and provably related to the success of empirical risk minimization under 0-1-loss. The beauty of this result is that it does not depend on the probability distribution over the data source $\mathcal{X}$. The restriction is that it only works for the 0-1-loss (in supervised learning) which—being nonconvex—is computationally intractable. But the main insight is conceptual: there is a well-defined mathematical way of quantifying our previously informal understanding of a (too) complex theory.

1.7 Vapnik-Chervonenkis theory

Here we are in the (abstract) setting of supervised learning with two classes, generalizing our running example from Section 1.3.1. In this setting, the data source $\mathcal{X}$ is of the form $\mathcal{X} = \mathcal{D} \times \{0, 1\}$, with an unknown probability distribution over data $\mathcal{D}$, and with samples of the form $X = (\mathbf{x}, y)$ where $\mathbf{x} \in \mathcal{D}$ and $y = \mathbb{I}(\{\mathbf{x} \in H^\star\})$ for an unknown subset $H^\star \subseteq \mathcal{D}$. The goal is to learn $H^\star$.

The hypothesis class $\mathcal{H}$ consists of candidate subsets $H \subseteq \mathcal{D}$ and in particular contains $H^\star$. For $H \in \mathcal{H}$ and $X = (\mathbf{x}, y) \in \mathcal{X}$, the 0-1-loss is

$$\ell(H, X) = \mathbb{I}(\{\mathbb{I}(\{\mathbf{x} \in H\}) \neq y\}) = I(\{\mathbf{x} \in H \oplus H^\star\}), \tag{1.12}$$

telling us whether H misclassifies $\mathbf{x}$. We have $\ell(H^\star, X) = 0$ for all X, and hence the minimum expected risk is $\ell(H^\star) = 0$.

In our running example, $\mathcal{H}$ is the set of all halfplanes in $\mathbb{R}^2$, but in the general theory, $\mathcal{H}$ may consist of arbitrary subsets of $\mathcal{D}$.

1.7.1 The growth function

Now we introduce the crucial complexity measure for $\mathcal{H}$, its growth function. Let us fix n training samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ from $\mathcal{D}$ (we ignore their labels y_i as these come from $H^\star$ only and have nothing to do with $\mathcal{H}$). For hypothesis $H \in \mathcal{H}$, we now look at the *cut*

$$H \cap \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

induced by H. In our running example, the cut induced by halfplane H is the set of training samples in the halfplane.

How many different cuts $H \cap \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ can we get as we let H run through $\mathcal{H}$? For sure at most 2^n. We define

$$\mathcal{H} \cap \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} = \{H \cap \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} : H \in \mathcal{H}\}$$

to be the set of cuts that $\mathcal{H}$ induces on $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, and we let

$$\mathcal{H}(n) := \max\{|\mathcal{H} \cap \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}| : \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathcal{D}\} \tag{1.13}$$

be the largest number of cuts that $\mathcal{H}$ induces on any sequence of n training samples. The function $\mathcal{H} : \mathbb{N} \to \mathbb{N}$ (please note and excuse the slight abuse

of notation) is the *growth function* of $\mathcal{H}$. We have already observed that $\mathcal{H}(n) \leq 2^n$. But this may be a (gross) overestimate.

In order to understand the function $\mathcal{H}(n)$ for halfplanes, let us look at small cases first. We first note that $\mathcal{H}(3) = 8$. Indeed, taking as training samples the three corners of a triangle, we can cut out all subsets by halfplanes; see Figure 1.5.

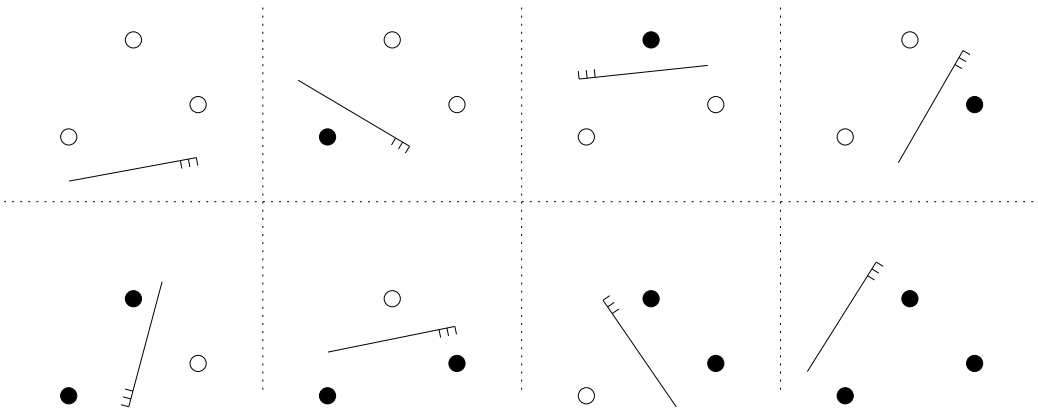


Figure 1.5: Halfplanes can cut a 3-point set in all possible ways.

Next, we claim that $\mathcal{H}(4) = 14$, so it is not possible for halfplanes to cut out all 16 subsets of a 4-point set. A simple argument (omitted) shows that $|\mathcal{H} \cap \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}|$ is maximized when the 4 points are in *general position* (no three on a line). Then there are only two types of configurations, and in each of them, exactly two subsets cannot be cut out by a halfplane; see Figure 1.6.

Figure 1.6: Left: Four points in convex position. Right: One point in the convex hull of the others. In both cases, the set of black points (and its complement) cannot be cut out by a halfplane.

Here is the bound for general n showing that the truth is very far away from the worst-case bound of 2^n.

Example 1.3. *Let $\mathcal{H}$ be the set of all halfplanes in $\mathbb{R}^2$. Then*

$$\mathcal{H}(n) = \mathcal{O}\left(n^2\right).$$

Proof. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^2$. For every halfplane H inducing a nonempty cut, we construct a *canonical* halfplane $H' \subseteq H$ that induces the same cut. To obtain H', we translate H until there is a sample point $\mathbf{x}_i$ on the boundary (we may have $H' = H$). For each i, the halfplanes having $\mathbf{x}_i$ on the boundary induce only $O(n)$ cuts: while rotating a line around $\mathbf{x}_i$, the cut induced by any of its two halfplanes can change only when the line passes another sample point. The bound of $O(n^2)$ follows. We remark that the precise bound is $\mathcal{H}(n) = 2\binom{n}{2} + 2$, see Exercise 3. □

Let us next look at the growth function of the counterexample in Section 1.5.1.

Example 1.4. *Let $\mathcal{H}$ be the collection of all subsets of $[0,1]$ of length $1/2$, $n \in \mathbb{N}$. Then*

$$\mathcal{H}(n) = 2^n.$$

Proof. Let S be a set of distinct samples from $[0,1]$. For every $T \subseteq S$, we construct $H \in \mathcal{H}$ such that $H \cap S = T$. For this, we let U be a set of length $1/2$ disjoint from S and set $H = T \cup U$. □

Exercise 4 asks you to prove that there is also a countable collection $\mathcal{H}$ of subsets of $[0,1]$ with $\mathcal{H}(n) = 2^n$ for all n. No finite collection can have this property, since $\mathcal{H}(n) \leq |\mathcal{H}|$. Indeed, to induce $\mathcal{H}(n)$ different cuts, we need at least $\mathcal{H}(n)$ different hypotheses.

1.7.2 The result

Recall from Theorem 1.2 that in order for an (approximate) empirical risk minimizer to also (approximately) minimize the expected risk, condition (1.9) is sufficient: $\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon$ with probability at least $1 - \delta$. The result is that this can indeed be achieved for any hypothesis class with polynomially bounded growth function, such as the class of halfplanes in $\mathbb{R}^2$ (Example 1.3).

This follows from Theorem 2 of Vapnik and Chervonenkis [VC71] that we state below. This theorem handles the case $H^\star = \emptyset$. Exercise 5 asks you to derive the general case.

Assuming that $H^\star = \emptyset$, the 0-1 loss (1.12) simplifies to

$$\ell(H, X) = \mathbb{I}(\{\mathbf{x} \in H\}).$$

This means that we can think of H as an event whose expected loss is simply its *probability*:

$$\ell(H) = \mathrm{prob}(H).$$

The empirical risk

$$\ell_n(H) = \frac{1}{n}\ell(H, X_i) = \frac{1}{n}\mathbb{I}(\{\mathbf{x}_i \in H\}) =: \mathrm{prob}_n(H)$$

is the *relative frequency* of H, the empirical approximation of $\mathrm{prob}(H)$. The result is therefore already contained in the title of the Vapnik-Chervonenkis paper [VC71]: *On the uniform convergence of relative frequencies of events to their probabilities*.

Theorem 1.5 (Theorem 2 [VC71])**.** *Let $\mathcal{D}$ be a probability space, $\mathcal{H}$ a set of events, $n \in \mathbb{N}$, $\varepsilon > 0$. Then*

$$\sup_{H \in \mathcal{H}} |\mathrm{prob}_n(H) - \mathrm{prob}(H)| > \varepsilon$$

with probability at most

$$4\mathcal{H}(2n) \cdot \exp(-\varepsilon^2 n/8).$$

If the growth function is polynomially bounded ($\mathcal{H}(n) = O(n^k)$ for some constant k), then the error probability in the theorem tends to 0 as $n \to \infty$.

Here is the general version that follows from it (Exercise 5). To be self-contained at this point, we repeat some terminology from before.

Theorem 1.6. *Let $\mathcal{X} = \mathcal{D} \times \{0, 1\}$ be a data source, $\mathcal{H} \subseteq 2^{\mathcal{D}}$ a hypothesis class, $H^\star \in \mathcal{H}$ an unknown ground truth classifier.*

For $H \in \mathcal{H}$ and $X = (\mathbf{x}, y) \in \mathcal{X}$ with $y = \mathbb{I}(\{\mathbf{x} \in H^\star\})$, let

$$\ell(H, X) = \mathbb{I}(\{\mathbb{I}(\{\mathbf{x} \in H\}) \neq y\}) = I(\{\mathbf{x} \in H \oplus H^\star\})$$

be the 0-1-loss, telling us whether H misclassifies X. Let

$$\begin{aligned} \ell(H) &= \mathbb{E}_{\mathcal{X}}[\ell(H, X)] \text{ and} \\ \ell_n(H) &= \frac{1}{n}\sum_{i=1}^{n} \ell(H, X_i), \quad n \in \mathbb{N}, \end{aligned}$$

be the expected risk and the empirical risk of H, respectively, where $X_1, X_2, \ldots, X_n$ are training data, independently chosen from $\mathcal{X}$ ($\ell_n(H)$ is a random variable). Let $\varepsilon > 0$. Then

$$\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| > \varepsilon$$

with probability at most

$$4\mathcal{H}(2n) \cdot \exp(-\varepsilon^2 n/8).$$

If the growth function is polynomially bounded (as in the case of half-planes) Theorem 1.2 follows: as $n \to \infty$, the minimum empirical risk approximates the minimum expected risk up to any desired accuracy.

What if the growth function is not polynomially bounded? It seems that as long as it grows less than exponentially, we are still fine. But here is a surprising fact: *either* $\mathcal{H}(n) = O(n^k)$ for some constant k, *or* $\mathcal{H}(n) = 2^n$ for all n. So there is nothing between polynomial growth and worst-case growth [VC71, Theorem 1]. In other words, empiricial risk minimization either works perfectly as $n \to \infty$ (in case of a polynomially bounded growth function), or not all (in case of $\mathcal{H}(n) = 2^n$ for all n). By "not at all", we mean that in this case (and under suitable asumptions), distribution-independent PAC learning cannot be achieved [BEHW89]. The literature also presents these results in terms of the *VC-dimension* of $\mathcal{H}$ that is either some finite number k (in which case $\mathcal{H}(n) = O(n^k)$), or it is infinite (and then $\mathcal{H}(n) = 2^n$).

1.8 Distribution-dependent guarantees

Often, we know (or assume) something about the probability distribution underlying our data source, and this additional information can lead to provable guarantees for empirical risk minimization that we otherwise wouldn't get. Let us consider *linear regression with fixed design and centered noise* as an example for this scenario.

We have fixed ground truth vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d, n \geq d$ (this is the design). We assume that the $\mathbf{x}_i$ span $\mathbb{R}^d$. The data source is $\mathcal{Y} = \mathbb{R}^n$ and comes with an unknown vector $\beta^\star \in \mathbb{R}^d$. A sample $\mathbf{y} \in \mathcal{Y}$ has entries $y_i = \mathbf{x}_i^\top \beta^\star + w_i$, where the w_i are independent noise terms with expectation 0 and variance σ^2 each.

This means, there is a ground truth linear function $\mathbf{x}_i \mapsto \mathbf{x}_i^\top \beta^\star$ that we would like to learn, but the function values y_i that we get are corrupted with centered noise.

The goal is to learn $\beta^\star$, using just *one* sample $\mathbf{y} \in \mathcal{Y}$ (which already provides n values). As hypothesis class, we therefore use $\mathcal{H} = \mathbb{R}^d$.

If $\sigma^2 = 0$ meaning that there is no noise, we can simply compute $\beta^\star$ from the design by solving the following system of equations in d unknonws $\beta_1, \ldots, \beta_d$.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{pmatrix}.$$

In the presence of noise, we will not be able to nail down $\beta^\star$ exactly, but we can try to get close to it. A natural loss function $\ell : \mathcal{H} \times \mathcal{Y} \to \mathbb{R}$ is

$$\ell(\beta, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \beta)^2,$$

telling us by how much (in squared Euclidean norm) β fails to explain the observed values $\mathbf{y}$. The expected risk is

$$\ell(\beta) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^\top (\beta^\star - \beta))^2 + \sigma^2. \tag{1.14}$$

Indeed, for each i, we have

$$\begin{aligned} \mathbb{E}_{\mathcal{Y}}[(y_i - \mathbf{x}_i^\top \beta)^2] &= \mathbb{E}_{\mathcal{Y}} \left[(\mathbf{x}_i^\top \beta^\star + w_i - \mathbf{x}_i^\top \beta)^2 \right] \\ &= \mathbb{E}_{\mathcal{Y}} \left[(\mathbf{x}_i^\top (\beta^\star - \beta))^2 + 2 w_i \mathbf{x}_i^\top (\beta^\star - \beta) + w_i^2 \right] \\ &= (\mathbf{x}_i^\top (\beta^\star - \beta))^2 + \sigma^2, \end{aligned}$$

using that $\mathbb{E}[w_i] = 0$ and $\mathbb{E}[w_i^2] = \mathrm{Var}[w_i] = \sigma^2$.

This means that an expected risk of σ^2 is unavoidable but can also be attained by choosing $\beta = \beta^\star$. The following result quantifies how good empirical risk minimization performs in this scenario. A proof can be found in the lecture notes of Rigollet and Hütter on High Dimensional Statistics.[1]

Theorem 1.7. *Given a sample $\mathbf{y} \in \mathbb{R}^n$, let $\tilde{\beta} = \tilde{\beta}_1$ minimize the empirical risk*

$$\ell_1(\beta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \beta)^2.$$

Then for any $\delta > 0$,

$$\frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^\top (\beta^\star - \tilde{\beta}))^2 = O\left(\sigma^2 \cdot \frac{d + \log(1/\delta)}{n}\right),$$

with probability at least $1 - \delta$. This further implies

$$\ell(\tilde{\beta}_n) \overset{(1.14)}{=} \ell(\beta^\star) + O\left(\sigma^2 \cdot \frac{d + \log(1/\delta)}{n}\right),$$

with probability at least $1 - \delta$.

In this case, empirical risk minimization is easy to do analytically by solving a *least squares* problem. We will get back to this in Section 2.4.2.

1.9 Worst-case versus average-case complexity

We have seen that empirical risk minimization can be successful in settings where we have no information about the probability distribution over our data source $\mathcal{X}$ (Section 1.7). But we may also have distribution-dependent guarantees as in the previous Section 1.8. Even in settings of the first kind, the training data are independent samples from the distribution in question, and not just any data. This has implications when we use an optimization algorithm to perform empirical risk minimization. In fact, what we care about is not the worst-case performance of the algorithm but the *average case performance*.

[1] `https://klein.mit.edu/~rigollet/PDFs/RigNotes17.pdf`, Theorem 2.2

The classical measure of algorithm performance is its *worst-case complexity*, the function that maps n to the *maximum* runtime of the algorithm over all possible inputs of size n. For example, the worst case complexity of (deterministic) Quicksort for sorting n numbers is $\Theta(n^2)$. But if the input numbers are independent samples from the same distribution, they come in random order; therefore, the *average-case complexity* of Quicksort is much better, namely $\mathcal{O}(n \log n)$. The average case complexity is the function that maps n to the *expected* runtime of the algorithm, taken over its input distribution.

Still, many sources that describe and analyze (optimization) algorithms in Data Science do this in terms of worst-case complexity. These lecture notes are not an exception. The main reason is that it's in most cases very difficult to understand the average-case complexity. There are two problems.

Lack of Knowledge. The first problem occurs if we have no information about the probability distribution over our data source $\mathcal{X}$. In case of Quicksort, we can in this case still argue that the average-case complexity is better than the worst-case complexity, but this is not a "Data Science phenomenon". If the only thing that we need towards an improved average-case complexity is that the input comes in random order, then we can artificially enforce this for *any* input, by simply permuting the input randomly before feeding it to the algorithm. If we do this in case of Quicksort, we obtain, *randomized* Quicksort whose maximum *expected* runtime on n numbers is $\mathcal{O}(n \log n)$, leading to an optimal worst case bound for the expected performance.

True "Data Science" analyses of the average-case complexity typically require some understanding of the data source $\mathcal{X}$ that we do not have.

Too specific knowledge. If we know (or assume) the distribution over the data source $\mathcal{X}$, we may be able to actually compute the average-case complexity of the algorithm *over that specific distribution*. But this typically doesn't give us results for other distributions that may occur in other applications. While there is only one worst-case complexity of an algorithm, there are infinitely many average-case complexities, and it's typically already an arduous task to analyze one of them. Unless the result covers an important distribution (or even an important family of distributions), we

have hardly more than an ad-hoc result for one particular application.

Worst-case complexity may be pessimistic in any given concrete application, but it does provide a valid upper bound for the runtime in *all* applications, via *one* proof. This makes worst-case complexity an attrative complexity measure from a theoretical point of view. Having said this, one should always be aware that the resulting bounds may be pessimistic (sometimes to the point of being useless), and that even as a theoretician, one should strive for better results that ideally cover a number of relevant applications.

1.10 The story of the simplex method

The simplex method for linear programming is probably the optimization algorithm whose complexity has puzzled (and is still puzzling) researchers the most since its invention in the 1940s. As the term simplex *method* suggests, it is actually a family of algorithms, with members differing in their *pivot rules* according to which they decide how to make progress on the way when there are several choices. In this section, we provide a historical overview with a focus on the computational complexity of the simplex method. The tension between worst-case and average-case complexity is present from the very beginning and has ultimately led to the development of smoothed complexity which in some sense combines the best of both worlds.

1.10.1 Initial wandering

In his seminal textbook from 1963, George Dantzig, the inventor of the simplex method, is setting the stage [Dan16, p. 160]:

> While the simplex method appears a natural one to try in the n-dimensional space of the variables, it might be expected, *a priori*, to be inefficient, as there could be considerable wandering ... before an optimal extreme point is reached.

Here, Dantzig is alluding to the fact that in 1963, no theoretical results are known to rule out the pessimistic scenario of "considerable wandering" and bad runtime. He then continues:

> However, empirical evidence with thousands of practical problems indicates that the number of iterations is usually close to the number of basic variables in the final set which were not present in the initial set.

As there are n (nonnegative) variables, this is stating that the runtime in practice is $O(n)$. With m denoting the number of constraints, Dantzig is hinting at the possibility of a theoretical average-case analysis:

> Some believe that for a randomly chosen problem with fixed m, the number of iterations grows in proportion to n.

Dantzig does not further elaborate on what he means by a "randomly chosen" linear program.

In 1969, David Gale publishes an article in the *American Mathematical Monthly* where he presents a variant of the simplex method for solving a system of a linear inequalities that he considers to be a more fundamental problem than linear programming itself (which asks for an optimal solution to a system of linear inequalities) [Gal69]. After summarizing the (un)known complexity results, he concludes:

> Thus there is a large and embarassing gap between what has been observed and what has been proved. This gap stood as a challenge to workers in the field for twenty years now and remains, in my opinion, the principal open question in the theory of linear computation.

1.10.2 Worst-case complexity

Three years later, Victor Klee and George J. Minty make progress on this question [KM72], but not in the desired direction of proving a theoretical bound that matches the observations. Instead, they prove that the worst-case complexity of the simplex method (with Dantzig's original pivot rule) is exponential: there are linear programs (the *Klee-Minty cubes*) in n nonnegative variables that require the method to perform 2^n iterations. But Klee and Minty also clearly point out the limitations of their research:

> On the other hand, our results may not be especially significant for the practical aspect of linear programming (see the final section for comments on this point).

Despite these limitations, their results creates quite some excitement in the community; a sequence of papers ensues in which many other pivot rules are shown to require an exponential number of iterations as well. While each of these papers exhibits a different construction, they all seem to follow a similar approach; Manfred Padberg calls this the period of *worstcasitis* [Pad95]. Only later, Nina Amenta and Günter Ziegler will show that the "similar approach" intuition can be formalized, and that all known constructions are in fact special cases of a general *deformed product* construction [AZ99]. One particular pivot rule, developed by Norman Zadeh in 1980 with the goal of defeating all "similar approaches", will eventually be shown to also require exponentially many iterations; but this will happen only in 2011, via an indeed quite different approach. While this involves money and nudity, it is for the purpose of our discussion a side story, so we refer the interested reader to Günter Ziegler's blog entry.[2]

1.10.3 Average-case complexity

The first substantial average-case analysis of the simplex method is performed by Karl Heinz Borgwardt in a sequence of results that are summarized in his book from 1987 [Bor87]. By "substantial" we mean an analysis that goes beyond easy and artificial cases. In the preface, Borgwardt writes:

> The subject and purpose of this book is to explain the great efficiency in practice by assuming certain distributions on the "real-world"-problems. Other stochastic models are realistic as well and so this analysis should be considered as one of many possibilities.

In Borgwardt's distribution, the origin is assumed to satisfy all the constraints; the (normal vectors of the) constraints as well as the (vector defining the) objective function are rotationally symmetric random vectors. Special cases of this setting occur if the constraints and objective function are chosen from a multivariate normal distribution, or from a ball (or sphere) centered at the origin. Borgwardt proves that the number of

[2] `https://gilkalai.wordpress.com/2011/01/20/gunter-ziegler-1000-from-beverly-hills-for-a-math-problem-ipam-remote-blogging/`

iterations of the simplex method with the *shadow-vertex* pivot rule is polynomial in expectation over the distribution. The analysis is very technical and involved; on a high level, it boils down to proving that the shadow (projection onto a two-dimensional plane) of a random linear program has in expectation only a small number of vertices. As these are exactly the ones that the shadow-vertex pivot rule visits, polynomial runtime on average follows.

Borgwardt's distribution may be what Dantzig had in mind when he was talking about "randomly chosen" linear programs. On the other hand, linear programs observed in practice are typically not random but highly structured. If they follow any distribution at all, it is certainly not the one assumed by Borgwardt. Therefore, Borgwardt's average-case analysis does not offer a full explanation for the efficiency of the simplex method in practice. But it is still an important step forward as it shows that there is a natural (although practically not very relevant) distribution over linear programs on which the simplex method is fast on average. In Section 0.9, Borgwardt also speculates what the right "real-world"-model is and writes the following:

> This is a philosophical question and nobody can answer it satisfactorily. But one should discuss the ideas, conjectures and experiences of practical and theoretical experts of linear programming.

1.10.4 Smoothed complexity

In 2004, and bluntly ignoring Borgwardt's "nobody can", Daniel Spielman and Shang-Hua Teng finally provide a "real-world"-model that is versatile enough to encompass many applications and still allows to prove that the simplex method is fast on average over a random linear program chosen according to the model [ST04]. The *smoothed analysis* they suggest is a hybrid between worst-case and average-case analyis. The high-level description of Spielman and Teng is crisp:

> Worst-case analysis can improperly suggest that an algorithm will perform poorly by examining its performance under the most contrived circumstances. ...

> ...However, average-case analysis may be unconvincing as the inputs encountered in many application domains may bear little resemblance to the random inputs that dominate the analysis. ...
>
> ...In smoothed analysis, we measure the performance of an algorithm under slight random perturbations of arbitrary inputs.

Following Spielman and Teng [ST04], we define the three complexity measures formally. Let $C_A(X)$ be the runtime of algorithm A on input $X \sim \mathcal{X}$. Here $\mathcal{X}$ is again some data source. If A is randomized, we consider the expected runtime. We let $\mathcal{X}_n$ denote the set of all inputs of encoding size n. A has *worst-case complexity* $f(n)$ if

$$\max_{X \sim \mathcal{X}_n} C_A(X) = f(n).$$

The maximum may be achieved in "the most contrived circumstances" and not in typical circumstances. On the plus side, the worst-case complexity is independent from the distribution over $\mathcal{X}$ that we may not know. So the worst-case complexity can in principle be determined.

Algorithm A has *average-case complexity* $f(n)$ if

$$\mathbb{E}_{X \sim \mathcal{X}_n}[C_A(X)] = f(n).$$

This seems to be exactly what we want: the expected runtime over random data sampled from $\mathcal{X}$. The catch is that—not knowing the distribution over $\mathcal{X}$—we can also not determine the average-case complexity. Therefore, similar to expected loss, this is an idealized measure that we may only hope to approximate. Unlike in empirical risk minimization, the approximation here does not consist of a computation, but a *proof* that is supposed to work for all values of n. Towards this, one makes some (simplifying) assumptions on the distribution that allow to prove a bound the average-case complexity under these assumptions. The consequence of this approach is that $\mathcal{X}$ "may bear little resemblance to the random inputs that dominate the analysis."

Smoothed complexity looks at the worst-case *expected* complexity after injecting random noise into the data. For a real number $\sigma \geq 0$, the algorithm has *smoothed complexity* $f(n, \sigma)$ if

$$\max_{X \sim \mathcal{X}_n} \mathbb{E}_{w_i \sim \mathcal{N}(\mathbf{0}, \sigma \|X\|)} C_A(X + \mathbf{w}) = f(n, \sigma).$$

Some explanations are in order here. We assume that the data are such that we *can* inject arbitrarily small noise. In discrete settings (for example $\mathcal{X} = \{0,1\}^n$ to model n coin flips), this is not the case. Concretely, we assume that the input X can be written as a vector of real numbers, and for all indices i, we add independent centered Gaussian noise w_i to the i-th entry of this vector. The standard deviation of each noise term is proportional to the size $\|X\|$ of the input which may for example be measured by Euclidean norm. The factor of proportionality is σ. It is important to understand that the smoothed complexity is independent of the probability distribution over $\mathcal{X}$.

If $\sigma = 0$, we simply have the worst-case complexity. If σ is large, the noise dominates, and the smoothed complexity becomes meaningless for the application. So the scenario that we are interested in is that of small nonzero σ.

The smoothed complexity is called *polynomial* if $f(n, \sigma)$ is polynomial in n and $1/\sigma$. This allows the runtime to tend to infinity as $\sigma \to 0$, but at a rate that is polynomial in $1/\sigma$. Smoothed complexity is a very natural complexity measure in Data Science where data come from measurements or experiments and are therefore *per se* noisy. In this case, the measurement or experiment can itself be considered as injecting the random noise into (unknown) ground truth data. Smoothed complexity then covers the worst possible ground truth data.

The main technical achievement of Spielman and Teng (earning them a number of prestigious prizes) is to show that the smoothed complexity of the shadow vertex simplex algorithm is polynomial, while its worst-case complexity is known to be exponential. Intuitively, this means that the worst-case linear programs are rare and isolated points in the input space: by slightly perturbing them, we arrive at linear programs that the algorithm can solve in polynomial time. Indeed, the deformed products that serve as worst-case inputs for the shadow vertex and other pivot rules are very sensitive to noise; their geometric features are highly structured in tiny regions of space, with the consequence that this structure completely falls apart under small perturbations.

Smoothed analysis does not help for problems where the worst-case inputs have some volume. Let's say that somewhere in input space, there is a ball of fixed radius only containing (near) worst-case inputs. In this case, injecting small random noise does not speed up the algorithm. Also, the smoothed complexity of an algorithm is usually hard to determine, and

even for discrete algorithms such as the simplex method, smoothed analysis is integral-heavy due to the Gaussian noise terms. When applicable and technically feasible, smoothed analysis is an excellent tool to analyze Data Science algorithms, but this is by far not always the case. The 2009 survey of Spielman and Teng contains a few examples where smoothed analysis works [ST09].

1.10.5 An open end

Coming back to the simplex method for one last time: it is still unknown whether there is a pivot rule under which the simplex method has a polynomial number of iterations in the worst case. While one candidate after the other has been ruled out over the last 50 years, the theoretical possibility of such a "wonder rule" remains. But it will be very hard and probably require new methodology to discover it. As a corollary, such a rule would prove the *polynomial Hirsch conjecture* in the affirmative, something that researchers have tried in vain for decades, with considerable effort. For details, we refer the interested reader to the paper by Santos whose disproof of the original (much stronger) Hirsch conjecture from 1957 is a significant breakthrough [San12].

1.11 The estimation-optimization tradeoff

We recall from Section 1.4 and equation (1.7) that the goal in empirical risk minimization is not to find the absolutely best explanation of the training data but an almost best one $\tilde{H}_n$. This is motivated by the fact that the empirical risk of $\tilde{H}_n$ is anyway only an approximation of its expected risk, the measure we actually care about. As we inevitably lose precision in going from empirical to expected risk, it doesn't help to optimize the empirical risk to a significantly higher precision. Let us call the precision that we lose in going from empirical to expected risk the *estimation error*; the precision we lose in finding only an almost best explanation of the training data is the *optimization* error. The literature discusses a third kind of error, the *appoximation* error that arises when the expected risk minimizer $H^\star$ is not in our hypothesis class $\mathcal{H}$, so that even the best explanation from $\mathcal{H}$ loses some precision as compared to $H^\star$. As we have not discussed this scenario, we will ignore the approximation error here.

In a given application, we may have constraints on how many training data we can afford to sample, and on how much time we can afford to spend on optimization. Constraints on the number of training samples typically come from the fact that samples are expensive to obtain. Indeed, a training sample may come from an actual physical measurement or an experiment that has a significant cost; or it requires human intervention to label a training sample with its correct class. Reducing the cost of human intervention is all that services such as Amazon Mechanical Turk are about. We call this scenario *small-scale learning*.

Constraints on optimization time typically come from large training data. The "good old days" where every algorithm of polynomial runtime was considered efficient are long gone in Data Science. With large data, we typically need *linear-time* or even *sublinear-time* algorithms in order to cope with the data. This is the scenario of *large-scale learning*.

In small-scale learning, it doesn't hurt to go for as small an optimization error as we can. But in *large-scale learning*, we may need to give up on some optimization precision in order to be able to stay within the optimization time budget.

The *estimation-optimization tradeoff* consists in finding the most efficient way of spending the resources under the given constraints. The optimization algorithms that we will analyze in this course support this tradeoff. They are usually stepwise methods, gradually improving a candidate solution. The runtime guarantees that we provide are of the form "on data of this and that type, this and that algorithm is guaranteed to have optimization error at most ε after at most $c(\varepsilon)$ many steps." Here, c is a function that grows as $\varepsilon \to 0$, and c is typically not even defined at $\varepsilon = 0$. Hence, most of our algorithms cannot even be used to "optimize to the end." Bus as this is not needed, our main concern is to bound the growth of c as $\varepsilon \to 0$, and algorithms can signficantly differ in this growth. For example, an algorithm with $c(\varepsilon) = O(\log(1/\varepsilon))$ is preferable over one with $c(\varepsilon) = O(1/\varepsilon)$, and the latter is better than an algorithm with $c(\varepsilon) = O(1/\varepsilon^2)$.

Following up on Section 1.9, we point out that our bounds on $c(\varepsilon)$ will usually be worst-case bounds, and as such, they may be overly pessimistic in a concrete application even if they are tight on contrived input. But given the difficulty of obtaining average-case bounds in more then very specific applications, the worst-case bounds still provide some (and sometimes the only) useful guidance concerning optimization time.

The material of this section is based on (and discussed in much more

detail by) Bottou and Bousquet[BB07]; we refer the interested reader to this paper.

1.12 Further listening

Theoretical results in machine learning, in particular the VC theory (Section 1.7) as well as distribution-dependent guarantees (Section 1.8) are covered in the ETH lectures **263-5300-00L Guarantees for Machine Learning**, **401-2684-00L Mathematics of Machine Learning**, and **263-4508-00L Algorithmic Foundations of Data Science**.

The lecture **252-0526-00L Statistical Learning Theory** advocates and explains an approach to learning that is quite different from empirical risk minimization and yields better results in a number of applications. The method is called *maximum entropy and posterior agreement*.

The lectures **227-0690-11L Large-Scale Convex Optimization** and **263-4400-00L Advanced Graph Algorithms and Optimization** have a significant overlap with Chapters 2 (Theory of Convex Functions) and 3 (Gradient Descent) of this lecture.

The course **401-3901-00L Linear & Combinatorial Optimization** is concerned with discrete optimization and in particular discusses polyedral approaches (linear programming and the simplex method are key ingredients of the polyhedral approach). An introductory course teaching in particular the simplex method (Section 1.10) is **401-0647-00L Introduction to Mathematical Optimization**.

1.13 Exercises

Exercise 1. *Let $\mathcal{X} = \{0, 1\}$ be such that the event $X = 1$ has probability $p^\star$ for $X \sim \mathcal{X}$. We want to model the task of finding $p^\star$ as an expected risk minimization problem.*

For $X \in \mathcal{X}$ and $H \in \mathcal{H} = [0, 1]$, we define $\ell(H, X) = (X - H)^2$. The expected risk of H is $\ell(H) = \mathbb{E}_{\mathcal{X}}[\ell(H, X)]$.

(i) Compute $\ell(H)$ for given $H \in \mathcal{H}$.

(ii) Prove that $p^\star$ is the unique minimizer of the expected risk ℓ, and that the minimum expexted risk is $p^\star(1 - p^\star)$, the variance of the biased coin.

Exercise 2. *Prove that there exists a countable collection $\mathcal{H}$ of subsets $H \subseteq [0,1]$ such that (i) every $H \in \mathcal{H}$ has length $1/2$; (ii) for every finite set $S \subseteq [0,1]$, there is $H \in \mathcal{H}$ with $H \cap S = \emptyset$.*

Exercise 3. *Let $\mathcal{H}$ be the set of all halfplanes in $\mathbb{R}^2$, and let S be a set of $n \geq 1$ points in $\mathbb{R}^2$. Prove that $\mathcal{H}$ cuts S in at most $2\binom{n}{2} + 2$ many different ways, and that there are sets S for which this bound is attained. More precisely, prove that the set $\mathcal{H} \cap S = \{H \cap S : H \in \mathcal{H}\}$ has size at most $2\binom{n}{2} + 2$, with equality if and only if S is in* general position*, meaning that no three points of S are on a common line.*

Exercise 4. *Prove that there exists a countable collection $\mathcal{H}$ of subsets $H \subseteq [0,1]$ with the following property: for every finite set $S \subseteq [0,1]$ and every $T \subseteq S$, there is $H \in \mathcal{H}$ with $H \cap S = T$.*

Exercise 5. *Derive Theorem 1.6 from Theorem 1.5!*

Chapter 2

Theory of Convex Functions

Contents

This chapter develops the basic theory of convex functions that we will need later. Much of the material is also covered in other courses, so we will refer to the literature for standard material and focus more on material that we feel is less standard (but important in our context).

2.1 Mathematical Background

2.1.1 Notation

For vectors in $\mathbb{R}^d$, we use bold font, and for their coordinates normal font, e.g. $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$. $\mathbf{x}_1, \mathbf{x}_2, \ldots$ denotes a sequence of vectors. Vectors are considered as column vectors, unless they are explicitly transposed. So $\mathbf{x}$ is a column vector, and $\mathbf{x}^\top$, its transpose, is a row vector. $\mathbf{x}^\top \mathbf{y}$ is the scalar product $\sum_{i=1}^d x_i y_i$ of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

$\|\mathbf{x}\|$ denotes the Euclidean norm (ℓ_2-norm or 2-norm) of vector $\mathbf{x}$,

$$\|\mathbf{x}\|^2 = \mathbf{x}^\top \mathbf{x} = \sum_{i=1}^d x_i^2.$$

We also use

$$\mathbb{N} = \{1, 2, \ldots\} \text{ and } \mathbb{R}_+ := \{x \in \mathbb{R} : x \geq 0\}$$

to denote the natural and non-negative real numbers, respectively. We are freely using basic notions and material from linear algebra and analysis, such as open and closed sets, vector spaces, matrices, continuity, convergence, limits, triangle inequality, among others.

2.1.2 The Cauchy-Schwarz inequality

Lemma 2.1 (Cauchy-Schwarz inequality)**.** *Let* $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$*. Then*

$$|\mathbf{u}^\top \mathbf{v}| \leq \|\mathbf{u}\| \, \|\mathbf{v}\| \, .$$

The inequality holds beyond the Euclidean norm; all we need is an inner product, and a norm induced by it. But here, we only discuss the Euclidean case.

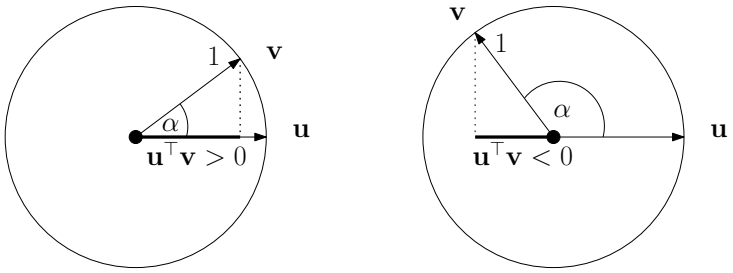
For nonzero vectors, the Cauchy-Schwarz inequality is equivalent to

$$-1 \leq \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\| \, \|\mathbf{v}\|} \leq 1,$$

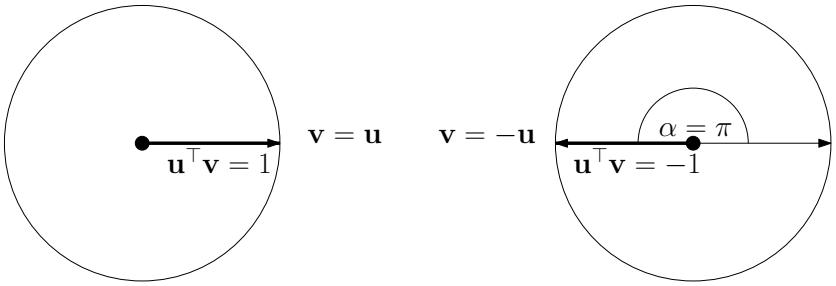
and this fraction can be used to define the angle α between $\mathbf{u}$ and $\mathbf{v}$:

$$\cos(\alpha) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\| \, \|\mathbf{v}\|},$$

where $\alpha \in [0, \pi]$. The following shows the situation for two unit vectors ($\|\mathbf{u}\| = \|\mathbf{v}\| = 1$): The scalar product $\mathbf{u}^\top \mathbf{v}$ is the length of the projection of $\mathbf{v}$ onto $\mathbf{u}$ (which is considered to be negative when $\alpha > \pi/2$). This is just the highschool definition of the cosine.



Hence, equality in Cauchy-Schwarz is obtained if $\alpha = 0$ ($\mathbf{u}$ and $\mathbf{v}$ point into the same direction), or if $\alpha = \pi$ ($\mathbf{u}$ and $\mathbf{v}$ point into opposite directions):



Fix $\mathbf{u} \neq \mathbf{0}$. We see that the vector $\mathbf{v}$ maximizing the scalar product $\mathbf{u}^\top \mathbf{v}$ among all vectors $\mathbf{v}$ of some fixed length is a positive multiple of $\mathbf{u}$, while the scalar product is minimized by a negative multiple of $\mathbf{u}$.

Proof of the Cauchy-Schwarz inequality. There are many proof, but the authors particularly like this one: define the quadratic function

$$f(x) = \sum_{i=1}^{d} (u_i x + v_i)^2 = \left(\sum_{i=1}^{d} u_i^2 \right) x^2 + \left(2 \sum_{i=1}^{d} u_i v_i \right) x + \left(\sum_{i=1}^{d} v_i^2 \right) =: ax^2 + bx + c.$$

We know that $f(x) = ax^2 + bx + c = 0$ has the two solutions

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

This is known as the *Mitternachtsformel* in German-speaking countries, as you are supposed to know it even when you are asleep at midnight.

As by definition, $f(x) \geq 0$ for all x, $f(x) = 0$ has at most one real solution, and this is equivalent to having *discriminant* $b^2 - 4ac \leq 0$. Plugging in the definitions of a, b, c, we get

$$b^2 - 4ac = \left(2\sum_{i=1}^{d} u_i v_i\right)^2 - 4\left(\sum_{i=1}^{d} u_i^2\right)\left(\sum_{i=1}^{d} v_i^2\right) = 4(\mathbf{u}^\top \mathbf{v})^2 - 4\|\mathbf{u}\|^2 \|\mathbf{v}\|^2 \leq 0.$$

Dividing by 4 and taking square roots yields the Cauchy-Schwarz inequality.

2.1.3 The spectral norm

Definition 2.2 (Spectral norm)**.** *Let A be an $(m \times d)$-matrix. Then*

$$\|A\| := \max_{\mathbf{v} \in \mathbb{R}^d, \mathbf{v} \neq 0} \frac{\|A\mathbf{v}\|}{\|\mathbf{v}\|} = \max_{\|\mathbf{v}\|=1} \|A\mathbf{v}\|$$

is the 2-norm (or spectral norm) of A.

In words, the spectral norm is the largest factor by which a vector can be stretched in length under the mapping $\mathbf{v} \to A\mathbf{v}$. Note that as a simple consequence,

$$\|A\mathbf{v}\| \leq \|A\|\|\mathbf{v}\|$$

for all $\mathbf{v}$.

It is good to remind ourselves what a norm is, and why the spectral norm is actually a norm. We need that it is absolutely homegeneous: $\|\lambda A\| = |\lambda|\|A\|$ which follows from the fact that the Euclidean norm is absolutely homegeneous. Then we need the triangle inequality: $\|A + B\| \leq \|A\| + \|B\|$ for two matrices of the same dimensions. Again, this follows from the triangle inequality for the Euclidean norm. Finally, we need that $\|A\| = 0$ implies $A = 0$. Which is true, since for any nonzero matrix A, there is a vector $\mathbf{v}$ such that $A\mathbf{v}$ and hence the Euclidean norm of $A\mathbf{v}$ is nonzero.

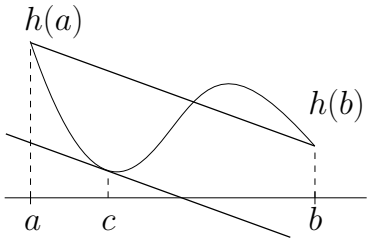
2.1.4 The mean value theorem

We also recall the *mean value theorem* that we will frequently need:

Theorem 2.3 (Mean value theorem)**.** *Let $a < b$ be real numbers, and let $h : [a, b] \to \mathbb{R}$ be a continuous function that is differentiable on (a, b); we denote the derivative by h'. Then there exists $c \in (a, b)$ such that*

$$h'(c) = \frac{h(b) - h(a)}{b - a}.$$

Geometrically, this means the following: We can interpret the value $(h(b)-h(a))/(b-a)$ as the slope of the line through the two points $(a, h(a))$ and $(b, h(b))$. Then the mean value theorem says that between a and b, we find a tangent to the graph of h that has the same slope:



2.1.5 The fundamental theorem of calculus

If a function h is *continuously* differentiable in an interval $[a, b]$, we have another way of expressing $h(b) - h(a)$ in terms of the derivative.

Theorem 2.4 (Fundamental theorem of calculus)**.** *Let $a < b$ be real numbers, and let $h : \mathbf{dom}(h) \to \mathbb{R}$ be a differentiable function on an open domain $\mathbf{dom}(h) \supset [a, b]$, and such that h' is continuous on $[a, b]$. Then*

$$h(b) - h(a) = \int_a^b h'(t)dt.$$

This theorem is the theoretical underpinning of typical definite integral computations in high school. For example, to evaluate $\int_2^4 x^2 dx$, we integrate x^2 (giving us $x^3/3$), and then compute

$$\int_2^4 x^2 dx = \frac{4^3}{3} - \frac{2^3}{3} = \frac{56}{3}.$$

2.1.6 Differentiability

For univariate functions $f : \mathbf{dom}(f) \to \mathbb{R}$ with $\mathbf{dom}(f) \subseteq \mathbb{R}$, differentiability is covered in high school. We will need the concept for multivariate and vector-valued functions $f : \mathbf{dom}(f) \to \mathbb{R}^m$ with $\mathbf{dom}(f) \subseteq \mathbb{R}^d$. Mostly, we deal with the case $m = 1$: real-valued functions in d variables. As we frequently need this material, we include a refresher here.

Definition 2.5. *Let $f : \mathbf{dom}(f) \to \mathbb{R}^m$ where $\mathbf{dom}(f) \subseteq \mathbb{R}^d$. the function f is called* differentiable *at $\mathbf{x}$ in the interior of $\mathbf{dom}(f)$ if there exists an $(m \times d)$-matrix A and an error function $r : \mathbb{R}^d \to \mathbb{R}^m$ defined in some neighborhood of $\mathbf{0} \in \mathbb{R}^d$ such that for all $\mathbf{y}$ in some neighborhood of $\mathbf{x}$,*

$$f(\mathbf{y}) = f(\mathbf{x}) + A(\mathbf{y} - \mathbf{x}) + r(\mathbf{y} - \mathbf{x}),$$

where

$$\lim_{\mathbf{v} \to \mathbf{0}} \frac{\|r(\mathbf{v})\|}{\|\mathbf{v}\|} = \mathbf{0}.$$

It then also follows that the matrix A is unique, and it is called the differential *or* Jacobian *of f at $\mathbf{x}$. We will denote it by $Df(\mathbf{x})$. More precisely, $Df(\mathbf{x})$ is the matrix of* partial derivatives *at the point $\mathbf{x}$,*

$$Df(\mathbf{x})_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}).$$

f is called differentiable *if f is differentiable at all $\mathbf{x} \in \mathbf{dom}(f)$ (which implies that $\mathbf{dom}(f)$ is open).*

Differentiability at $\mathbf{x}$ means that in some neighborhood of $\mathbf{x}$, f is approximated by a (unique) affine function $f(\mathbf{x}) + Df(\mathbf{x})(\mathbf{y} - \mathbf{x})$, up to a sublinear error term. If $m = 1$, $Df(\mathbf{x})$ is a row vector typically denoted by $\nabla f(\mathbf{x})^\top$, where the (column) vector $\nabla f(\mathbf{x})$ is called the *gradient* of f at $\mathbf{x}$. Geometrically, this means that the graph of the affine function $f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x})$ is a *tangent hyperplane* to the graph of f at $(\mathbf{x}, f(\mathbf{x}))$; see Figure 2.1.

It also follows easily that a differentiable function is continuous, see Exercise 6.

Let us do a simple example to illustrate the concept of differentiability.

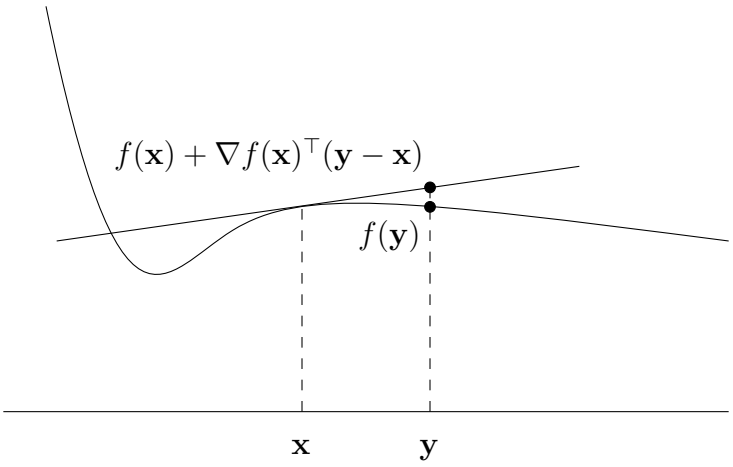


Figure 2.1: If f is differentiable at $\mathbf{x}$, the graph of f is locally (around $\mathbf{x}$) approximated by a tangent hyperplane

Example 2.6. *Consider the function $f(x) = x^2$. We know that its derivative is $f'(x) = 2x$. But why? For fixed x and $y = x + v$, we compute*

$$\begin{aligned} f(y) = (x+v)^2 &= x^2 + 2vx + v^2 \\ &= f(x) + 2x \cdot v + v^2 \\ &= f(x) + A(y-x) + r(y-x), \end{aligned}$$

where $A := 2x, r(y-x) = r(v) := v^2$. We have $\lim_{v\to 0} \frac{|r(v)|}{|v|} = \lim_{v\to 0} |v| = 0$. Hence, $A = 2x$ is indeed the differential (a.k.a. derivative) of f at x.

In computing differentials, the *chain rule* is particularly useful.

Lemma 2.7 (Chain rule)**.** *Let $f : \mathbf{dom}(f) \to \mathbb{R}^m, \mathbf{dom}(f) \subseteq \mathbb{R}^d$ and $g : \mathbf{dom}(g) \to \mathbb{R}^d$. Suppose that $\mathbf{g}$ is differentiable at $\mathbf{x} \in \mathbf{dom}(g)$ and that f is differentiable at $g(\mathbf{x}) \in \mathbf{dom}(f)$. Then $f \circ g$ (the composition of f and g) is differentiable at $\mathbf{x}$, with the differential given by the matrix equation*

$$D(f \circ g)(\mathbf{x}) = Df(g(\mathbf{x}))Dg(\mathbf{x}).$$

Here is an application of the chain rule that we will use frequently. Let $f : \mathbf{dom}(f) \to \mathbb{R}^m$ be a differentiable function with (open) convex domain, and fix $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$. There is an open interval I containing $[0, 1]$ such

that $\mathbf{x} + t(\mathbf{y} - \mathbf{x}) \in \mathbf{dom}(f)$ for all $t \in I$. Define $g : I \to \mathbb{R}^d$ by $g(t) = \mathbf{x} + t(\mathbf{y} - \mathbf{x})$ and set $h = f \circ g$. Thus, $h : I \to \mathbb{R}^m$ with $h(t) = f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$, and for all $t \in I$, we have

$$h'(t) = Dh(t) = Df(g(t))Dg(t) = Df(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x}). \tag{2.1}$$

Since we mostly consider real-valued functions, we will encounter differentials in the form of gradients. For example, if $f(\mathbf{x}) = \mathbf{c}^\top \mathbf{x} = \sum_{j=1}^d c_j x_j$, then $\nabla f(\mathbf{x}) = \mathbf{c}$; and if $f(\mathbf{x}) = \|\mathbf{x}\|^2 = \sum_{j=1}^d x_j^2$, then $\nabla f(\mathbf{x}) = 2\mathbf{x}$.

2.2 Convex sets

Definition 2.8. *A set $C \subseteq \mathbb{R}^d$ is* convex *if for any two points $\mathbf{x}, \mathbf{y} \in C$, the connecting line segment is contained in C. In formulas, if for all $\lambda \in [0, 1]$, $\lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \in C$; see Figure 2.2.*

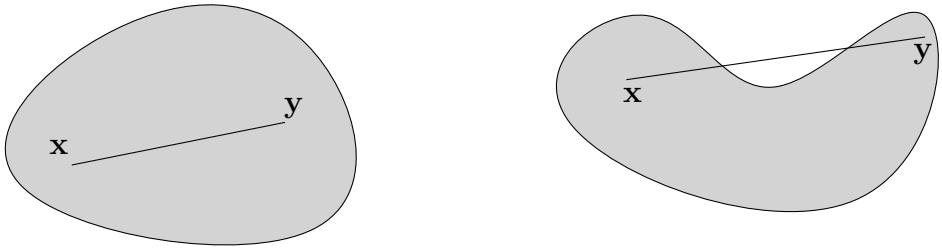


Figure 2.2: A convex set (left) and a non-convex set (right)

Observation 2.9. *Let $C_i, i \in I$ be convex sets, where I is a (possibly infinite) index set. Then $C = \bigcap_{i \in I} C_i$ is a convex set.*

For $d = 1$, convex sets are *intervals*.

2.2.1 The mean value inequality

The mean value inequality can be considered as as generalization of the mean value theorem to multivariate and vector-valued functions over convex sets (a "mean value equality" does not exist in this full generality).

To motivate it, let us consider the univariate and real-valued case first. Let $f : \mathbf{dom}(f) \to R$ be differentiable and suppose that f has bounded derivatives over an interval $X \subseteq \mathbf{dom}(f)$, meaning that for some real number B, we have $|f'(x)| \leq B$ for all $x \in X$. The mean value theorem then gives the *mean value inequality*

$$|f(y) - f(x)| = |f'(c)(y - x)| \leq B|y - x|$$

for all $x, y \in X$ and some in-between c. In other words, f is not only continuous but actually B-*Lipschitz* over X.

Vice versa, suppose that f is B-Lipschitz over a nonempty *open* interval X, then for all $c \in X$,

$$|f'(c)| = |\lim_{\delta \to 0} \frac{f(c + \delta) - f(c)}{\delta}| \leq B,$$

so f has bounded derivatives over X. Hence, over an open interval, Lipschitz functions are exactly the ones with bounded derivative. Even if the interval is not open, bounded derivatives still yield the Lipschitz property, but the other direction may fail. As a trivial example, the Lipschitz condition is always satisfied over a singleton interval $X = \{x\}$, but that does not say anything about the derivative at x. In any case, we need X to be an interval; if X has "holes", the previous arguments break down.

These considerations extend to multivariate and vector-valued functions over *convex* subsets of the domain.

Theorem 2.10. *Let $f : \mathbf{dom}(f) \to \mathbb{R}^m$ be differentiable, $X \subseteq \mathbf{dom}(f)$ a convex set, $B \in \mathbb{R}^+$. If $X \subseteq \mathbf{dom}(f)$ is nonemepty and open, the following two statements are equivalent.*

(i) f is B-Lipschitz, meaning that

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq B \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in X$$

(ii) f has differentials bounded by B (in spectral norm), meaning that

$$\|Df(\mathbf{x})\| \leq B, \quad \forall \mathbf{x} \in X.$$

Moreover, for every (not necessarily open) convex $X \subseteq \mathbf{dom}(f)$, (ii) implies (i), and this is the mean value inequality*.*

Proof. Suppose that f is B-Lipschitz over an open set X. For $\mathbf{v} \in \mathbb{R}^d$, $\mathbf{v} \to \mathbf{0}$, differentiability at $\mathbf{x} \in X$ yields for small $\mathbf{v} \in \mathbb{R}^d$ that $\mathbf{x} + \mathbf{v} \in X$ and therefore

$$B \|\mathbf{v}\| \geq \|f(\mathbf{x} + \mathbf{v}) - f(\mathbf{x})\| = \|Df(\mathbf{x})\mathbf{v} + r(\mathbf{v})\| \geq \|Df(\mathbf{x})\mathbf{v}\| - \|r(\mathbf{v})\|,$$

where $\|r(\mathbf{v})\| / \|\mathbf{v}\| \to 0$, the first inequality uses (i), and the last is the reverse triangle inequality. Rearranging and dividing by $\|\mathbf{v}\|$, we get

$$\frac{\|Df(\mathbf{x})\mathbf{v}\|}{\|\mathbf{v}\|} \leq B + \frac{\|r(\mathbf{v})\|}{\|\mathbf{v}\|}.$$

Let $\mathbf{v}^\star$ be a unit vector such that $\|Df(\mathbf{x})\| = \|Df(\mathbf{x})\mathbf{v}^\star\| / \|\mathbf{v}^\star\|$ and let $\mathbf{v} = t\mathbf{v}^\star$ for $t \to 0$. Then we further get

$$\|Df(\mathbf{x})\| \leq B + \frac{\|r(\mathbf{v})\|}{\|\mathbf{v}\|} \to B,$$

and $\|Df(\mathbf{x})\| \leq B$ follows, so differentials are bounded by B.

For the other direction, suppose that differentials are bounded by B over X (not necessarily open); we proceed as in [FM91].

For fixed $\mathbf{x}, \mathbf{y} \in X \subseteq \mathbf{dom}(f), \mathbf{x} \neq \mathbf{y}$, and $\mathbf{z} \in \mathbb{R}^m$ (to be determined later), we define

$$h(t) = \mathbf{z}^\top f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$$

over $\mathbf{dom}(h) = [0, 1]$, in which case the chain rule yields

$$h'(t) = \mathbf{z}^\top Df(x + t(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x}), \quad t \in (0, 1),$$

see also (2.1). Note that $\mathbf{x} + t(\mathbf{y} - \mathbf{x}) \in X$ for $t \in [0, 1]$ by convexity of X. The mean value theorem guarantees $c \in (0, 1)$ such that $h'(c) = h(1) - h(0)$. Now we compute

$$\begin{aligned}
\left\|\mathbf{z}^\top (f(\mathbf{y}) - f(\mathbf{x}))\right\| &= |h(1) - h(0)| = |h'(c)| \\
&= \mathbf{z}^\top Df(x + c(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x}) \\
&\leq \|\mathbf{z}\| \|Df(x + c(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x})\| \quad \text{(Cauchy-Schwarz)} \\
&\leq \|\mathbf{z}\| \|Df(x + c(\mathbf{y} - \mathbf{x}))\| \|(\mathbf{y} - \mathbf{x})\| \quad \text{(spectral norm)} \\
&\leq B \|\mathbf{z}\| \|(\mathbf{y} - \mathbf{x})\| \quad \text{(bounded differentials).}
\end{aligned}$$

We assume w.l.o.g. that $f(\mathbf{x}) \neq f(\mathbf{y})$, as otherwise, (i) trivially holds; now we set

$$\mathbf{z} = \frac{f(\mathbf{y}) - f(\mathbf{x})}{\|f(\mathbf{y}) - f(\mathbf{x})\|}.$$

With this, the previous inequality reduces to (i), so f is indeed B-Lipschitz over X. □

2.3 Convex functions

We are considering real-valued functions $f : \mathbf{dom}(f) \to \mathbb{R}$, $\mathbf{dom}(f) \subseteq \mathbb{R}^d$.

Definition 2.11 ([BV04, 3.1.1]). *A function $f : \mathbf{dom}(f) \to \mathbb{R}$ is* convex *if (i) $\mathbf{dom}(f)$ is convex and (ii) for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$ and all $\lambda \in [0, 1]$, we have*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}). \tag{2.2}$$

Geometrically, the condition means that the line segment connecting the two points $(\mathbf{x}, f(\mathbf{x})), (\mathbf{y}, f(\mathbf{y})) \in \mathbb{R}^{d+1}$ lies pointwise above the graph of f; see Figure 2.3. (Whenever we say "above", we mean "above or on".) An important special case arises when $f : \mathbb{R}^d \to \mathbb{R}$ is an affine function, i.e. $f(\mathbf{x}) = \mathbf{c}^\top \mathbf{x} + c_0$ for some vector $\mathbf{c} \in \mathbb{R}^d$ and scalar $c_0 \in \mathbb{R}$. In this case, (2.2) is always satisfied with equality, and line segments connecting points on the graph lie pointwise on the graph.

While the graph of f is the set $\{(\mathbf{x}, f(\mathbf{x})) \in \mathbb{R}^{d+1} : \mathbf{x} \in \mathbf{dom}(f)\}$, the *epigraph* (Figure 2.4) is the set of points above the graph,

$$\mathbf{epi}(f) := \{(\mathbf{x}, \alpha) \in \mathbb{R}^{d+1} : \mathbf{x} \in \mathbf{dom}(f), \alpha \geq f(\mathbf{x})\}.$$

Observation 2.12. *f is a convex function if and only if $\mathbf{epi}(f)$ is a convex set.*

Proof. This is easy but let us still do it to illustrate the concepts. Let f be a convex function and consider two points $(\mathbf{x}, \alpha), (\mathbf{y}, \beta) \in \mathbf{epi}(f)$, $\lambda \in [0, 1]$. This means, $f(\mathbf{x}) \leq \alpha$, $f(\mathbf{y}) \leq \beta$, hence by convexity of f,

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \leq \lambda\alpha + (1-\lambda)\beta.$$

Therefore, by definition of the epigraph,

$$\lambda(\mathbf{x}, \alpha) + (1-\lambda)(\mathbf{y}, \beta) = (\lambda\mathbf{x} + (1-\lambda)\mathbf{y}, \lambda\alpha + (1-\lambda)\beta) \in \mathbf{epi}(f),$$

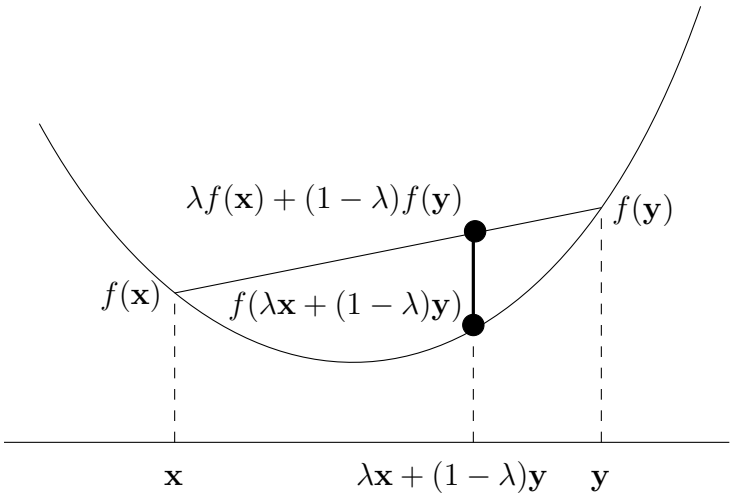


Figure 2.3: A convex function

so epi(f) is a convex set. In the other direction, let epi(f) be a convex set and consider two points $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$, $\lambda \in [0,1]$. By convexity of epi(f), we have

$$\text{epi}(f) \ni \lambda(\mathbf{x}, f(\mathbf{x})) + (1-\lambda)(\mathbf{y}, f(\mathbf{y})) = (\lambda\mathbf{x} + (1-\lambda)\mathbf{y}, \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y})),$$

and this is just a different way of writing (2.2). □

Lemma 2.13 (Jensen's inequality)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbf{dom}(f)$, and $\lambda_1, \ldots, \lambda_m \in \mathbb{R}_+$ such that $\sum_{i=1}^m \lambda_i = 1$. Then*

$$f\left(\sum_{i=1}^m \lambda_i \mathbf{x}_i\right) \leq \sum_{i=1}^m \lambda_i f(\mathbf{x}_i).$$

For $m = 2$, this is (2.2). The proof of the general case is Exercise 7.

Lemma 2.14. *Let f be convex and suppose that $\mathbf{dom}(f)$ is open. Then f is continuous.*

This is not entirely obvious (see Exercise 8) and really needs $\mathbf{dom}(f) \subseteq \mathbb{R}^d$. It becomes false if we consider convex functions over vector spaces of infinite dimension. In fact, in this case, even linear functions (which are in particular convex) may fail to be continuous.

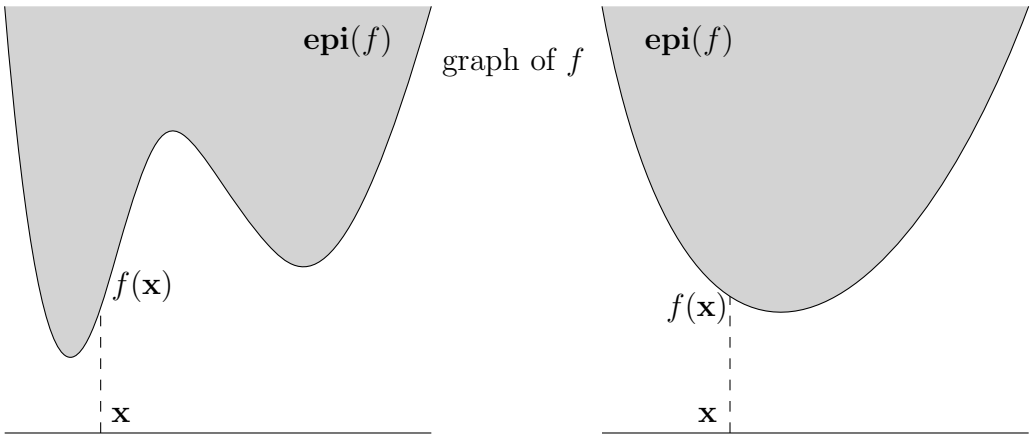


Figure 2.4: Graph and epigraph of a non-convex function (left) and a convex function (right)

Lemma 2.15. *There exists an (infinite dimensional) vector space V and a linear function $f : V \to \mathbb{R}$ such that f is discontinuous at all $\mathbf{v} \in V$.*

Proof. This is a classical example. Let us consider the vector space V of all univariate polynomials; the vector space operations are addition of two polynomials, and multiplication of a polynomial with a scalar. We consider a polynomial such as $3x^5 + 2x^2 + 1$ as a function $x \mapsto 3x^5 + 2x^2 + 1$ over the domain $[-1, 1]$.

The standard norm in a function space such as V is the *supremum norm* $\|\cdot\|_\infty$, defined for any bounded function $h : [-1, 1] \to \mathbb{R}$ via $\|h\|_\infty := \sup_{x\in[-1,1]} |h(x)|$. Polynomials are continuous and as such bounded over $[-1, 1]$.

We now consider the linear function $f : V \to \mathbb{R}$ defined by $f(p) = p'(0)$, the derivative of p at 0. The function f is linear, simply because the derivative is a linear operator. As $\mathbf{dom}(f)$ is the whole space V, $\mathbf{dom}(f)$ is open. We claim that f is discontinuous at 0 (the zero polynomial). Since f is linear, this implies discontinuity at every polynomial $p \in V$. To prove discontinuity at 0, we first observe that $f(0) = 0$ and then show that there are polynomials p of arbitrarily small supremum norm with $f(p) = 1$. Indeed,

for $n, k \in \mathbb{N}, n > 0$, consider the polynomial

$$p_{n,k}(x) = \frac{1}{n}\sum_{i=0}^{k}(-1)^i\frac{(nx)^{2i+1}}{(2i+1)!} = \frac{1}{n}\left(nx - \frac{(nx)^3}{3!} + \frac{(nx)^5}{5!} - \cdots \pm \frac{(nx)^{2k+1}}{(2k+1)!}\right)$$

which—for any fixed n and sufficiently large k—approximates the function

$$s_n(x) = \frac{1}{n}\sin(nx) = \frac{1}{n}\sum_{i=0}^{\infty}(-1)^i\frac{(nx)^{2i+1}}{(2i+1)!}$$

up to any desired precision over the whole interval $[-1, 1]$ (Taylor's theorem with remainder). In formulas, $\|p_{n,k} - s_n\|_\infty \to 0$ as $k \to \infty$. Moreover, $\|s_n\|_\infty \to 0$ as $n \to \infty$. Using the triangle inequality, this implies that $\|p_{n,k}\| \to 0$ as $n, k \to \infty$. On the other hand, $f(p_{n,k}) = p'_{n,k}(0) = 1$ for all n, k. □

2.3.1 First-order characterization of convexity

As an example of a convex function, let us consider $f(x_1, x_2) = x_1^2 + x_2^2$. The graph of f is the *unit paraboloid* in $\mathbb{R}^3$ which looks convex. However, to verify (2.2) directly is somewhat cumbersome. Next, we develop better ways to do this if the function under consideration is differentiable.

Lemma 2.16 ([BV04, 3.1.3])**.** *Suppose that* $\mathbf{dom}(f)$ *is open and that* f *is differentiable; in particular, the* gradient *(vector of partial derivatives)*

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

exists at every point $\mathbf{x} \in \mathbf{dom}(f)$*. Then* f *is convex if and only if* $\mathbf{dom}(f)$ *is convex and*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \tag{2.3}$$

holds for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$*.*

Geometrically, this means that for all $\mathbf{x} \in \mathbf{dom}(f)$, the graph of f lies above its tangent hyperplane at the point $(\mathbf{x}, f(\mathbf{x}))$; see Figure 2.5.

Proof. Suppose that f is convex, meaning that for $t \in (0, 1)$,

$$f(\mathbf{x}+t(\mathbf{y}-\mathbf{x})) = f((1-t)\mathbf{x}+t\mathbf{y}) \leq (1-t)f(\mathbf{x})+tf(\mathbf{y}) = f(\mathbf{x})+t(f(\mathbf{y})-f(\mathbf{x})).$$

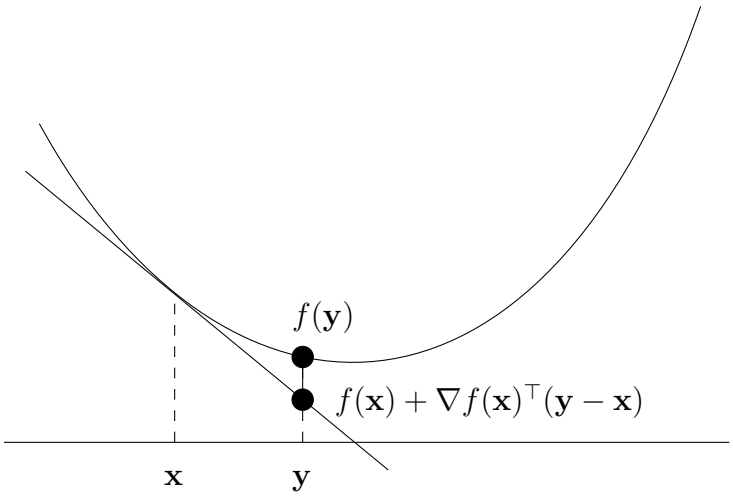


Figure 2.5: First-order characterization of convexity

Dividing by t and using differentiability at $\mathbf{x}$, we get

$$\begin{aligned} f(\mathbf{y}) &\geq f(\mathbf{x}) + \frac{f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - f(\mathbf{x})}{t} \\ &= f(\mathbf{x}) + \frac{\nabla f(\mathbf{x})^\top t(\mathbf{y} - \mathbf{x}) + r(t(\mathbf{y} - \mathbf{x}))}{t} \\ &= f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) + \frac{r(t(\mathbf{y} - \mathbf{x}))}{t}, \end{aligned}$$

where the error term $r(t(\mathbf{y} - \mathbf{x}))/t$ goes to 0 as $t \to 0$. The inequality $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x})$ follows.

Now suppose this inequality holds for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$, let $\lambda \in [0, 1]$, and define $\mathbf{z} := \lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \in \mathbf{dom}(f)$ (by convexity of $\mathbf{dom}(f)$). Then we have

$$\begin{aligned} f(\mathbf{x}) &\geq f(\mathbf{z}) + \nabla f(\mathbf{z})^\top(\mathbf{x} - \mathbf{z}), \\ f(\mathbf{y}) &\geq f(\mathbf{z}) + \nabla f(\mathbf{z})^\top(\mathbf{y} - \mathbf{z}). \end{aligned}$$

After multiplying the first inequality by λ and the second one by $(1 - \lambda)$, the gradient terms cancel in the sum of the two inequalities, and we get

$$\lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}) \geq f(\mathbf{z}) = f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}).$$

This is convexity. □

For $f(x_1, x_2) = x_1^2 + x_2^2$, we have $\nabla f(\mathbf{x}) = (2x_1, 2x_2)$, hence (2.3) boils down to

$$y_1^2 + y_2^2 \geq x_1^2 + x_2^2 + 2x_1(y_1 - x_1) + 2x_2(y_2 - x_2),$$

which after some rearranging of terms is equivalent to

$$(y_1 - x_1)^2 + (y_2 - x_2)^2 \geq 0,$$

hence true. There are relevant convex functions that are not differentiable, see Figure 2.6 for an example. More generally, Exercise 14 asks you to prove that the ℓ_1-norm (or 1-norm) $f(\mathbf{x}) = \|\mathbf{x}\|_1$ is convex.

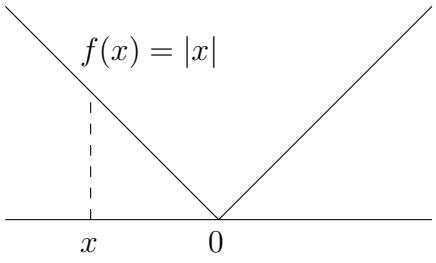


Figure 2.6: A non-differentiable convex function

There is another useful and less standard first-order characterization of convexity that we can easily derive from the standard one above.

Lemma 2.17. *Suppose that* $\mathbf{dom}(f)$ *is open and that* f *is differentiable. Then* f *is convex if and only if* $\mathbf{dom}(f)$ *is convex and*

$$(\nabla f(\mathbf{y}) - \nabla f(\mathbf{x}))^\top (\mathbf{y} - \mathbf{x}) \geq 0 \tag{2.4}$$

holds for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$.

The inequality (2.4) is known as *monotonicity of the gradient*.

Proof. If f is convex, the first-order characterization in Lemma 2.16 yields

$$\begin{aligned} f(\mathbf{y}) &\geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \\ f(\mathbf{x}) &\geq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}), \end{aligned}$$

for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$. After adding up these two inequalities, $f(\mathbf{x}) + f(\mathbf{y})$ appears on both sides and hence cancels, so that we get

$$0 \geq \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) = (\nabla f(\mathbf{y}) - \nabla f(\mathbf{x}))^\top (\mathbf{x} - \mathbf{y}).$$

Multiplying this by -1 yields (2.4).

For the other direction, suppose that monotonicty of the gradient (2.4) holds. Then we in particular have

$$(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}))^\top (t(\mathbf{y} - \mathbf{x})) \geq 0$$

for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$ and $t \in (0, 1)$. Dividing by t, this yields

$$(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}))^\top (\mathbf{y} - \mathbf{x})) \geq 0. \tag{2.5}$$

Fix $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$. For $t \in [0, 1]$, let $h(t) := f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$. In our case where f is real-valued, (2.1) yields $h'(t) = \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))^\top (\mathbf{y} - \mathbf{x}), t \in (0, 1)$. Hence, (2.5) can be rewritten as

$$h'(t) \geq \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \quad t \in (0, 1).$$

By the mean value theorem, there is $c \in (0, 1)$ such that $h'(c) = h(1) - h(0)$. Then

$$\begin{aligned} f(\mathbf{y}) = h(1) = h(0) + h'(c) &= f(\mathbf{x}) + h'(c) \\ &\geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}). \end{aligned}$$

This is the first-order characterization of convexity (Lemma 2.16). □

2.3.2 Second-order characterization of convexity

If $f : \mathbf{dom}(f) \to \mathbb{R}$ is twice differentiable (meaning that f is differentiable and the gradient function ∇f is also differentiable), convexity can be characterized as follows.

Lemma 2.18. *Suppose that $\mathbf{dom}(f)$ is open and that f is twice differentiable; in particular, the* Hessian *(matrix of second partial derivatives)*

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d}(\mathbf{x}) \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_d \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d}(\mathbf{x}) \end{pmatrix}$$

exists at every point $\mathbf{x} \in \mathbf{dom}(f)$ and is symmetric. Then f is convex if and only if $\mathbf{dom}(f)$ is convex, and for all $\mathbf{x} \in \mathbf{dom}(f)$, we have

$$\nabla^2 f(\mathbf{x}) \succeq 0 \quad \textit{(i.e. } \nabla^2 f(\mathbf{x}) \textit{ is positive semidefinite).} \tag{2.6}$$

(A symmetric matrix M is positive semidefinite*, denoted by $M \succeq 0$, if $\mathbf{x}^\top M\mathbf{x} \geq 0$ for all $\mathbf{x}$, and* positive definite*, denoted by $M \succ 0$, if $\mathbf{x}^\top M\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$.)*

The fact that the Hessians of a twice *continuously* differentiable function are symmetric is a classical result known as the Schwarz theorem [AE08, Corollary 5.5]. But symmetry in fact already holds if f is twice differentiable [Die69, (8.12.3)]. However, if f is only twice *partially* differentiable, we may get non-symmetric Hessians [AE08, Remark 5.6].

Proof. Once again, we employ our favorite univariate function $h(t) := f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$, for fixed $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$ and $t \in I$ where $I \supset [0, 1]$ is a suitable open interval. But this time, we also need h's second derivative. For $t \in I, \mathbf{v} := \mathbf{y} - \mathbf{x}$, we have

$$\begin{aligned} h'(t) &= \nabla f(\mathbf{x} + t\mathbf{v})^\top \mathbf{v}, \\ h''(t) &= \mathbf{v}^\top \nabla^2 f(\mathbf{x} + t\mathbf{v})\mathbf{v}. \end{aligned}$$

The formula for $h'(t)$ has already been derived in the proof of Lemma 2.17, and the formula for $h''(t)$ is Exercise 15.

If f is convex, we always have $h''(0) \geq 0$, as we will show next. Given this, $\nabla^2 f(\mathbf{x}) \succeq 0$ follows for every $\mathbf{x} \in \mathbf{dom}(f)$: by openness of $\mathbf{dom}(f)$, for every $\mathbf{v} \in \mathbb{R}^d$ of sufficiently small norm, there is $\mathbf{y} \in \mathbf{dom}(f)$ such that $\mathbf{v} = \mathbf{y} - \mathbf{x}$, and then $\mathbf{v}^\top \nabla^2 f(\mathbf{x})\mathbf{v} = h''(0) \geq 0$. By scaling, this inequality extends to all $\mathbf{v} \in \mathbb{R}^d$.

To show $h''(0) \geq 0$, we observe that for all sufficiently small δ, $\mathbf{x} + \delta\mathbf{v} \in \mathbf{dom}(f)$ and hence

$$\frac{h'(\delta) - h'(0)}{\delta} = \frac{(\nabla f(\mathbf{x} + \delta\mathbf{v}) - \nabla f(\mathbf{x}))^\top \mathbf{v}}{\delta} = \frac{(\nabla f(\mathbf{x} + \delta\mathbf{v}) - \nabla f(\mathbf{x}))^\top \delta\mathbf{v}}{\delta^2} \geq 0,$$

by monotonicity of the gradient for convex f (Lemma 2.17). It follows that $h''(0) = \lim_{\delta \to 0}(h'(\delta) - h'(0))/\delta \geq 0$.

For the other direction, the mean value theorem applied to h' yields $c \in (0, 1)$ such that $h'(1) - h'(0) = h''(c)$, and spelled out, this is

$$\nabla f(\mathbf{y})^\top \mathbf{v} - \nabla f(\mathbf{x})^\top \mathbf{v} = \mathbf{v}^\top \nabla^2 f(\mathbf{x} + c\mathbf{v})\mathbf{v} \geq 0, \tag{2.7}$$

since $\nabla^2 f(\mathbf{z}) \succeq 0$ for all $\mathbf{z} \in \mathbf{dom}(f)$. Hence, we have proved monotonicity of the gradient which by Lemma 2.17 implies convexity of f. □

Geometrically, Lemma 2.18 means that the graph of f has non-negative curvature everywhere and hence "looks like a bowl". For $f(x_1, x_2) = x_1^2 + x_2^2$, we have

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix},$$

which is a positive definite matrix. In higher dimensions, the same argument can be used to show that the squared distance $d_{\mathbf{y}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|^2$ to a fixed point $\mathbf{y}$ is a convex function; see Exercise 9. The non-squared Euclidean distance $\|\mathbf{x} - \mathbf{y}\|$ is also convex in $\mathbf{x}$, as a consequence of Lemma 2.19(ii) below and the fact that every seminorm (in particular the Euclidean norm $\|x\|$) is convex (Exercise 16). The squared Euclidean distance has the advantage that it is differentiable, while the Euclidean distance itself (whose graph is an "ice cream cone" for $d = 2$) is not.

2.3.3 Operations that preserve convexity

There are three important operations that preserve convexity.

Lemma 2.19 (Exercise 10)**.**

(i) Let $f_1, f_2, \ldots, f_m$ be convex functions, $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{R}_+$. Then $f := \max_{i=1}^m f_i$ as well as $f := \sum_{i=1}^m \lambda_i f_i$ are convex on $\mathbf{dom}(f) := \bigcap_{i=1}^m \mathbf{dom}(f_i)$.

(ii) Let f be a convex function with $\mathbf{dom}(f) \subseteq \mathbb{R}^d$, $g : \mathbb{R}^m \to \mathbb{R}^d$ an affine function, meaning that $g(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$, for some matrix $A \in \mathbb{R}^{d \times m}$ and some vector $\mathbf{b} \in \mathbb{R}^d$. Then the function $f \circ g$ (that maps $\mathbf{x}$ to $f(A\mathbf{x} + \mathbf{b})$) is convex on $\mathbf{dom}(f \circ g) := \{\mathbf{x} \in \mathbb{R}^m : g(\mathbf{x}) \in \mathbf{dom}(f)\}$.

2.4 Minimizing convex functions

The main feature that makes convex functions attractive in optimization is that every local minimum is a global one, so we cannot "get stuck" in local optima. This is quite intuitive if we think of the graph of a convex function as being bowl-shaped.

Definition 2.20. *A local minimum of $f : \mathbf{dom}(f) \to \mathbb{R}$ is a point $\mathbf{x}$ such that there exists $\varepsilon > 0$ with*

$$f(\mathbf{x}) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in \mathbf{dom}(f) \text{ satisfying } \|\mathbf{y} - \mathbf{x}\| < \varepsilon.$$

Lemma 2.21. *Let $\mathbf{x}^\star$ be a local minimum of a convex function $f : \mathbf{dom}(f) \to \mathbb{R}$. Then $\mathbf{x}^\star$ is a global minimum, meaning that*

$$f(\mathbf{x}^\star) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in \mathbf{dom}(f).$$

Proof. Suppose there exists $\mathbf{y} \in \mathbf{dom}(f)$ such that $f(\mathbf{y}) < f(\mathbf{x}^\star)$ and define $\mathbf{y}' := \lambda \mathbf{x}^\star + (1-\lambda)\mathbf{y}$ for $\lambda \in (0,1)$. From convexity (2.2), we get that that $f(\mathbf{y}') < f(\mathbf{x}^\star)$. Choosing λ so close to 1 that $\|\mathbf{y}' - \mathbf{x}^\star\| < \varepsilon$ yields a contradiction to $\mathbf{x}^\star$ being a local minimum. □

This does not mean that a convex function always has a global minimum. Think of $f(x) = x$ as a trivial example. But also if f is bounded from below over $\mathbf{dom}(f)$, it may fail to have a global minimum ($f(x) = e^x$). To ensure the existence of a global minimum, we need additional conditions. For example, it suffices if outside some ball B, all function values are larger than some value $f(\mathbf{x}), \mathbf{x} \in B$. In this case, we can restrict f to B, without changing the smallest attainable value. And on B (which is compact), f attains a minimum by continuity (Lemma 2.14). An easy example: for $f(x_1, x_2) = x_1^2 + x_2^2$, we know that outside any ball containing $\mathbf{0}$, $f(\mathbf{x}) > f(\mathbf{0}) = 0$.

Another easy condition in the differentiable case is given by the following result.

Lemma 2.22. *Suppose that $f : \mathbf{dom}(f) \to \mathbb{R}$ is convex and differentiable over an open domain $\mathbf{dom}(f) \subseteq \mathbb{R}^d$. Let $\mathbf{x} \in \mathbf{dom}(f)$. If $\nabla f(\mathbf{x}) = \mathbf{0}$, then $\mathbf{x}$ is a global minimum.*

Proof. Suppose that $\nabla f(\mathbf{x}) = \mathbf{0}$. According to Lemma 2.16, we have

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) = f(\mathbf{x})$$

for all $\mathbf{y} \in \mathbf{dom}(f)$, so $\mathbf{x}$ is a global minimum. □

The converse is also true and does not even require convexity.

Lemma 2.23. *Suppose that $f : \mathbf{dom}(f) \to \mathbb{R}$ is differentiable over an open domain $\mathbf{dom}(f) \subseteq \mathbb{R}^d$. Let $\mathbf{x} \in \mathbf{dom}(f)$. If $\mathbf{x}$ is a global minimum then $\nabla f(\mathbf{x}) = \mathbf{0}$.*

Proof. Suppose that $\nabla f(\mathbf{x})_i \neq 0$ for some i. For $t \in \mathbb{R}$, we define $\mathbf{x}(t) = \mathbf{x} + t\mathbf{e}_i$, where $\mathbf{e}_i$ is the i-th unit vector. For $|t|$ sufficiently small, we have $\mathbf{x}(t) \in \mathbf{dom}(f)$ since $\mathbf{dom}(f)$ is open. Let $z(t) = f(\mathbf{x}(t))$. By the chain rule, $z'(0) = \nabla f(\mathbf{x})^\top \mathbf{e}_i = \nabla f(\mathbf{x})_i \neq 0$. Hence, z decreases in one direction as we move away from 0, and this yields $f(\mathbf{x}(t)) < f(\mathbf{x})$ for some t, so $\mathbf{x}$ is not a global minimum. □

2.4.1 Strictly convex functions

In general, a global minimum of a convex function is not unique (think of $f(x) = 0$ as a trivial example). However, if we forbid "flat" parts of the graph of f, a global minimum becomes unique (if it exists at all).

Definition 2.24 ([BV04, 3.1.1])**.** *A function $f : \mathbf{dom}(f) \to \mathbb{R}$ is* strictly convex *if (i) $\mathbf{dom}(f)$ is convex and (ii) for all $\mathbf{x} \neq \mathbf{y} \in \mathbf{dom}(f)$ and all $\lambda \in (0, 1)$, we have*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) < \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}). \tag{2.8}$$

This means that the open line segment connecting $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$ is pointwise *strictly* above the graph of f. For example, $f(x) = x^2$ is strictly convex.

Lemma 2.25 ([BV04, 3.1.4])**.** *Suppose that $\mathbf{dom}(f)$ is open and that f is twice continuously differentiable. If the Hessian $\nabla^2 f(\mathbf{x}) \succ \mathbf{0}$ for every $\mathbf{x} \in \mathbf{dom}(f)$ (i.e., $\mathbf{z}^\top \nabla^2 f(\mathbf{x}) \mathbf{z} > 0$ for any $\mathbf{z} \neq \mathbf{0}$), then f is strictly convex.*

The converse is false, though: $f(x) = x^4$ is strictly convex but has vanishing second derivative at $x = 0$.

Lemma 2.26. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be strictly convex. Then f has at most one global minimum.*

Proof. Suppose $\mathbf{x}^\star \neq \mathbf{y}^\star$ are two global minima with $f_{\min} = f(\mathbf{x}^\star) = f(\mathbf{y}^\star)$, and let $\mathbf{z} = \frac{1}{2}\mathbf{x}^\star + \frac{1}{2}\mathbf{y}^\star$. By (2.8),

$$f(\mathbf{z}) < \frac{1}{2} f_{\min} + \frac{1}{2} f_{\min} = f_{\min},$$

a contradiction to $\mathbf{x}^\star$ and $\mathbf{y}^\star$ being global minima. □

2.4.2 Example: Least squares

Suppose we want to fit a hyperplane to a set of data points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ in $\mathbb{R}^d$, based on the hypothesis that the points actually come (approximately) from a hyperplane. A classical method for this is *least squares*. For concreteness, let us do this in $\mathbb{R}^2$. Suppose that the data points are

$$(1, 10), (2, 11), (3, 11), (4, 10), (5, 9), (6, 10), (7, 9), (8, 10),$$

Figure 2.7 (left).

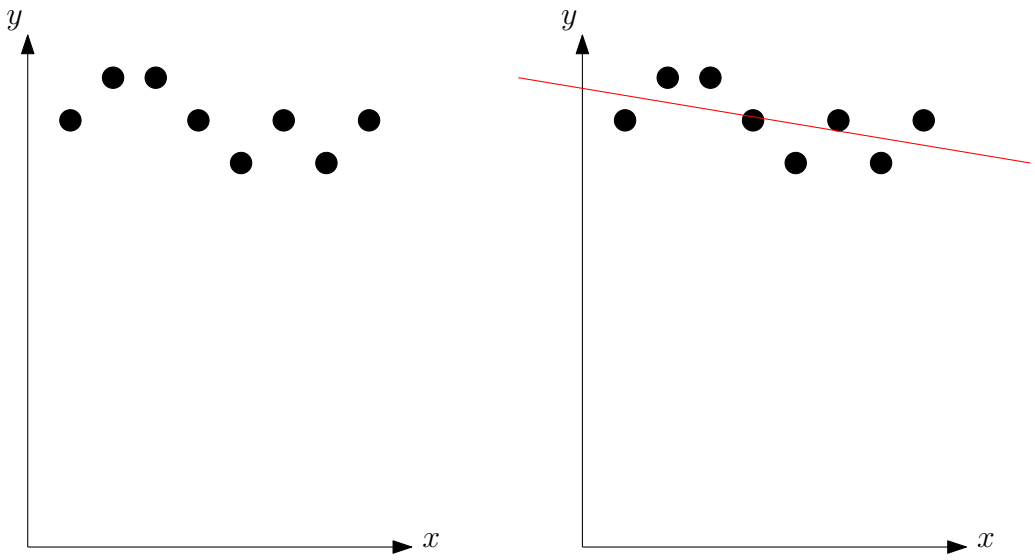


Figure 2.7: Data points in $\mathbb{R}^2$ (left) and least-squares fit (right)

Also, for simplicity (and quite appropriately in this case), let us restrict to fitting a linear model, or more formally to fit non-vertical lines of the form $y = w_0 + w_1 x$. If (x_i, y_i) is the i-th data point, the least squares fit chooses w_0, w_1 such that the *least squares objective*

$$f(w_0, w_1) = \sum_{i=1}^{8} (w_1 x_i + w_0 - y_i)^2$$

is minimized. It easily follows from Lemma 2.19 that f is convex. In fact,

$$f(w_0, w_1) = 204w_1^2 + 72w_1 w_0 - 706w_1 + 8w_0^2 - 160w_0 + 804, \tag{2.9}$$

so we can check convexity directly using the second order condition. We have gradient

$$\nabla f(w_0, w_1) = (72w_1 + 16w_0 - 160, 408w_1 + 72w_0 - 706)$$

and Hessian

$$\nabla^2(w_0, w_1) = \begin{pmatrix} 16 & 72 \\ 72 & 408 \end{pmatrix}.$$

A 2×2 matrix is positive semidefinite if the diagonal elements and the determinant are positive, which is the case here, so f is actually strictly convex and has a unique global minimum. To find it, we solve the linear system $\nabla f(w_0, w_1) = (0, 0)$ of two equations in two unknowns and obtain the global minimum

$$(w_0^\star, w_1^\star) = \Big(\frac{43}{4}, -\frac{1}{6}\Big).$$

Hence, the "optimal" line is

$$y = -\frac{1}{6}x + \frac{43}{4},$$

see Figure 2.7 (right).

2.4.3 Constrained Minimization

Frequently, we are interested in minimizing a convex function only over a subset X of its domain.

Definition 2.27. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be convex and let $X \subseteq \mathbf{dom}(f)$ be a convex set. A point $\mathbf{x} \in X$ is a* minimizer *of f over X if*

$$f(\mathbf{x}) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in X.$$

If f is differentiable, minimizers of f over X have a very useful characterization.

Lemma 2.28 ([BV04, 4.2.3])**.** *Suppose that $f : \mathbf{dom}(f) \to \mathbb{R}$ is convex and differentiable over an open domain $\mathbf{dom}(f) \subseteq \mathbb{R}^d$, and let $X \subseteq \mathbf{dom}(f)$ be a convex set. Point $\mathbf{x}^\star \in X$ is a minimizer of f over X if and only if*

$$\nabla f(\mathbf{x}^\star)^\top (\mathbf{x} - \mathbf{x}^\star) \geq 0 \quad \forall \mathbf{x} \in X.$$

If X does not contain the global minimum, then Lemma 2.28 has a nice geometric interpretation. Namely, it means that X is contained in the halfspace $\{\mathbf{x} \in \mathbb{R}^d : \nabla f(\mathbf{x}^\star)^\top(\mathbf{x} - \mathbf{x}^\star) \geq 0\}$ (normal vector $\nabla f(\mathbf{x}^\star)$ at $\mathbf{x}^\star$ pointing into the halfspace); see Figure 2.8. In still other words, $\mathbf{x} - \mathbf{x}^\star$ forms a non-obtuse angle with $\nabla f(\mathbf{x}^\star)$ for all $\mathbf{x} \in X$.

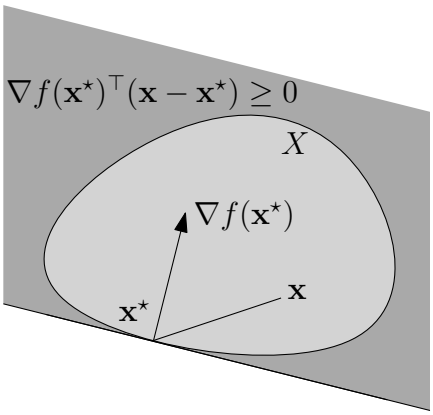


Figure 2.8: Optimality condition for constrained optimization

We typically write constrained minimization problems in the form

$$\operatorname{argmin}\{f(\mathbf{x}) : \mathbf{x} \in X\} \tag{2.10}$$

or

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in X. \end{array} \tag{2.11}$$

2.5 Existence of a minimizer

The existence of a minimizer (or a global minimum if $X = \mathbf{dom}(f)$) will be an assumption made by most minimization algorithms that we discuss later. In practice, such algorithms are being used (and often also work) if there is no minimizer. By "work", we mean in this case that they compute a point $\mathbf{x}$ such that $f(\mathbf{x})$ is close to $\inf_{\mathbf{y}\in X} f(\mathbf{y})$, assuming that the infimum is finite (as in $f(x) = e^x$). But a sound theoretical analysis usually requires the existence of a minimizer. Therefore, this section develops tools that may helps us in analyzing whether this is the case for a given

convex function. To avoid technicalities, we restrict ourselves to the case $\mathbf{dom}(f) = \mathbb{R}^d$.

2.5.1 Sublevel sets and the Weierstrass Theorem

Definition 2.29. *Let $f : \mathbb{R}^d \to \mathbb{R}$, $\alpha \in \mathbb{R}$. The set*

$$f^{\leq\alpha} := \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) \leq \alpha\}$$

is the α-sublevel set of f; see Figure 2.9

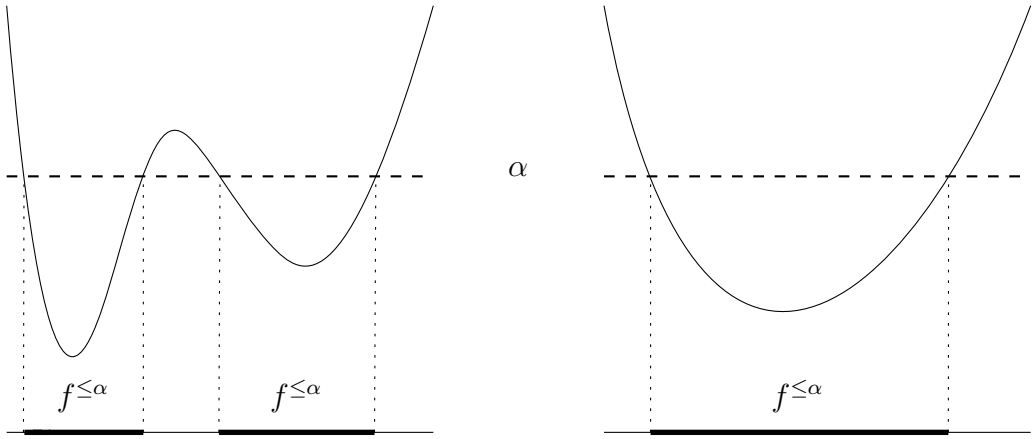


Figure 2.9: Sublevel set of a non-convex function (left) and a convex function (right)

It is easy to see from the definition that every sublevel set of a convex function is convex. Moreover, as a consequence of continuity of f, sublevel sets are closed. The following (known as the Weierstrass Theorem) just formalizes an argument that we have made earlier.

Theorem 2.30. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous function, and suppose there is a nonempty and bounded sublevel set $f^{\leq\alpha}$. Then f has a global minimum.*

Proof. As the set $(-\infty, \alpha]$ is closed, its pre-image $f^{\leq\alpha}$ by the continuous function f is closed. We know that f—as a continuous function—attains a minimum over the (non-empty) closed and bounded (= compact) set $f^{\leq\alpha}$ at some $\mathbf{x}^\star$. This $\mathbf{x}^\star$ is also a global minimum as it has value $f(\mathbf{x}^\star) \leq \alpha$, while any $\mathbf{x} \notin f^{\leq\alpha}$ has value $f(\mathbf{x}) > \alpha \geq f(\mathbf{x}^\star)$. □

Note that Theorem 2.30 holds for convex functions as convexity on $\mathbb{R}^d$ implies continuity (Exercise 8).

2.5.2 Recession cone and lineality space

What happens if there is no bounded sublevel set? Then f may ($f(x) = 0$) or may not ($f(x) = x, f(x) = e^x$) have a global minimum. But what cannot happen is that some nonempty sublevel sets are bounded, and others are not. Also, in the unbounded case, all nonempty sublevel sets have the same "reason" for being unbounded, namely the same *directions of recession*. Again, we assume for simplicity that $\mathbf{dom}(f) = \mathbb{R}^d$, and we are only considering unconstrained minimization. But most of the material can be adapted to general domains and to constrained optimization over a convex set $X \subseteq \mathbf{dom}(f)$. We closely follow Bertsekas [Ber05].

Recession cone and lineality space of a convex set.

Definition 2.31. *Let $C \subseteq \mathbb{R}^d$ be a convex set. Then $\mathbf{y} \in \mathbb{R}^d$ is a* direction of recession *of C if for some $\mathbf{x} \in C$ and all $\lambda \geq 0$, it holds that $\mathbf{x} + \lambda\mathbf{y} \in C$.*

This means that C is unbounded in direction $\mathbf{y}$. Whether $\mathbf{y}$ is a direction of recession only depends on C and not on the particular $\mathbf{x}$, assuming that C is closed (otherwise, it may be false).

Lemma 2.32. *Let $C \subseteq \mathbb{R}^d$ be a nonempty closed convex set, and let $\mathbf{y} \in \mathbb{R}^d$. The following statements are equivalent.*

(i) $\exists \mathbf{x} \in C : \mathbf{x} + \lambda\mathbf{y} \in C$ *for all* $\lambda \geq 0$.

(ii) $\forall \mathbf{x} \in C : \mathbf{x} + \lambda\mathbf{y} \in C$ *for all* $\lambda \geq 0$.

Proof. We need to show that (i) implies (ii), so choose $\mathbf{x} \in C, \mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{x} + \lambda\mathbf{y} \in C$ for all $\lambda \geq 0$. Fix $\lambda > 0$, let $\mathbf{z} = \lambda\mathbf{y}$ and $\mathbf{x}' \in C$. To get (ii), we prove that $\mathbf{x}' + \mathbf{z} \in C$. To this end, we define sequences $(\mathbf{w}_k), (\mathbf{z}_k), k \in \mathbb{N}$ via

$$\begin{aligned} \mathbf{w}_k &:= \mathbf{x} + k\mathbf{z} \in C \quad \text{(by (i))} \\ \mathbf{z}_k &:= \frac{1}{k}(\mathbf{w}_k - \mathbf{x}') = \mathbf{z} + \frac{1}{k}(\mathbf{x} - \mathbf{x}'), \end{aligned}$$

see Figure 2.10. By definition of a convex set, we have $\mathbf{x}' + \mathbf{z}_k = \frac{1}{k}\mathbf{w}_k + \left(1 - \frac{1}{k}\right)\mathbf{x}' \in C$. Moreover, $\mathbf{z}_k$ converges to $\mathbf{z}$, so $\mathbf{x}' + \mathbf{z}_k$ converges to $\mathbf{x}' + \mathbf{z} \in C$, and this is an element of C, since C is closed. □

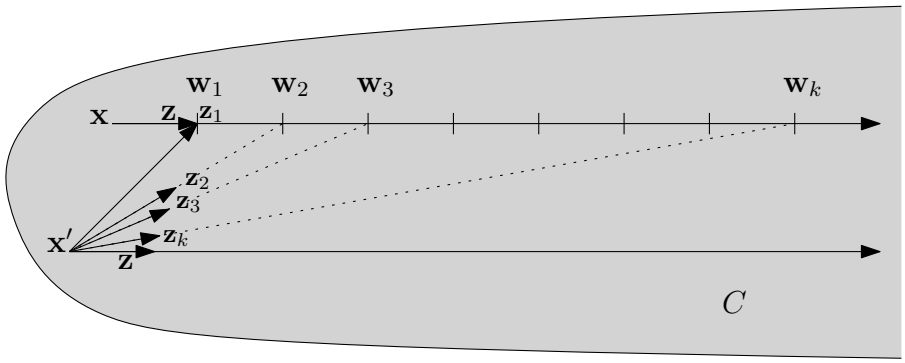


Figure 2.10: The proof of Lemma 2.32

The directions of recession of C actually form a *convex cone*, a set that is closed under taking non-negative linear combinations. This is known as the *recession cone* $R(C)$ of C; see Figure 2.11 (left).

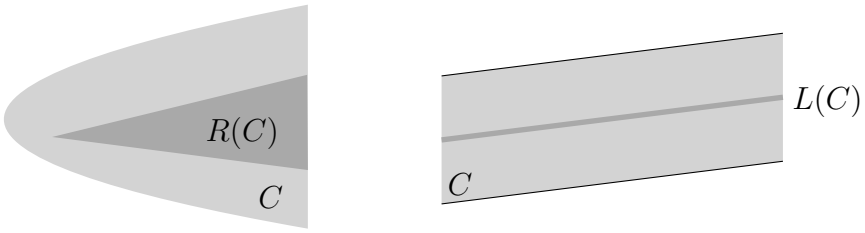


Figure 2.11: The recession cone and lineality space of a convex set

Lemma 2.33. *Let $C \subseteq \mathbb{R}^d$ be a closed convex set, and let $\mathbf{y}_1, \mathbf{y}_2$ be directions of recession of C; $\lambda_1, \lambda_2 \in \mathbb{R}_+$. Then $\mathbf{y} = \lambda_1\mathbf{y}_1 + \lambda_2\mathbf{y}_2$ is also a direction of recession of C.*

Proof. The statement is trivial if $\mathbf{y} = \mathbf{0}$. Otherwise, after scaling $\mathbf{y}$ by $1/(\lambda_1 + \lambda_2) > 0$ (which does not affect the property of being a direction

of recession), we may assume that $\lambda_1 + \lambda_2 = 1$. Now, for all $\mathbf{x} \in C$ and all $\lambda \in \mathbb{R}$, we get

$$\mathbf{x} + \lambda\mathbf{y} = \mathbf{x} + \lambda_1\lambda\mathbf{y}_1 + \lambda_2\lambda\mathbf{y}_2 = \lambda_1(\underbrace{\mathbf{x} + \lambda\mathbf{y}_1}_{\in C}) + \lambda_2(\underbrace{\mathbf{x} + \lambda\mathbf{y}_2}_{\in C}) \in C.$$

□

Definition 2.34. *Let $C \subseteq \mathbb{R}^d$ be a convex set. $\mathbf{y} \in \mathbb{R}^d$ is a* direction of constancy *of C if both $\mathbf{y}$ and $-\mathbf{y}$ are directions of recession of C.*

This means that C is unbounded along the whole line spanned by $\mathbf{y}$. The directions of constancy form a linear subspace, the *lineality space* $L(C)$ of C; see Figure 2.11 (right).

Lemma 2.35. *Let $C \subseteq \mathbb{R}^d$ be a closed convex set, and let $\mathbf{y}_1, \mathbf{y}_2$ be directions of constancy of C; $\lambda_1, \lambda_2 \in \mathbb{R}$. Then $\mathbf{y} = \lambda_1\mathbf{y}_1 + \lambda_2\mathbf{y}_2$ is also a direction of constancy of C.*

Proof. After replacing $\mathbf{y}_i$ with the direction of recession $-\mathbf{y}_i$ if necessary ($i = 1, 2$), we may assume that $\lambda_1, \lambda_2 \geq 0$, so $\mathbf{y}$ is a direction of recession by Lemma 2.33. The same argument works for $-\mathbf{y}$, so $\mathbf{y}$ is a direction of constancy. □

Recession cone and lineality space of a convex function. For this, we look at directions of recession of sublevel sets (which are closed and convex in our case).

Lemma 2.36. *Let $f : \mathbb{R}^d \to R$ be a convex function. Any two nonempty sublevel sets $f^{\leq\alpha}, f^{\leq\alpha'}$ have the same recession cones, i.e. $R(f^{\leq\alpha}) = R(f^{\leq\alpha'})$.*

Proof. Let $\mathbf{y}$ be a direction of recession for $f^{\leq\alpha}$, i.e. for all $\mathbf{x} \in f^{\leq\alpha}$ and all $\lambda \geq 0$, we have

$$f(\mathbf{x} + \lambda\mathbf{y}) \leq \alpha.$$

We claim that this implies the stronger bound

$$f(\mathbf{x} + \lambda\mathbf{y}) \leq f(\mathbf{x}). \tag{2.12}$$

In words, f is non-increasing along any direction of recession. Using this, the statement follows: Because $f^{\leq\alpha}$ and $f^{\leq\alpha'}$ are nonempty, there exists

$\mathbf{x}' \in f^{\leq\alpha} \cap f^{\leq\alpha'}$, and then we have $f(\mathbf{x}' + \lambda\mathbf{y}) \leq f(\mathbf{x}') \leq \alpha'$, so $\mathbf{y}$ is a direction of recession for $f^{\leq\alpha'}$.

To prove (2.12), we fix λ and let $\mathbf{z} = \lambda\mathbf{y}$. With $\mathbf{w}_k := \mathbf{x} + k\mathbf{z} \in f^{\leq\alpha}$, we have

$$\mathbf{x} + \mathbf{z} = \left(1 - \frac{1}{k}\right)\mathbf{x} + \frac{1}{k}\mathbf{w}_k,$$

so convexity of f and the fact that $\mathbf{w}_k \in f^{\leq\alpha}$ yields

$$f(\mathbf{x} + \mathbf{z}) \leq \left(1 - \frac{1}{k}\right) f(\mathbf{x}) + \frac{1}{k} f(\mathbf{w}_k) \leq \left(1 - \frac{1}{k}\right) f(\mathbf{x}) + \frac{1}{k}\alpha. \tag{2.13}$$

Thus, as $k \to \infty$, the right side of (2.13) tends to $f(x)$, and therefore $f(\mathbf{x} + \mathbf{z}) \leq f(\mathbf{x})$; see Figure 2.12. □

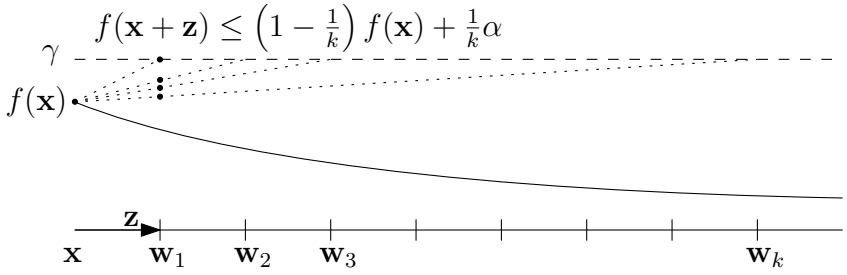


Figure 2.12: The proof of (2.12)

Definition 2.37. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. Then $\mathbf{y} \in \mathbb{R}^d$ is a direction of recession (of constancy, respectively) of f if $\mathbf{y}$ is a direction of recession (of constancy, respectively) for some (equivalently, for every) nonempty sublevel set. The set of directions of recession of f is called the* recession cone *$R(f)$ of f. The set of directions of constancy of f is called the* lineality space *$L(f)$ of f.*

We can characterize recession cone and lineality space of f directly, without looking at sublevel sets (the proof is Exercise 11). The conditions of Lemma 2.38(ii) and Lemma 2.39(ii) finally explain the terms "direction of recession" and "direction of constancy".

Lemma 2.38. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. The following statements are equivalent.*

(i) $\mathbf{y} \in \mathbb{R}^d$ *is a direction of recession of* f*.*

(ii) $f(\mathbf{x} + \lambda\mathbf{y}) \leq f(\mathbf{x})$ *for all* $\mathbf{x} \in \mathbb{R}^d$ *and all* $\lambda \in \mathbb{R}_+$*.*

(iii) $(\mathbf{y}, 0)$ *is a ("horizontal") direction of recession of (the closed convex set)* $epi(f)$*.*

Lemma 2.39. *Let* $f : \mathbb{R}^d \to \mathbb{R}$ *be convex. The following statements are equivalent.*

(i) $\mathbf{y} \in \mathbb{R}^d$ *is a direction of constancy of* f*.*

(ii) $f(\mathbf{x} + \lambda\mathbf{y}) = f(\mathbf{x})$ *for all* $\mathbf{x} \in \mathbb{R}^d$ *and all* $\lambda \in \mathbb{R}$*.*

(iii) $(\mathbf{y}, 0)$ *is a ("horizontal") direction of constancy of (the closed convex set)* $epi(f)$*.*

2.5.3 Coercive convex functions

Definition 2.40. *A convex function* f *is* coercive *if its recession cone is trivial, meaning that* $\mathbf{0}$ *is its only direction of recession.*[1]

Coercivity means that along any direction, $f(\mathbf{x})$ goes to infinity. An example of a coercive convex function is $f(x_1, x_2) = x_1^2 + x_2^2$. Non-coercive functions are $f(x) = x$ and $f(x) = e^x$ (any $y \leq 0$ is a direction of recession). For a constant function $f : \mathbb{R}^d \to \mathbb{R}$, every direction $\mathbf{y}$ is a direction of recession. In general, affine functions are never coercive.

Lemma 2.41. *Let* $f : \mathbb{R}^d \to \mathbb{R}$ *be a coercive convex function. Then every nonempty sublevel set* $f^{\leq\alpha}$ *is bounded.*

This may seem obvious, as $f^{\leq\alpha}$ is bounded in every direction by coercivity. But we still need an argument that there is a global bound.

Proof. Let $f^{\leq\alpha}$ be a nonempty sublevel, and assume without loss of generality that $\mathbf{0} \in f^{\leq\alpha}$, i.e., $\alpha \geq f(\mathbf{0})$.

Let $S^{d-1} = \{\mathbf{y} \in \mathbb{R}^d : \|y\| = 1\}$ be the unit sphere. We define a function $g : S^{d-1} \to \mathbb{R}$ via

$$g(\mathbf{y}) = \max\{\lambda \geq 0 : f(\lambda\mathbf{y}) \leq \alpha\}, \quad \mathbf{y} \in S^{d-1}.$$

[1]The usual definition of a coercive function is that $f(\mathbf{x}) \to \infty$ whenever $\|\mathbf{x}\| \to \infty$. In the convex case, both definitions agree.

Since f is continuous and has no nonzero direction of recession, we know that for each $\mathbf{y} \in S^{d-1}$ the set $\{\lambda \geq 0 : f(\lambda \mathbf{y}) \leq \alpha\}$ is closed and bounded (it is actually an interval, by convexity of f), so the maximum exists and $g(\mathbf{y})$ is well-defined. We claim that g is continuous.

Let $(\mathbf{y}'_k)_{k \in \mathbb{N}}$ be a sequence of unit vectors such that $\lim_{k\to\infty} \mathbf{y}_k = \mathbf{y} \in S^{d-1}$. We need to show that $\lim_{k\to\infty} g(\mathbf{y}') = g(\mathbf{y})$. Let us fix $\varepsilon > 0$ arbitrarily small. For $\underline{\lambda} := g(\mathbf{y}) - \varepsilon \geq 0$, we have $f(\underline{\lambda}\mathbf{y}) \leq \alpha$ (an easy consequence of convexity of f and $\alpha \geq f(\mathbf{0})$). And for $\overline{\lambda} := g(\mathbf{y}) + \varepsilon$, we get $f(\overline{\lambda}\mathbf{y}) > \alpha$ by definition of $g(\mathbf{y})$. Continuity of f then yields $\lim_{k\to\infty} f(\underline{\lambda}\mathbf{y}_k) = f(\underline{\lambda}\mathbf{y}) \leq \alpha$ and $\lim_{k\to\infty} f(\overline{\lambda}\mathbf{y}_k) = f(\overline{\lambda}\mathbf{y}) > \alpha$. Hence, for sufficiently large k, $g(\mathbf{y}_k) \in [\underline{\lambda}, \overline{\lambda}] = [g(\mathbf{y}) - \varepsilon, g(\mathbf{y}) + \varepsilon]$, and $\lim_{k\to\infty} g(\mathbf{y}') = g(\mathbf{y})$ follows.

As a continuous function, g attains a maximum $\lambda^\star$ over the compact set S^{d-1}, and this means that $f^{\leq\alpha}$ is contained in the closed $\lambda^\star$-ball around the origin. Hence, $f^{\leq\alpha}$ is bounded. □

Together with Theorem 2.30, we obtain

Theorem 2.42. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a coercive convex function. Then f has a global minimum.*

2.5.4 Weakly coercive convex functions

It turns out that we can allow nontrivial directions of recession and still guarantee a global minimum.

Definition 2.43. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. Function f is called* weakly coercive *if its recession cone equals its lineality space, i.e. every direction of recession is a direction of constancy.*

Function $f(x) = 0$ is a trivial example of a non-coercive but weakly coercive function. A more interesting example is $f(x_1, x_2) = x_1^2$. Here, the directions of recession are all vectors of the form $\mathbf{y} = (0, x_2)$, and these are at the same time directions of constancy.

Theorem 2.44. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a weakly coercive convex function. Then f has a global minimum.*

Proof. We know that the lineality space L of f is a linear subspace of $\mathbb{R}^d$: by Definition 2.37, L is the lineality space of every nonempty (closed and convex) sublevel set, and as such it is closed under taking linear combinations

(Lemma 2.35). Let $L^\perp$ be the orthogonal complement of L. Restricted to $L^\perp$, f is coercive, as $L^\perp$ is orthogonal to any direction of constancy, equivalently to every direction of recession, since f is weakly coercive. Therefore, $L^\perp$ can contain only the trivial direction of recession. It follows that $f_{|L^\perp}$ has a global minimum $\mathbf{x}^\star \in L^\perp$ by Theorem 2.42 (which we can apply after identifying $L^\perp$ w.l.o.g. with $\mathbb{R}^m$ for some $m \leq n$). This is also a global minimum of f. To see this, let $\mathbf{z} \in \mathbb{R}^d$ and write it (uniquely) in the form $\mathbf{z} = \mathbf{x} + \mathbf{y}$ with $\mathbf{x} \in L^\perp$ and $\mathbf{y} \in L$. Then we get

$$f(\mathbf{z}) = f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) \geq f(\mathbf{x}^\star),$$

where the second equality follows from $\mathbf{y}$ being a direction of constancy; see Lemma 2.39(ii). □

2.6 Examples

In the following two sections, we give two examples of convex function minimization tasks that arise from machine learning applications.

2.6.1 Handwritten digit recognition

Suppose you want to write a program that recognizes handwritten decimal digits $0, 1, \ldots, 9$. You have a set P of grayscale images (28×28 pixels, say) that represent handwritten decimal digits, and for each image $\mathbf{x} \in P$, you know the digit $d(\mathbf{x}) \in \{0, \ldots, 9\}$ that it represents, see Figure 2.13. You want to train your program with the set P, and after that, use it to recognize handwritten digits in arbitrary 28×28 images.

The classical approach is the following. We represent an image as a *feature vector* $\mathbf{x} \in \mathbb{R}^{784}$, where x_i is the gray value of the i-th pixel (in some order). During the training phase, we compute a matrix $W \in \mathbb{R}^{10 \times 784}$ and then use the vector $\mathbf{y} = W\mathbf{x} \in \mathbb{R}^{10}$ to predict the digit seen in an arbitrary image $\mathbf{x}$. The idea is that $y_j, j = 0, \ldots, 9$ corresponds to the probability of the digit being j. This does not work directly, since the entries of $\mathbf{y}$ may be negative and generally do not sum up to 1. But we can convert $\mathbf{y}$ to a vector $\mathbf{z}$ of actual probabilities, such that a small y_j leads to a small probability z_j and a large y_j to a large probability z_j. How to do this is not canonical, but here is a well-known formula that works:

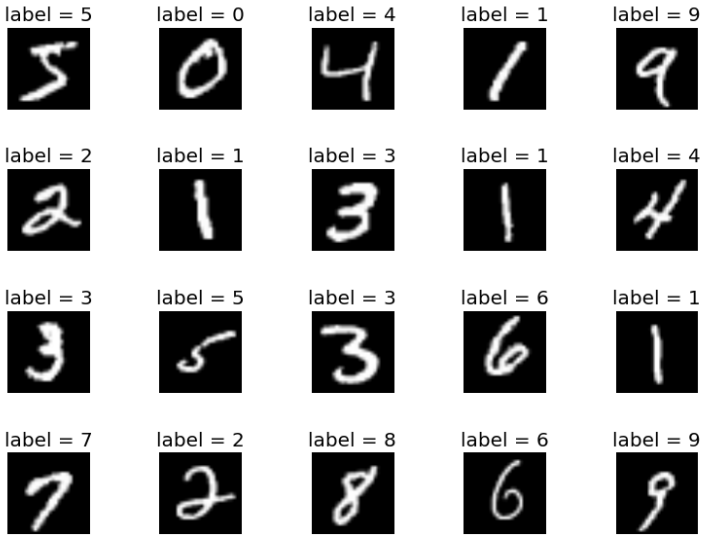


Figure 2.13: Some training images from the MNIST data set (picture from `http://corochann.com/mnist-dataset-introduction-1138.html`

$$z_j = z_j(\mathbf{y}) = \frac{e^{y_j}}{\sum_{k=0}^{9} e^{y_k}}. \tag{2.14}$$

The classification then simply outputs digit j with probability z_j. The matrix W is chosen such that it (approximately) minimizes the classification error on the training set P. Again, it is not canonical how we measure classification error; here we use the following *loss function* to evaluate the error induced by a given matrix W.

$$\ell(W) = -\sum_{\mathbf{x}\in P} \ln\left(z_{d(\mathbf{x})}(W\mathbf{x})\right) = \sum_{\mathbf{x}\in P}\left(\ln\left(\sum_{k=0}^{9} e^{(W\mathbf{x})_k}\right) - (W\mathbf{x})_{d(\mathbf{x})}\right). \tag{2.15}$$

This function "punishes" images for which the correct digit j has low probability z_j (corresponding to a significantly negative value of $\log z_j$). In an ideal world, the correct digit would always have probability 1, resulting in $\ell(W) = 0$. But under (2.14), probabilities are always strictly between 0 and 1, so we have $\ell(W) > 0$ for all W.

Exercise 12 asks you to prove that ℓ is convex. In Exercise 13, you will characterize the situations in which ℓ has a global minimum.

2.6.2 Master's Admission

The computer science department of a well known Swiss university is admitting top international students to its MSc program, in a competitive application process. Applicants are submitting various documents (GPA, TOEFL test score, GRE test scores, reference letters,...). During the evaluation of an application, the admission committee would like to compute a (rough) forecast of the applicant's performance in the MSc program, based on the submitted documents.[2]

Data on the actual performance of students admitted in the past is available. To keep things simple in the following example, Let us base the forecast on GPA (grade point average) and TOEFL (Test of English as a Foreign Language) only. GPA scores are normalized to a scale with a minimum of 0.0 and a maximum of 4.0, where admission starts from 3.5. TOEFL scores are on an integer scale between 0 and 120, where admission starts from 100.

Table 2.1 contains the known data. GGPA (graduation grade point average on a Swiss grading scale) is the average grade obtained by an admitted student over all courses in the MSc program. The Swiss scale goes from 1 to 6 where 1 is the lowest grade, 6 is the highest, and 4 is the lowest passing grade.

As in Section 2.4.2, we are attempting a linear regression with least squares fit, i.e. we are making the hypothesis that

$$\mathrm{GGPA} \approx w_0 + w_1 \cdot \mathrm{GPA} + w_2 \cdot \mathrm{TOEFL}. \tag{2.16}$$

However, in our scenario, the relevant GPA scores span a range of only 0.5 while the relevant TOEFL scores span a range of 20. The resulting least squares objective would be somewhat ugly; we already saw this in our previous example (2.9), where the data points had large second coordinate, resulting in the w_1-scale being very different from the w_2-scale. This time, we normalize first, so that w_1 und w_2 become comparable and allow us to understand the relative influences of GPA and TOEFL.

[2] Any resemblance to real departments is purely coincidental. Also, no serious department will base performance forecasts on data from 10 students, as we will do it here.

GPA	TOEFL	GGPA
3.52	100	3.92
3.66	109	4.34
3.76	113	4.80
3.74	100	4.67
3.93	100	5.52
3.88	115	5.44
3.77	115	5.04
3.66	107	4.73
3.87	106	5.03
3.84	107	5.06

Table 2.1: Data for 10 admitted students: GPA and TOEFL scores (at time of application), GGPA (at time of graduation)

The general setting is this: we have n *inputs* $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where each vector $\mathbf{x}_i \in \mathbb{R}^d$ consists of d input variables; then we have n *outputs* $y_1, \ldots, y_n \in \mathbb{R}$. Each pair $(\mathbf{x}_i, y_i)$ is an *observation*. In our case, $d = 2, n = 10$, and for example, $((3.93, 100), 5.52)$ is an observation (of a student doing very well).

With variable *weights* $w_0, \mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$, we plan to minimize the least squares objective

$$f(w_0, \mathbf{w}) = \sum_{i=1}^{n} (w_0 + \mathbf{w}^\top \mathbf{x}_i - y_i)^2.$$

We first want to assume that the inputs and outputs are *centered*, meaning that

$$\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}, \quad \frac{1}{n} \sum_{i=1}^{n} y_i = 0.$$

This can be achieved by simply subtracting the mean $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$ from every input and the mean $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ from every output. In our example, this yields the numbers in Table 2.2 (left).

After centering, the global minimum $(w_0^\star, \mathbf{w}^\star)$ of the least squares objective satisfies $w_0^\star = 0$ while $\mathbf{w}^\star$ is unaffected by centering (Exercise 17), so that we can simply omit the variable w_0 in the sequel.

GPA	TOEFL	GGPA
-0.24	-7.2	-0.94
-0.10	1.8	-0.52
-0.01	5.8	-0.05
-0.02	-7.2	-0.18
0.17	-7.2	0.67
0.12	7.8	0.59
0.01	7.8	0.19
-0.10	-0.2	-0.12
0.11	-1.2	0.17
0.07	-0.2	0.21

GPA	TOEFL	GGPA
-2.04	-1.28	-0.94
-0.88	0.32	-0.52
-0.05	1.03	-0.05
-0.16	-1.28	-0.18
1.42	-1.28	0.67
1.02	1.39	0.59
0.06	1.39	0.19
-0.88	-0.04	-0.12
0.89	-0.21	0.17
0.62	-0.04	0.21

Table 2.2: Centered observations (left); normalized inputs (right)

Finally, we assume that all d input variables are on the same scale, meaning that

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1, \quad j = 1, \ldots, d.$$

To achieve this for fixed j (assuming that no variable is 0 in all inputs), we multiply all x_{ij} by $s(j) = \sqrt{n/\sum_{i=1}^{n} x_{ij}^2}$ (which, in the optimal solution $\mathbf{w}^\star$, just multiplies $w_j^\star$ by $1/s(j)$, an argument very similar to the one in Exercise 17). For our data set, the resulting normalized data are shown in Table 2.2 (right). Now the least squares objective (after omitting w_0) is

$$\begin{aligned} f(w_1, w_2) &= \sum_{i=1}^{10} (w_1 x_{i1} + w_2 x_{i2} - y_i)^2 \\ &\approx 10w_1^2 + 10w_2^2 + 1.99 w_1 w_2 - 8.7 w_1 - 2.79 w_2 + 2.09. \end{aligned}$$

This is minimized at

$$\mathbf{w}^\star = (w_1^\star, w_2^\star) \approx (0.43, 0.097),$$

so if our initial hypothesis (2.16) is true, we should have

$$y_i \approx y_i^\star = 0.43 x_{i1} + 0.097 x_{i2} \tag{2.17}$$

in the normalized data. This can quickly be checked, and the results are not perfect, but not too bad, either; see Table 2.3 (ignore the last column for now).

x_{i1}	x_{i2}	y_i	$y_i^\star$	$z_i^\star$
-2.04	-1.28	-0.94	-1.00	-0.87
-0.88	0.32	-0.52	-0.35	-0.37
-0.05	1.03	-0.05	0.08	-0.02
-0.16	-1.28	-0.18	-0.19	-0.07
1.42	-1.28	0.67	0.49	0.61
1.02	1.39	0.59	0.57	0.44
0.06	1.39	0.19	0.16	0.03
-0.88	-0.04	-0.12	-0.38	-0.37
0.89	-0.21	0.17	0.36	0.38
0.62	-0.04	0.21	0.26	0.27

Table 2.3: Outputs $y_i^\star$ predicted by the linear model (2.17) and by the model $z_i^\star = 0.43x_{i1}$ that simply ignores the second input variable

What we also see from (2.17) is that the first input variable (GPA) has a much higher influence on the output (GGPA) than the second one (TOEFL). In fact, if we drop the second one altogether, we obtain outputs $z_i^\star$ (last column in Table 2.3) that seem equivalent to the predicted outputs $y_i^\star$ within the level of noise that we have anyway.

We conclude that TOEFL scores are probably not indicative for the performance of admitted students, so the admission committee should not care too much about them. Requiring a minimum score of 100 might make sense, but whenever an applicant reaches at least this score, the actual value does not matter.

The LASSO. So far, we have computed linear functions $y = 0.43x_1 + 0.097x_2$ and $z = 0.43x_1$ that "explain" the historical data from Table 2.1. However, they are optimized to fit the historical data, not the future. We may have *overfitting*. This typyically leads to unrealiable predictions of high variance in the future. Also, ideally, we would like non-indicative variables (such as the TOEFL in our example) to actually have weight 0, so that the model "knows" the important variables and is therefore better to interpret.

The question is: how can we in general improve the quality of our forecast? There are various heuristics to identify the "important" variables

(subset selection). A very simple one is just to forget about weights close to 0 in the least squares solution. However, for this, we need to define what it means to be close to 0; and it may happen that small changes in the data lead to different variables being dropped if their weights are around the threshold. On the other end of the spectrum, there is *best subset selection* where we compute the least squares solution subject to the constraint that there are at most k nonzero weights, for some k that we believe is the right number of important variables. This is NP-hard, though.

A popular approach that in many cases improves forecasts and at the same time identifies important variables has been suggested by Tibshirani in 1996 [Tib96]. Instead of minimizing the least squares objective globally, it is minimized over a suitable ℓ_1-ball (ball in the 1-norm $\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$):

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} \|\mathbf{w}^\top \mathbf{x}_i - y_i\|^2 \\ \text{subject to} & \|\mathbf{w}\|_1 \leq R, \end{array} \tag{2.18}$$

where $R \in \mathbb{R}_+$ is some parameter. In our case, if we for example

$$\begin{array}{ll} \text{minimize} & f(w_1, w_2) = 10w_1^2 + 10w_2^2 + 1.99w_1w_2 - 8.7w_1 - 2.79w_2 + 2.09 \\ \text{subject to} & |w_1| + |w_2| \leq 0.2, \end{array} \tag{2.19}$$

we obtain weights $\mathbf{w}^\star = (w_1^\star, w_2^\star) = (0.2, 0)$: the non-indicative TOEFL score has disappeared automatically! For $R = 0.3$, the same happens (with $w_1^\star = 0.3$, respectively). For $R = 0.4$, the TOEFL score starts creeping back in: we get $(w_1^\star, w_2^\star) \approx (0.36, 0.036)$. For $R = 0.5$, we have $(w_1^\star, w_2^\star) \approx (0.41, 0.086)$, while for $R = 0.6$ (and all larger values of R), we recover the original solution $(w_1^\star, w_2^\star) = (0.43, 0.097)$.

It is important to understand that using the "fixed" weights (which may be significantly shrunken), we make predictions *worse* on the historical data (this must be so, since least squares was optimal for the historical data). But future predictions may benefit (a lot). To quantify this benefit, we need to make statistical assumptions about future observations; this is beyond the scope of our treatment here.

The phenomenon that adding a constraint on $\|\mathbf{w}\|_1$ tends to set weights to 0 is not restricted to $d = 2$. The constrained minimization problem (2.18) is called the *LASSO* (least absolute shrinkage and selection operator) and has the tendency to assign weights of 0 and thus to select a subset of input variables, where R controls how aggressive the selection is.

In our example, it is easy to get an intuition why this works. Let us look at the case $R = 0.2$. The smallest value attainable in (2.19) is the smallest α such that that the (elliptical) sublevel set $f^{\leq\alpha}$ of the least squares objective f still intersects the ℓ_1-ball $\{(w_1, w_2) : |w_1|+|w_2| \leq 0.2\}$. This smallest value turns out to be $\alpha = 0.75$, see Figure 2.14. For this value of α, the sublevel set intersects the ℓ_1-ball exactly in one point, namely $(0.2, 0)$.

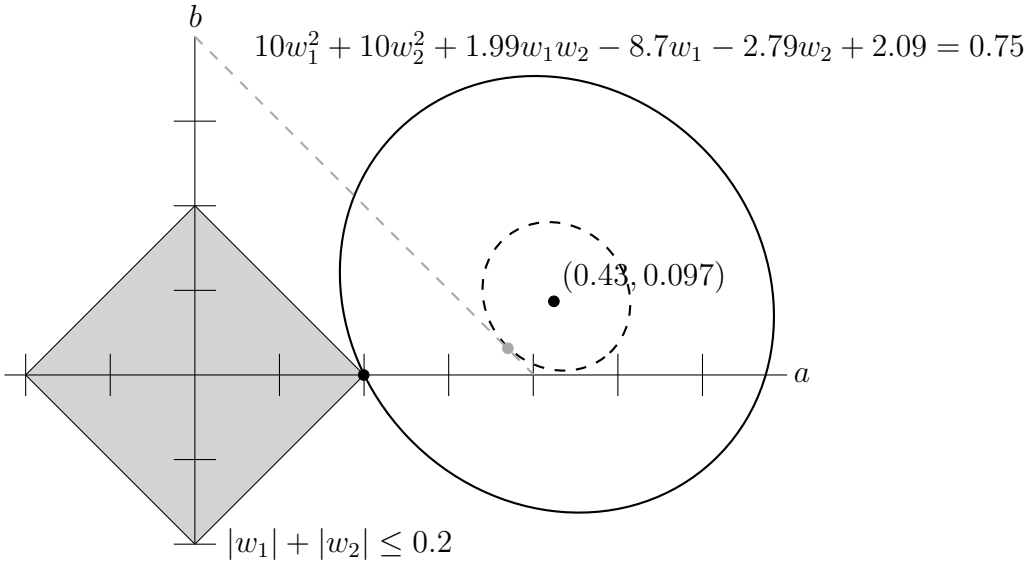


Figure 2.14: Lasso

At $(0.2, 0)$, the ellipse $\{(w_1, w_2) : f(w_1, w_2) = \alpha\}$ is "vertical enough" to just intersect the corner of the ℓ_1-ball. The reason is that the center of the ellipse is relatively close to the w_1-axis, when compared to its size. As R increases, the relevant value of α decreases, the ellipse gets smaller and less vertical around the w_1-axis; until it eventually stops intersecting the ℓ_1-ball $\{(w_1, w_2) : |w_1|+|w_2| \leq R\}$ in a corner (dashed situation in Figure 2.14, for $R = 0.4$).

Even though we have presented a toy example in this section, the background is real. The theory of admission and in particular performance forecasts has been developed in a recent PhD thesis by Zimmermann [Zim16].

2.7 Convex programming

Convex programs are specific convex constrained minimization problems. They arise when we minimize a convex function f over a convex set X defined by finitely many convex inequality and affine equality constraints. This turns out to be an important class of problems with a rich theory. For a large part of this section, we do not need to assume convexity.

According to Boyd and Vandenberge [BV04, 4.1.1], an optimization problem in standard form is given by

$$\begin{array}{ll} \text{minimize} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p \end{array} \tag{2.20}$$

The problem has domain $\mathcal{D} = (\cap_{i=0}^{m} \mathbf{dom}(f_i)) \cap (\cap_{i=1}^{p} \mathbf{dom}(h_i))$. We assume that $\mathcal{D}$ is open. You may think of $\mathcal{D}$ as equal to $\mathbb{R}^d$ in many cases.

A convex program arises when the f_i are convex functions, and the h_i are affine functions with domain $\mathbb{R}^d$. In this case, Observation 2.9 has the following consequences: the problem domain $\mathcal{D}$ is convex, and so are all sets of the form $\{\mathbf{x} \in \mathcal{D} : f_i(\mathbf{x}) \leq 0\}$ and $\{\mathbf{x} \in \mathcal{D} : h_i(\mathbf{x}) = 0\}$ (here we use that the h_i are affine). Also

$$X = \{\mathbf{x} \in \mathbb{R}^d : f_i(\mathbf{x}) \leq 0, i = 1, \ldots, m; \;\; h_i(\mathbf{x}) = 0, i = 1, \ldots, p\},$$

the *feasible region* of (2.20) is then a convex set. So we are in constrained minimization as discussed in Section 2.4.3, with a feasible region X induced by finitely many (in)equality constraints.

2.7.1 Lagrange duality

Lagrange duality is a powerful tool that (under suitable conditions) allows us to express the optimization problem (2.20) differently, and this dual view often provides new insights and may help us in solving the original problem. For example, we will see below that linear programming duality with its many applications is a special case of Lagrange duality.

Definition 2.45. *Given optimization problem (2.20), its* Lagrangian *is the function* $L : \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ *given by*

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}). \tag{2.21}$$

The λ_i, ν_i are called Lagrange multipliers.

The Lagrange dual function *is the function $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R} \cup \{-\infty\}$ defined by*

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \tag{2.22}$$

The fact that g can assume value $-\infty$ is not a pathology but typical. We will see that the "interesting" $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ are the ones for which $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) > -\infty$.

Let's discuss linear programming, the case where all involved functions are affine. Concretely, we consider a linear program of the form

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array} \tag{2.23}$$

This is of the form (2.20): the vector equation $A\mathbf{x} = \mathbf{b}$ summarizes the equality constraints induced by $h_i(\mathbf{x}) := \mathbf{a}_i^\top \mathbf{x} - b_i, i = 1, \ldots, p$, while the nonnegativity constraints come from $f_i(\mathbf{x}) := -x_i, i = 1, \ldots, m$. We finally have $f_0(\mathbf{x}) := \mathbf{c}^\top \mathbf{x}$.

Then the Lagrangian is

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathbf{c}^\top \mathbf{x} - \boldsymbol{\lambda}^\top \mathbf{x} + \boldsymbol{\nu}^\top (A\mathbf{x} - \mathbf{b}) = -\mathbf{b}^\top \boldsymbol{\nu} + (\mathbf{c}^\top - \boldsymbol{\lambda}^\top + \boldsymbol{\nu}^\top A)\mathbf{x}.$$

It follows that $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) > -\infty$ if and only if $\mathbf{c}^\top - \boldsymbol{\lambda}^\top + \boldsymbol{\nu}^\top A = \mathbf{0}$. And in this case, we have $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = -\mathbf{b}^\top \boldsymbol{\nu}$.

The significance of the Lagrangian dual function is that it provides a *lower bound* on the infimum value of (2.20), provided that $\boldsymbol{\lambda} \geq \mathbf{0}$. But a nontrivial lower bound is obtained only if $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) > -\infty$, this is why we are interested in this case.

Lemma 2.46 (Weak Lagrange duality)**.** *Let $\mathbf{x}$ be a feasible solution of the optimization problem (2.20), meaning that $f_i(\mathbf{x}) \leq 0$ for $i = 1, \ldots, m$ and $h_i(\mathbf{x}) = 0$ for $i = 1, \ldots, p$. Let g be the Lagrange dual function of (2.20) and $\boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\nu} \in \mathbb{R}^p$ such that $\boldsymbol{\lambda} \geq \mathbf{0}$. Then*

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\mathbf{x}).$$

Proof.

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \underbrace{\sum_{i=1}^m \lambda_i f_i(\mathbf{x})}_{\leq 0} + \underbrace{\sum_{i=1}^p \nu_i h_i(\mathbf{x})}_{=0} \leq f_0(\mathbf{x}).$$

□

It is natural to ask how $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ must be chosen such that we get the best lower bound. The answer is provided by the *Lagrange dual*.

Definition 2.47. *Let g be the Lagrange dual function of the optimization problem (2.20). Then the Lagrange dual of (2.20) is the optimization problem*

$$\begin{array}{ll} \textit{maximize} & g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \textit{subject to} & \boldsymbol{\lambda} \geq \mathbf{0}. \end{array} \tag{2.24}$$

By Lemma 2.46, the supremum value of the Lagrange dual (2.24) is a lower bound for the infimum value of the (primal) problem (2.20).

What is even nicer is that the Lagrange dual is a convex program, even if (2.20) is not! By this, we mean that the equivalent minimization problem

$$\begin{array}{ll} \text{minimize} & -g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \text{subject to} & \boldsymbol{\lambda} \geq \mathbf{0}. \end{array}$$

is a convex program in standard form (2.20). As there are no equality constraints, and the inequality constraints are obviously convex, we only need to show that the function $-g$ is a convex function. Here we have a slight problem, since $-g$ may also assume value ∞, and our Definition 2.11 of convexity does not cover this case. But this is easy to fix, and we defer the details (and the proof of convexity in this extended setting) to Exercise 18.

As an example, let's look at our linear program (2.23) again. We have already seen that its Lagrangian dual function satisfies

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \begin{cases} -\mathbf{b}^\top \boldsymbol{\nu} & \text{if } \mathbf{c}^\top - \boldsymbol{\lambda}^\top + \boldsymbol{\nu}^\top A = \mathbf{0}, \\ -\infty & \text{otherwise.} \end{cases}$$

So the Lagrange dual (2.24) becomes

$$\begin{array}{ll} \text{maximize} & -\mathbf{b}^\top \boldsymbol{\nu} \\ \text{subject to} & \mathbf{c}^\top + \boldsymbol{\nu}^\top A \geq \mathbf{0} \end{array}$$

Renaming $-\boldsymbol{\nu}$ to $\mathbf{y}$ and transposing the constraints, we arrive at the "standard" dual linear program

$$\begin{array}{ll} \text{maximize} & \mathbf{b}^\top \mathbf{y} \\ \text{subject to} & A^\top \mathbf{y} \leq \mathbf{c} \end{array} \tag{2.25}$$

In the case of linear programming, the primal (2.23) and the dual (2.25) have the same optimal value: $\inf \mathbf{c}^\top \mathbf{x} = \sup \mathbf{b}^\top \mathbf{y}$. It may happen that this

value is $-\infty$ (if the primal is unbounded and the dual is infeasible), or ∞ (if the primal is infeasible and the dual is unbounded). If the value is finite, it is attained in both the primal and the dual, so we actually have $\min \mathbf{c}^\top \mathbf{x} = \max \mathbf{b}^\top \mathbf{y}$.

This is the strong duality theorem of linear programming [MG07, Section 6.1]. It strenghtens weak duality which says that $\inf \mathbf{c}^\top \mathbf{x} \geq \sup \mathbf{b}^\top \mathbf{y}$.

For general *convex* programs (2.20), strong duality still holds (here, we *do* need convexity!), but some extra conditions are needed. There are a number of known sufficient conditions; these are usually named *constraint qualifications*. Here is a concrete result.

Theorem 2.48 ([BV04, 5.3.2])**.** *Suppose that (2.20) is a convex program with a feasible solution* $\tilde{\mathbf{x}}$ *that in additions satisfies* $f_i(\tilde{\mathbf{x}}) < 0, i = 1, \ldots, m$ *(a* Slater point*). Then the infimum value of the primal (2.20) equals the supremum value of its Lagrange dual (2.24). Moreover, if this value is finite, it is attained by a feasible solution of the dual. (2.24).*

Unlike in linear programming, a finite value is not necessarily attained by a feasible solution of the primal. So in the case of finite value, the theorem can be summarized as $\inf f_0(\mathbf{x}) = \max g(\boldsymbol{\lambda}, \boldsymbol{\nu})$; see Exercise 19 for an illustration of the theorem.

A common application of Lagrange duality is to turn the "hard" constraints of the optimization problem (2.20) into "soft" ones by moving them to the objective function. Instead of the constrained minimization problem (2.20), we consider (for some fixed $\boldsymbol{\lambda} \geq \mathbf{0}$ and fixed $\boldsymbol{\nu}$) the unconstrained minimization problem

$$\text{minimize} \quad f_0(\mathbf{x}) + \sum_{i=1}^m \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^p \nu_i h_i(\mathbf{x}). \tag{2.26}$$

As the objective function is the Lagrangian $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$, the infimum value of this unconstrained problem is by definition the value $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ of the Lagrange dual function. If we have strong Lagrange duality, there exist $\boldsymbol{\lambda} = \boldsymbol{\lambda}^\star \geq \mathbf{0}$ and $\boldsymbol{\nu} = \boldsymbol{\nu}^\star$ such that the unconstrained problem (2.26) has the same infimum as the constrained optimization problem (2.20). For *all* $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\nu}$, we know that the infimum of (2.26) provides a lower bound on the infimum of (2.20).

In practice, one could repeatedly solve (2.26), with a number of sensible candidates for $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\nu}$, and use the largest resulting value as an approximation of the infimum value of (2.20). Naturally, this approach

comes without any theoretical guarantees, unless we know that strong duality actually holds, and that "sensible" is quantifiable in some way.

Strong duality ($\inf f_0(\mathbf{x}) = \sup g(\boldsymbol{\lambda}, \boldsymbol{\nu})$) may also hold when there is no Slater point, or even when (2.20) is not a convex program. Theorem 2.48 simply provides one particular and very useful sufficient condition.

2.7.2 Karush-Kuhn-Tucker conditions

A case of particular interest is that strong duality holds and the joint value is attained in both the primal and the dual, meaning that $\min f_0(\mathbf{x}) = \max g(\boldsymbol{\lambda}, \boldsymbol{\nu})$. If the defining functions of the optimization problem (2.20) are differentiable, the Karush-Kuhn-Tucker conditions provide necessary and—under convexity—also sufficient conditions for this case to occur.

Definition 2.49 (Zero duality gap)**.** *Let $\tilde{\mathbf{x}}$ be feasible for the primal (2.20) and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ feasible for the Lagrange dual (2.24). The primal and dual solutions $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ are said to have* zero duality gap *if $f_0(\tilde{\mathbf{x}}) = g(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$.*

If $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ have zero duality gap, we have the following crucial chain of (in)equalities:

$$\begin{aligned} f_0(\tilde{\mathbf{x}}) &= g(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}}) \\ &= \inf_{\mathbf{x}\in\mathcal{D}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \tilde{\nu}_i h_i(\mathbf{x}) \right) \\ &\leq f_0(\tilde{\mathbf{x}}) + \sum_{i=1}^{m} \underbrace{\tilde{\lambda}_i f_i(\tilde{\mathbf{x}})}_{\leq 0} + \sum_{i=1}^{p} \underbrace{\tilde{\nu}_i h_i(\tilde{\mathbf{x}})}_{0} \\ &\leq f_0(\tilde{\mathbf{x}}). \end{aligned} \tag{2.27}$$

As a consequence, all inequalities in the box are actually equalities. From this we can draw two interesting consequences.

Lemma 2.50 (Complementary slackness)**.** *If $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ have zero duality gap, then*

$$\tilde{\lambda}_i f_i(\tilde{\mathbf{x}}) = 0, \quad i = 1, \ldots, m.$$

This is called complementary slackness, since if there is slack in the i-th inequality of the primal ($f_i(\tilde{\mathbf{x}}) < 0$), then there is no slack in the i-th inequality of the dual ($\tilde{\lambda}_i = 0$); and vice versa.

Lemma 2.51 (Vanishing Lagrangian gradient)**.** *If $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ have zero duality gap, and if all f_i and h_i are differentiable, then*

$$\nabla f_0(\tilde{\mathbf{x}}) + \sum_{i=1}^{m} \tilde{\lambda}_i \nabla f_i(\tilde{\mathbf{x}}) + \sum_{i=1}^{p} \tilde{\nu}_i \nabla h_i(\tilde{\mathbf{x}}) = \mathbf{0}.$$

Proof. According to the (in)equality in the third line of (2.27), $\tilde{\mathbf{x}}$ minimizes the differentiable function

$$f_0(\mathbf{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \tilde{\nu}_i h_i(\mathbf{x}),$$

and hence its gradient vanishes by Lemma 2.23. □

In summary, we get the following result.

Theorem 2.52 (Karush-Kuhn-Tucker necessary conditions)**.** *Let $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ be feasible solutions of the primal optimization problem (2.20) and its Lagrange dual (2.24), respectively, with zero duality gap. If all f_i and h_i in (2.20) are differentiable, then*

$$\tilde{\lambda}_i f_i(\tilde{\mathbf{x}}) = 0, \quad i = 1, \ldots, m, \tag{2.28}$$

$$\nabla f_0(\tilde{\mathbf{x}}) + \sum_{i=1}^{m} \tilde{\lambda}_i \nabla f_i(\tilde{\mathbf{x}}) + \sum_{i=1}^{p} \tilde{\nu}_i \nabla h_i(\tilde{\mathbf{x}}) = \mathbf{0}. \tag{2.29}$$

Theorem 2.53 (Karush-Kuhn-Tucker sufficient conditions)**.** *Let $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ be feasible solutions of the primal optimization problem (2.20) and its Lagrange dual (2.24).*

Further suppose that all f_i and h_i in (2.20) are differentiable, all f_i are convex, all h_i are affine, and (2.28) as well as (2.29) hold. Then $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ have zero duality gap.

Proof. If we can establish the chain of equalities in (2.27), the statement follows. The equality in the second line is the definition of the Lagrange dual function; in the third line, we obtain equality through the vanishing

Lagrangian gradient (2.29) along with Lemma 2.22, showing that $\tilde{\mathbf{x}}$ minimizes the function $f(\mathbf{x}) := L(\mathbf{x}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$. This is convex by Lemma 2.19 (i). The inequality under the brace in the third line is complementary slackness (2.28), and from this, equality in the fourth line follows. □

The Karush-Kuhn-Tucker conditions (primal and dual feasibilty, complementary slackness, vanishing Lagrangian gradient) can be of significant help in solving the primal problem (2.20). If we can find $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\nu}})$ satisfying them, we know that $\tilde{\mathbf{x}}$ is a minimizer of (2.20). This may be easier to "solve" than (2.20) itself.

However, we cannot always count on the Karush-Kuhn-Tucker conditions being solvable. Theorem 2.52 guarantees them only if there are primal and dual solutions of zero duality gap. But if the primal has a Slater point, then $\inf f_0(\mathbf{x}) = \max g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ by Theorem 2.48, and in this case, the Karush-Kuhn-Tucker conditions are indeed equivalent to the existence of a minimizer of (2.20).

2.7.3 Computational complexity

By a celebrated result of Khachyian from 1979, linear programs can be solved in polynomial time, using the (impractical) *ellipsoid method* [Kha80]. In 1984, Karmarkar showed that *interior point methods* provide a new and practical approach to solve linear programs in polynomial time [Kar84].

The scope of both methods is not restricted to linear programming; in fact, they had been developed before in order to solve convex programs. Still, the fact that they result in polynomial-time methods when applied to linear programming was a significant insight.

By design, both methods are iterative and can achieve any desired optimization error $\varepsilon > 0$ as introduced in Section 1.11. But they will usually never find the true minimizer, even if it exists. But in the case of linear programming (and some other "benign" convex optimization problems), there is a finite set of candidate solutions guaranteed to contain a minimizer. In linear programming, these are the basic feasible solutions. Moreover, from *any* feasible solution one can easily find a candidate solution that is not worse. Finally, one argues that if ε is small enough (depending on the description size of the problem), the only candidate solutions one can still find in this way are minimizers.

In general convex programs, this approach does not work. In particular, there may not be any minimizers, even if the program's infimum value is finite; see Exercise 19. Still, one can try to understand how long the methods take in order to reach optimization error at most ε. We argued in Section 1.11 that this is enough for most Data Science purposes.

For convex programs and interior point methods, this has been pioneered by Nesterov and Nemirovskii in their 1994 book [NN94]. Boyd and Vandenberghe present the full theory in Section 11 of their book [BV04]. Here we only give (part of) the high-level summary of the state of affairs [BV04, 11.5.5].

It is important to say upfront that the analysis does not work for arbitrary convex programs of the form (2.20); some assumptions are needed. Mainly, the function f_0 to be minimized has to be *self-concordant* which is a condition involving its second- and third-order derivatives. Also, we need some bound M on the maximum value of an optimal solution.

In a first phase, one needs to find a feasible solution, and the runtime of this phase inversely depends on how close the problem is to being infeasible. In the second phase, the number of iterations is of the order

$$O\left(\sqrt{m}\log\left(\frac{M-p^\star}{\varepsilon}\right)\right),$$

where $p^\star$ is the infimum value of (2.20). The bound dos not depend on p, the number of equality constraints, and also not on d, the number of variables. These two problem dimensions appear in the complexity of the individual iterations, but we will not go into this.

What we can say, though, is that the individual iterations are computationally very heavy, making them unsuitable for large-scale learning where optimization time is a bottleneck.

Hence, it makes sense to consider *simple* algorithms with *low computational cost per step*, even if they need more iterations than the best possible algorithms. This is the approach that we will take in the subsequent chapters.

2.8 Exercises

Exercise 6. *Prove that a differentiable function is continuous!*

Exercise 7. *Prove Jensen's inequality (Lemma 2.13)!*

Exercise 8. *Prove that a convex function (with $\mathbf{dom}(f)$ open) is continuous (Lemma 2.14)!*

Hint: *First prove that a convex function f is bounded on any cube $C = [l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_d, u_d] \subseteq \mathbf{dom}(f)$, with the maximum value occurring on some corner of the cube (a point $\mathbf{z}$ such that $z_i \in \{l_i, u_i\}$ for all i). Then use this fact to show that—given $\mathbf{x} \in \mathbf{dom}(f)$ and $\varepsilon > 0$—all $\mathbf{y}$ in a sufficiently small ball around $\mathbf{x}$ satisfy $|f(\mathbf{y}) - f(\mathbf{x})| < \varepsilon$.*

Exercise 9. *Prove that the function $d_{\mathbf{y}} : \mathbb{R}^d \to \mathbb{R}$, $\mathbf{x} \mapsto \|\mathbf{x} - \mathbf{y}\|^2$ is strictly convex for any $\mathbf{y} \in \mathbb{R}^d$. (Use Lemma 2.25.)*

Exercise 10. *Prove Lemma 2.19! Can (ii) be generalized to show that for two convex functions f, g, the function $f \circ g$ is convex as well?*

Exercise 11. *Prove Lemmata 2.38 and 2.39!*

Exercise 12. *Consider the function ℓ defined in (2.15). Prove that ℓ is convex!*

Exercise 13. *Consider the function ℓ defined in (2.15). Let us call an argument matrix W a* separator *for P if for all $\mathbf{x} \in P$,*

$$(W\mathbf{x})_{d(\mathbf{x})} = \max_{j=0}^{9} (W\mathbf{x})_j,$$

i.e. under (2.14), the correct digit has highest probability (possibly along with other digits). A separator is trivial *if for all $\mathbf{x} \in P$ and all $i, j \in \{0, \ldots, 9\}$,*

$$(W\mathbf{x})_i = (W\mathbf{x})_j.$$

For example, whenever the rows of W are pairwise identical, we obtain a trivial separator. But depending on the data, there may be other trivial separators. For example, if some pixel is black (gray value 0) in all images, arbitrarily changing the entries in the corresponding column of a trivial separator gives us another trivial separator. For a trivial separator W, (2.15) yields $\ell(W) = |P| \ln 10$.

Prove the following statement: ℓ has a global minimum if and only if all separators are trivial.

As a special case, consider the situation in which there exists a strong *(and in particular nontrivial) separator: a matrix $W^\star$ such that for all $\mathbf{x} \in P$ and all $j \neq d(\mathbf{x})$,*

$$(W^\star\mathbf{x})_{d(\mathbf{x})} > (W^\star\mathbf{x})_j,$$

i.e. the correct digit has unique highest probability. In this case, it is easy to see that $\ell(\lambda W^\star) \to_{\lambda\to\infty} 0$, so we cannot have a global minimum, as $\inf_W(\ell(W)) = 0$ is not attainable.

Exercise 14. *Prove that the function $f(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$ (ℓ_1-norm) is convex!*

Exercise 15. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be twice differentiable. For fixed $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$, consider the univariate function $h(t) = f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$ over a suitable open interval $\mathbf{dom}(h) \supseteq [0, 1]$ such that $\mathbf{x} + t(\mathbf{y} - \mathbf{x}) \in \mathbf{dom}(f)$ for all $t \in \mathbf{dom}(h)$. Let us abbreviate $\mathbf{v} = \mathbf{y} - \mathbf{x}$. We already know that $h'(t) = \nabla f(\mathbf{x} + t\mathbf{v})^\top \mathbf{v}$ for $t \in \mathbf{dom}(h)$. Prove that*

$$h''(t) = \mathbf{v}^\top \nabla^2 f(\mathbf{x} + t\mathbf{v})\mathbf{v}, \quad t \in \mathbf{dom}(h).$$

Exercise 16. *A seminorm is a function $f : \mathbb{R}^d \to \mathbb{R}$ satisfying the following two properties for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and all $\lambda \in \mathbb{R}$.*

(i) $f(\lambda\mathbf{x}) = |\lambda| f(\mathbf{x})$,

(ii) $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ *(triangle inequality).*

Prove that every seminorm is convex!

Exercise 17. *Suppose that we have centered observations $(\mathbf{x}_i, y_i)$ such that $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}, \sum_{i=1}^n y_i = 0$. Let $w_0^\star, \mathbf{w}^\star$ be the global minimum of the least squares objective*

$$f(w_0, \mathbf{w}) = \sum_{i=1}^n (w_0 + \mathbf{w}^\top \mathbf{x}_i - y_i)^2.$$

Prove that $w_0^\star = 0$. Also, suppose $\mathbf{x}_i'$ and y_i' are such that for all i, $\mathbf{x}_i' = \mathbf{x}_i + \mathbf{q}$, $y_i' = y_i + r$. Show that $(w_0, \mathbf{w})$ minimizes f if and only if $(w_0 - \mathbf{w}^\top \mathbf{q} + r, \mathbf{w})$ minimizes

$$f'(w_o, \mathbf{w}) = \sum_{i=1}^n (w_0 + \mathbf{w}^\top \mathbf{x}_i' - y_i')^2.$$

Exercise 18. *A function $f : \mathbf{dom}(f) \to \mathbb{R} \cup \{\infty\}$ is called convex if $\mathbf{dom}(f)$ is convex and the inequality (2.2) defining convexity holds when $f(\mathbf{x}), f(\mathbf{y}) < \infty$. This in particular implies that the "finite domain" $\{\mathbf{x} \in \mathbf{dom}(f) : f(\mathbf{x}) < \infty\}$ is convex as well (if it wasn't, we could construct $\mathbf{x}, \mathbf{y}$ in the finite domain and*

some convex combination $\lambda\mathbf{x}+(1-\lambda)\mathbf{y} \in \mathbf{dom}(f)$ *with* $f(\lambda\mathbf{x}+(1-\lambda)\mathbf{y}) = \infty$*, violating convexity).*

Prove that the negative of the Lagrangian dual function $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R} \cup \{-\infty\}$ *as in Definition 2.45 is convex in the above sense.*

Exercise 19. *Consider the optimization problem*

$$\begin{array}{ll} \textit{minimize} & x_2 - x_1 \\ \textit{subject to} & \sqrt{x_1^2+1} - x_2 \leq 0. \end{array}$$

(i) Prove that this is a convex program with a Slater point so that the conditions of Theorem 2.48 are satisfied!

(ii) Compute the Lagrange dual function g*, the maximum value* γ *of the Lagrange dual problem and a feasible solution of value* γ*.*

(iii) Show that the primal problem does not attain its infimum value (which is also γ *by Theorem 2.48)!*

Chapter 3

Gradient Descent

Contents

3.1 Overview

The gradient descent algorithm (including variants such as projected or stochastic gradient descent) is the most useful workhorse for minimizing loss functions in practice. The algorithm is extremely simple and surprisingly robust in the sense that it also works well for many loss functions that are not convex. While it is easy to construct (artificial) non-convex functions on which gradient descent goes completely astray, such functions do not seem to be typical in practice; however, understanding this on a theoretical level is an open problem, and only few results exist in this direction.

The vast majority of theoretical results concerning the performance of gradient descent hold for convex functions only. In this and the following chapters, we will present some of these results, but maybe more importantly, the main ideas behind them. As it turns out, the number of ideas that we need is rather small, and typically, they are shared between different results. Our approach is therefore to fully develop each idea once, in the context of a concrete result. If the idea reappears, we will typically only discuss the changes that are necessary in order to establish a new result from this idea. In order to avoid boredom from ideas that reappear too often, we omit other results and variants that one could also get along the lines of what we discuss.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex and differentiable function. We also assume that f has a global minimum $\mathbf{x}^\star$, and the goal is to find (an approximation of) $\mathbf{x}^\star$. This usually means that for a given $\varepsilon > 0$, we want to find $\mathbf{x} \in \mathbb{R}^d$ such that

$$f(\mathbf{x}) - f(\mathbf{x}^\star) < \varepsilon.$$

Notice that we are not making an attempt to get near to $\mathbf{x}^\star$ itself — there can be several minima $\mathbf{y}^\star \neq \mathbf{x}^\star$ with $f(\mathbf{x}^\star) = f(\mathbf{y}^\star)$.

Gradient descent is an *iterative* method, meaning that it generates a sequence $\mathbf{x}_0, \mathbf{x}_2, \ldots$ of solutions such that in some iteration T, we eventually have $f(\mathbf{x}_T) - f(\mathbf{x}^\star) < \varepsilon$.

Table 3.1 gives an overview of the results that we will prove. They concern several variants of gradient descent as well as several classes of functions. The significance of each algorithm and function class will briefly be discussed when it first appears.

In Chapter 6, we will also look at gradient descent on functions that

	Lipschitz convex functions	smooth convex functions	strongly convex functions	smooth & strongly convex functions
gradient descent	Thm. 3.1 $\mathcal{O}(1/\varepsilon^2)$	Thm. 3.8 $\mathcal{O}(1/\varepsilon)$		Thm. 3.14 $\mathcal{O}(\log(1/\varepsilon))$
accelerated gradient descent		Thm. 3.9 $\mathcal{O}(1/\sqrt{\varepsilon})$		
projected gradient descent	Thm. 4.2 $\mathcal{O}(1/\varepsilon^2)$	Thm. 4.4 $\mathcal{O}(1/\varepsilon)$		Thm. 4.5 $\mathcal{O}(\log(1/\varepsilon))$
subgradient descent	Thm. **??** $\mathcal{O}(1/\varepsilon^2)$		Thm. **??** $\mathcal{O}(1/\varepsilon)$	
stochastic gradient descent	Thm. **??** $\mathcal{O}(1/\varepsilon^2)$		Thm. **??** $\mathcal{O}(1/\varepsilon)$	

Table 3.1: Results on gradient descent. Below each theorem, the number of steps is given which the respective variant needs on the respective function class to achieve additive approximation error at most ε.

are not convex. In this case, provably small approximation error can still be obtained for some particularly well-behaved functions (we will give an example). For smooth (but not necessarily convex) functions, we generally cannot show convergence in error, but a (much) weaker convergence property still holds.

3.1.1 Convergence rates

You sometimes hear terms such as *linear* convergence, *quadratic* convergence, or *sublinear* convergence. They refer to iterative optimization methods and describe how quickly the error "goes down" from one iteration to the next. Let ε_t denote the error in iteration t; in the context of minimization , we often consider $\varepsilon_t = f(\mathbf{x}_t) - f(\mathbf{x}^\star)$, but there could be other error measures. An algorithm is said to exhibit (at least) *linear convergence*

whenever there is a real number $0 < c < 1$ such that

$$\varepsilon_{t+1} \leq c\varepsilon_t \quad \text{for all sufficiently large } t.$$

The word *linear* comes from the fact that the error in step $t+1$ is bounded by a linear function of the error in step t.

This means that for t large enough, the error goes down by at least a constant factor in each step. Linear convergence implies that an error of at most ε is achieved within $\mathcal{O}(\log(1/\varepsilon))$ iterations. For example, this is the bound provided by Theorem 3.14 (last entry in the first row of Table 3.1), and it is proved by showing linear convergence of the algorithm.

The term *superlinear* convergence refers to an algorithm for which there are constants $r > 1$ and $c > 0$ such that

$$\varepsilon_{t+1} \leq c(\varepsilon_t)^r \quad \text{for all sufficiently large } t.$$

The case $r = 2$ is known as *quadratic* convergence. Under quadratic convergence, an error of at most ε is achieved within $\mathcal{O}(\log\log(1/\varepsilon))$ iterations. We will see an algorithm with quadratic convergence in Chapter 8.

If a (converging) algorithm does not exhibit at least linear convergence, we say that it has *sublinear* convergence. One can also quantify sublinear convergence more precisely if needed.

3.2 The algorithm

Gradient descent is a very simple iterative algorithm for finding the desired approximation $\mathbf{x}$, under suitable conditions that we will get to. It computes a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ of vectors such that $\mathbf{x}_0$ is arbitrary, and for each $t \geq 0$, $\mathbf{x}_{t+1}$ is obtained from $\mathbf{x}_t$ by making a step of $\mathbf{v}_t \in \mathbb{R}^d$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_t.$$

How do we choose $\mathbf{v}_t$ in order to get closer to optimality, meaning that $f(\mathbf{x}_{t+1}) < f(\mathbf{x}_t)$?

From differentiability of f at $\mathbf{x}_t$ (Definition 2.5), we know that for $\|\mathbf{v}_t\|$ tending to 0,

$$f(\mathbf{x}_t + \mathbf{v}_t) = f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top \mathbf{v}_t + \underbrace{r(\mathbf{v}_t)}_{o(\|\mathbf{v}_t\|)} \approx f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top \mathbf{v}_t.$$

To get any decrease in function value at all, we have to choose $\mathbf{v}_t$ such that $\nabla f(\mathbf{x}_t)^\top \mathbf{v}_t < 0$. But among all steps $\mathbf{v}_t$ of the same length, we should in fact choose the one with the most negative value of $\nabla f(\mathbf{x}_t)^\top \mathbf{v}_t$, so that we maximize our decrease in function value. This is achieved when $\mathbf{v}_t$ points into the direction of the negative gradient $-\nabla f(\mathbf{x}_t)$. But as differentiability guarantees decrease only for small steps, we also want to control how far we go along the direction of the negative gradient.

Therefore, the step of gradient descent is defined by

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t). \tag{3.1}$$

Here, $\gamma > 0$ is a fixed *stepsize*, but it may also make sense to have γ depend on t. For now, γ is fixed. We hope that for some reasonably small integer t, in the t-th iteration we get that $f(\mathbf{x}_t) - f(\mathbf{x}^\star) < \varepsilon$; see Figure 3.1 for an example.

Now it becomes clear why we are assuming that $\mathbf{dom}(f) = \mathbb{R}^d$: The update step (3.1) may in principle take us "anywhere", so in order to get a well-defined algorithm, we want to make sure that f is defined and differentiable everywhere.

The choice of γ is critical for the performance. If γ is too small, the process might take too long, and if γ is too large, we are in danger of overshooting. It is not clear at this point whether there is a "right" stepsize.

3.3 Vanilla analysis

The first-order characterization of convexity provides us with a way to bound terms of the form $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$: With $\mathbf{x} = \mathbf{x}_t, \mathbf{y} = \mathbf{x}^\star$, (2.3) gives us

$$f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^\star). \tag{3.2}$$

So we have reduced the problem to the one of bounding $f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}^\star)$, and this is what we do next.

Let $\mathbf{x}_t$ be some iterate in the sequence (3.1). We abbreviate $\mathbf{g}_t := \nabla f(\mathbf{x}_t)$. By definition of gradient descent (3.1), $\mathbf{g}_t = (\mathbf{x}_t - \mathbf{x}_{t+1})/\gamma$, hence

$$\mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^\star) = \frac{1}{\gamma} (\mathbf{x}_t - \mathbf{x}_{t+1})^\top (\mathbf{x}_t - \mathbf{x}^\star). \tag{3.3}$$

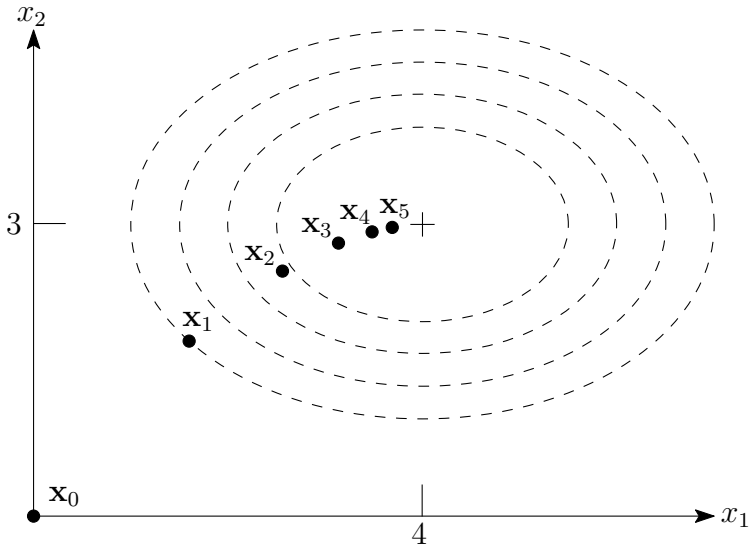


Figure 3.1: Example run of gradient descent on the quadratic function $f(x_1, x_2) = 2(x_1 - 4)^2 + 3(x_2 - 3)^2$ with global minimum $(4, 3)$; we have chosen $\mathbf{x}_0 = (0, 0), \gamma = 0.1$; dashed lines represent level sets of f (points of constant f-value)

Now we apply (somewhat out of the blue, but this will clear up in the next step) the basic vector equation $2\mathbf{v}^\top \mathbf{w} = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - \|\mathbf{v} - \mathbf{w}\|^2$ (a.k.a. the cosine theorem) to rewrite the same expression as

$$\begin{aligned} \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^\star) &= \frac{1}{2\gamma} \left(\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \right) \\ &= \frac{1}{2\gamma} \left(\gamma^2 \|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \right) \\ &= \frac{\gamma}{2} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \left(\|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \right) \end{aligned} \tag{3.4}$$

Next we sum this up over the iterations t, so that the latter two terms in

the bracket cancel in a telescoping sum.

$$\begin{aligned}\sum_{t=0}^{T-1} \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^\star) &= \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \left(\|\mathbf{x}_0 - \mathbf{x}^\star\|^2 - \|\mathbf{x}_T - \mathbf{x}^\star\|^2 \right) \\ &\leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^\star\|^2 \end{aligned} \tag{3.5}$$

Now we recall from (3.2) that

$$f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \mathbf{g}_t^\top (\mathbf{x}_t - \mathbf{x}^\star).$$

Hence we further obtain

$$\sum_{t=0}^{T-1} (f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\mathbf{g}_t\|^2 + \frac{1}{2\gamma} \|\mathbf{x}_0 - \mathbf{x}^\star\|^2. \tag{3.6}$$

This gives us an upper bound for the *average* error $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$, $t = 0, \ldots, T-1$, hence in particular for the error incurred by the iterate with the smallest function value. The last iterate is not necessarily the best one: gradient descent with fixed stepsize γ will in general also make steps that overshoot and actually increase the function value; see Exercise 23(i).

The question is of course: is this result any good? In general, the answer is no. A dependence on $\|\mathbf{x}_0 - \mathbf{x}^\star\|$ is to be expected (the further we start from $\mathbf{x}^\star$, the longer we will take); the dependence on the squared gradients $\|\mathbf{g}_t\|^2$ is more of an issue, and if we cannot control them, we cannot say much.

3.4 Lipschitz convex functions: $\mathcal{O}(1/\varepsilon^2)$ steps

Here is the cheapest "solution" to squeeze something out of the vanilla analysis (3.5): let us simply assume that all gradients of f are bounded in norm. Equivalently, such functions are Lipschitz continuous over $\mathbb{R}^d$ by Theorem 2.10. (A small subtetly here is that in the situation of real-valued functions, Theorem 2.10 is talking about the spectral norm of the $(1 \times d)$-matrix (or row vector) $\nabla f(\mathbf{x})^\top$, while below, we are talking about the Euclidean norm of the (column) vector $\nabla f(\mathbf{x})$; but these two norms are the same; see Exercise 20.)

Assuming bounded gradients rules out many interesting functions, though. For example, $f(x) = x^2$ (a supermodel in the world of convex functions) already doesn't qualify, as $\nabla f(x) = 2x$—and this is unbounded as x tends to infinity. But let's care about supermodels later.

Theorem 3.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$ and $\|\nabla f(\mathbf{x})\| \leq B$ for all $\mathbf{x}$. Choosing the stepsize*

$$\gamma := \frac{R}{B\sqrt{T}},$$

gradient descent (3.1) yields

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{RB}{\sqrt{T}}.$$

Proof. This is a simple calculation on top of (3.6): after plugging in the bounds $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$ and $\|\mathbf{g}_t\| \leq B$, we get

$$\sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{\gamma}{2}B^2T + \frac{1}{2\gamma}R^2,$$

so want to choose γ such that

$$q(\gamma) = \frac{\gamma}{2}B^2T + \frac{R^2}{2\gamma}$$

is minimized. Setting the derivative to zero yields the above value of γ, and $q(R/(B\sqrt{T})) = RB\sqrt{T}$. Dividing by T, the result follows. □

This means that in order to achieve $\min_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \varepsilon$, we need

$$T \geq \frac{R^2B^2}{\varepsilon^2}$$

many iterations. This is not particularly good when it comes to concrete numbers (think of desired error $\varepsilon = 10^{-6}$ when R, B are somewhat larger). On the other hand, the number of steps does not depend on d, the dimension of the space. This is very important since we often optimize in high-dimensional spaces. Of course, R and B may depend on d, but in many relevant cases, this dependence is mild.

What happens if we don't know R and/or B? An idea is to "guess" R and B, run gradient descent with T and γ resulting from the guess, check whether the result has absolute error at most ε, and repeat with a different guess otherwise. This fails, however, since in order to compute the absolute error, we need to know $f(\mathbf{x}^\star)$ which we typically don't. But Exercise 24 asks you to show that knowing R is sufficient.

3.5 Smooth convex functions: $\mathcal{O}(1/\varepsilon)$ steps

Our workhorse in the vanilla analysis was the first-order characterization of convexity: for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$, we have

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}). \tag{3.7}$$

Next we want to look at functions for which $f(\mathbf{y})$ can be bounded *from above* by $f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$, up to at most quadratic error. The following definition applies to all differentiable functions, convexity is not required.

Definition 3.2. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be a differentiable function, $X \subseteq \mathbf{dom}(f)$ convex and $L \in \mathbb{R}_+$. Function f is called* smooth *(with parameter L) over X if*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X. \tag{3.8}$$

If $X = \mathbf{dom}(f)$, f is simply called smooth.

Recall that (3.7) says that for any $\mathbf{x}$, the graph of f is above its tangential hyperplane at $(\mathbf{x}, f(\mathbf{x}))$. In contrast, (3.8) says that for any $\mathbf{x} \in X$, the graph of f is below a not-too-steep tangential paraboloid at $(\mathbf{x}, f(\mathbf{x}))$; see Figure 3.2.

This notion of smoothness has become standard in convex optimization, but the naming is somewhat unfortunate, since there is an (older) definition of a smooth function in mathematical analysis where it means a function that is infinitely often differentiable.

We have the following simple characterization of smoothness.

Lemma 3.3 (Exercise 21)**.** *Suppose that $\mathbf{dom}(f)$ is open and convex, and that $f : \mathbf{dom}(f) \to \mathbb{R}$ is differentiable. Let $L \in \mathbb{R}_+$. Then the following two statements are equivalent.*

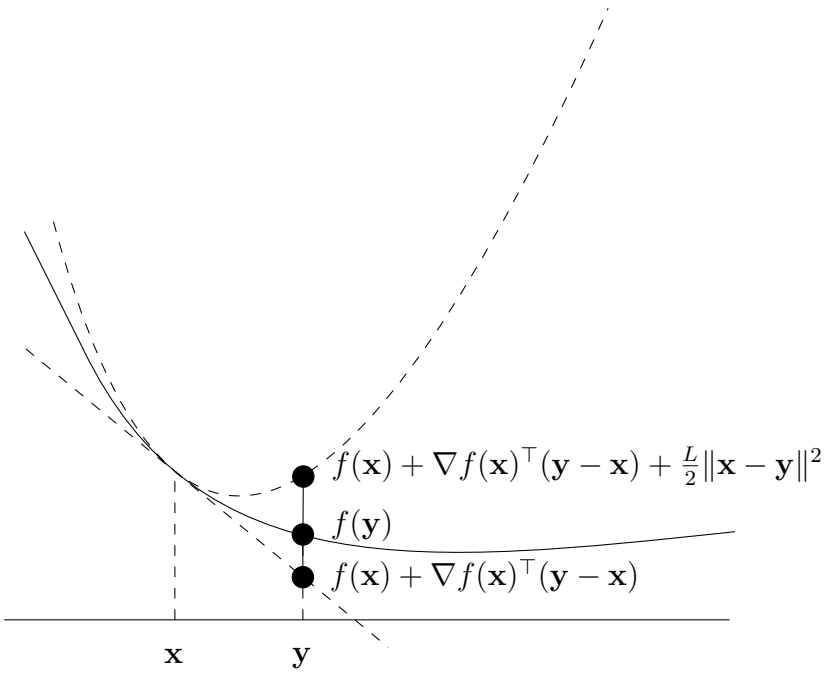


Figure 3.2: A smooth convex function

(i) f *is smooth with parameter* L*.*

(ii) g *defined by* $g(\mathbf{x}) = \frac{L}{2}\mathbf{x}^\top\mathbf{x} - f(\mathbf{x})$ *is convex over* $\mathbf{dom}(g) := \mathbf{dom}(f)$*.*

Let us discuss some cases. If $L = 0$, (3.7) and (3.8) together require that

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y}-\mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathbf{dom}(f),$$

meaning that f is an affine function. A simple calculation shows that our supermodel function $f(x) = x^2$ is smooth with parameter $L = 2$:

$$\begin{aligned} f(y) = y^2 &= x^2 + 2x(y-x) + (x-y)^2 \\ &= f(x) + f'(x)(y-x) + \frac{L}{2}(x-y)^2. \end{aligned}$$

More generally, we also claim that all quadratic functions of the form $f(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} + \mathbf{b}^\top\mathbf{x} + c$ are smooth, where Q is a $(d \times d)$ matrix, $\mathbf{b} \in \mathbb{R}^d$ and $c \in \mathbb{R}$. Because $\mathbf{x}^\top Q\mathbf{x} = \mathbf{x}^\top Q^\top\mathbf{x}$, we get that $f(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} = \frac{1}{2}\mathbf{x}^\top(Q +$

$Q^\top)\mathbf{x}$, where $\frac{1}{2}(Q + Q^\top)$ is symmetric. Therefore, we can assume without loss of generality that Q is symmetric, i.e., it suffices to show that quadratic functions defined by symmetric functions are smooth.

Lemma 3.4 (Exercise 22). *Let $f(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} + \mathbf{b}^\top\mathbf{x} + c$, where Q is a symmetric $(d \times d)$ matrix, $\mathbf{b} \in \mathbb{R}^d, c \in \mathbb{R}$. Then f is smooth with parameter $2\|Q\|$, where $\|Q\|$ is the spectral norm of Q (Definition 2.2).*

The (univariate) convex function $f(x) = x^4$ is not smooth (over $\mathbb{R}$): at $x = 0$, condition (3.8) reads as

$$y^4 \leq \frac{L}{2}y^2,$$

and there is obviously no L that works for all y. The function is smooth, however, over any bounded set X (Exercise 27).

In general—and this is the important message here—only functions of asymptotically at most quadratic growth can be smooth. It is tempting to believe that any such "subquadratic" function is actually smooth, but this is not true. Exercise 23(iii) provides a counterexample.

While bounded gradients are equivalent to Lipschitz continuity of f (Theorem 2.10), smoothness turns out to be equivalent to Lipschitz continuity of ∇f—if f is convex over the whole space. In general, Lipschitz continuity of ∇f implies smoothness, but not the other way around.

Lemma 3.5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable. The following two statements are equivalent.*

(i) f is smooth with parameter L.

(ii) $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

We will derive the direction (ii)⇒(i) as Lemma 6.1 in Chapter 6 (which neither requires convexity nor domain $\mathbb{R}^d$). The other direction is a bit more involved. A proof of the equivalence can be found in the lecture slides of L. Vandenberghe, `http://www.seas.ucla.edu/~vandenbe/236C/lectures/gradient.pdf`.

The operations that we have shown to preserve convexity (Lemma 2.19) also preserve smoothness. This immediately gives us a rich collection of smooth functions.

Lemma 3.6 (Exercise 25)**.**

(i) Let $f_1, f_2, \ldots, f_m$ be smooth with parameters $L_1, L_2, \ldots, L_m$, and let $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{R}_+$. Then the function $f := \sum_{i=1}^m \lambda_i f_i$ is smooth with parameter $\sum_{i=1}^m \lambda_i L_i$ over $\mathbf{dom}(f) := \bigcap_{i=1}^m \mathbf{dom}(f_i)$.

(ii) Let $f : \mathbf{dom}(f) \to \mathbb{R}$ with $\mathbf{dom}(f) \subseteq \mathbb{R}^d$ be smooth with parameter L, and let $g : \mathbb{R}^m \to \mathbb{R}^d$ be an affine function, meaning that $g(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$, for some matrix $A \in \mathbb{R}^{d \times m}$ and some vector $\mathbf{b} \in \mathbb{R}^d$. Then the function $f \circ g$ (that maps $\mathbf{x}$ to $f(A\mathbf{x} + \mathbf{b})$) is smooth with parameter $L\|A\|^2$ on $\mathbf{dom}(f \circ g) := \{\mathbf{x} \in \mathbb{R}^m : g(\mathbf{x}) \in \mathbf{dom}(f)\}$, where $\|A\|$ is the spectral norm of A (Definition 2.2).

We next show that for smooth convex functions, the vanilla analysis provides a better bound than it does under bounded gradients. In particular, we are now able to serve the supermodel $f(x) = x^2$.

We start with a preparatory lemma showing that gradient descent (with suitable stepsize γ) makes progress in function value on smooth functions in every step. We call this *sufficient decrease*, and maybe surprisingly, it does not require convexity.

Lemma 3.7 (Sufficient decrease)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and smooth with parameter L according to (3.8). With*

$$\gamma := \frac{1}{L},$$

gradient descent (3.1) satisfies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0.$$

More specifically, this already holds if f is smooth with parameter L over the line segment connecting $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$.

Proof. We apply the smoothness condition (3.8) and the definition of gradient descent that yields $\mathbf{x}_{t+1} - \mathbf{x}_t = -\nabla f(\mathbf{x}_t)/L$. We compute

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\ &= f(\mathbf{x}_t) - \frac{1}{L}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 \\ &= f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2. \end{aligned}$$

□

Theorem 3.8. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that f is smooth with parameter L according to (3.8). Choosing stepsize*

$$\gamma := \frac{1}{L},$$

gradient descent (3.1) yields

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{L}{2T}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2, \quad T > 0.$$

Proof. We apply sufficient decrease (Lemma 3.7) to bound the sum of the $\|\mathbf{g}_t\|^2 = \|\nabla f(\mathbf{x}_t)\|^2$ after step (3.6) of the vanilla analysis as follows:

$$\frac{1}{2L}\sum_{t=0}^{T-1}\|\nabla f(\mathbf{x}_t)\|^2 \leq \sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})) = f(\mathbf{x}_0) - f(\mathbf{x}_T). \tag{3.9}$$

With $\gamma = 1/L$, (3.6) then yields

$$\begin{aligned}\sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) &\leq \frac{1}{2L}\sum_{t=0}^{T-1}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2 \\ &\leq f(\mathbf{x}_0) - f(\mathbf{x}_T) + \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2,\end{aligned}$$

equivalently

$$\sum_{t=1}^{T}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2. \tag{3.10}$$

Because $f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t)$ for each $0 \leq t \leq T$ by Lemma 3.7, by taking the average we get that

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{1}{T}\sum_{t=1}^{T}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{L}{2T}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2.$$

□

This improves over the bounds of Theorem 3.1. With $R^2 := \|\mathbf{x}_0 - \mathbf{x}^\star\|^2$, we now only need

$$T \geq \frac{R^2 L}{2\varepsilon}$$

iterations instead of R^2B^2/ε^2 to achieve absolute error at most ε.

Exercise 26 shows that we do not need to know L to obtain the same asymptotic runtime.

Interestingly, the bound in Theorem 3.8 can be improved—but not by much. Fixing L and $R = \|\mathbf{x}_0 - \mathbf{x}^\star\|$, the bound is of the form $O(1/T)$. Lee and Wright have shown that a better upper bound of $o(1/T)$ holds, but that for any fixed $\delta > 0$, a lower bound of $\Omega(1/T^{1+\delta})$ also holds [LW19].

3.6 Acceleration for smooth convex functions: $\mathcal{O}(1/\sqrt{\varepsilon})$ steps

Let's take a step back, forget about gradient descent for a moment, and just think about what we actually use the algorithm for: we are minimizing a differentiable convex function $f : \mathbb{R}^d \to \mathbb{R}$, where we are assuming that we have acccess to the gradient vector $\nabla f(\mathbf{x})$ at any given point $\mathbf{x}$.

But is it clear that gradient descent is the best algorithm for this task? After all, it is just *some* algorithm that is using gradients to make progress locally, but there might be other (and better) such algorithms. Let us define a *first-order method* as an algorithm that only uses gradient information to minimize f. More precisely, we allow a first-order method to access f only via an oracle that is able to return values of f and ∇f at arbitrary points. Gradient descent is then just a specific first-order method.

For any class of convex functions, one can then ask a natural question: What is the best first-order method for the function class, the one that needs the smallest number of oracle calls in the worst case, as a function of the desired error ε? In particular, is there a method that asymptotically beats gradient descent?

There is an interesting history here: in 1979, Nemirovski and Yudin have shown that *every* first-order method needs in the worst case $\Omega(1/\sqrt{\varepsilon})$ steps (gradient evaluations) in order to achieve an additive error of ε on smooth functions [NY83]. Recall that we have seen an upper bound of $O(1/\varepsilon)$ for gradient descent in the previous section; in fact, this upper bound was known to Nemirovsky and Yudin already. Reformulated in the language of the previous section, there is a first-order method (gradient descent) that attains additive error $O(1/T)$ after T steps, and all first-order methods have additive error $\Omega(1/T^2)$ in the worst case.

The obvious question resulting from this was whether there actually exists a first-order method that has additive error $O(1/T^2)$ after T steps, on every smooth function. This was answered in the affirmative by Nesterov in 1983 when he proposed an algorithm that is now known as *(Nesterov's) accelerated gradient descent* [Nes83]. Nesterov's book (Sections 2.1 and 2.2) is a comprehensive source for both lower and upper bound [Nes18].

It is not easy to understand why the accelerated gradient descent algorithm is an optimal first-order method, and how Nesterov even arrived at it. A number of alternative derivations of optimal algorithms have been given by other authors, usually claiming that they provide a more natural or easier-to-grasp approach. However, each alternative approach requires some understanding of other things, and there is no well-established "simplest approach". Here, we simply throw the algorithm at the reader, without any attempt to motivate it beyond some obvious words. Then we present a short proof that the algorithm is indeed optimal.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex, differentiable, and smooth with parameter L. *Accelerated gradient descent* is the following algorithm: choose $\mathbf{z}_0 = \mathbf{y}_0 = \mathbf{x}_0$ arbitrary. For $t \geq 0$, set

$$\mathbf{y}_{t+1} := \mathbf{x}_t - \frac{1}{L}\nabla f(\mathbf{x}_t), \tag{3.11}$$

$$\mathbf{z}_{t+1} := \mathbf{z}_t - \frac{t+1}{2L}\nabla f(\mathbf{x}_t), \tag{3.12}$$

$$\mathbf{x}_{t+1} := \frac{t+1}{t+3}\mathbf{y}_{t+1} + \frac{2}{t+3}\mathbf{z}_{t+1}. \tag{3.13}$$

This means, we are performing a normal "smooth step" from $\mathbf{x}_t$ to obtain $\mathbf{y}_{t+1}$ and a more aggressive step from $\mathbf{z}_t$ to get $\mathbf{z}_{t+1}$. The next iterate $\mathbf{x}_{t+1}$ is a weighted average of $\mathbf{y}_{t+1}$ and $\mathbf{z}_{t+1}$, where we compensate for the more aggressive step by giving $\mathbf{z}_{t+1}$ a relatively low weight.

Theorem 3.9. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that f is smooth with parameter L according to (3.8). Accelerated gradient descent (3.11), (3.12), and (3.13), yields*

$$f(\mathbf{y}_T) - f(\mathbf{x}^\star) \leq \frac{2L\left\|\mathbf{z}_0 - \mathbf{x}^\star\right\|^2}{T(T+1)}, \quad T > 0.$$

Comparing this bound with the one from Theorem 3.8, we see that the error is now indeed $O(1/T^2)$ instead of $O(1/T)$; to reach error at most ε,

accelerated gradient descent therefore only needs $O(1/\sqrt{\varepsilon})$ steps instead of $O(1/\varepsilon)$.

Proof. The analysis uses a *potential function argument* [BG17]. We assign a potential $\Phi(t)$ to each time t and show that $\Phi(t+1) \leq \Phi(t)$. The potential is

$$\Phi(t) := t(t+1)\left(f(\mathbf{y}_t) - f(\mathbf{x}^\star)\right) + 2L\left\|\mathbf{z}_t - \mathbf{x}^\star\right\|^2.$$

If we can show that the potential always decreases, we get

$$\underbrace{T(T+1)\left(f(\mathbf{y}_T) - f(\mathbf{x}^\star)\right) + 2L\left\|\mathbf{z}_T - \mathbf{x}^\star\right\|^2}_{\Phi(T)} \leq \underbrace{2L\left\|\mathbf{z}_0 - \mathbf{x}^\star\right\|^2}_{\Phi(0)},$$

from which the statement immediately follows. For the argument, we need three well-known ingredients: (i) sufficient decrease (Lemma 3.7) for step (3.11) with $\gamma = 1/L$:

$$f(\mathbf{y}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2; \tag{3.14}$$

(ii) the vanilla analysis (Section 3.3) for step (3.12) with $\gamma = \frac{t+1}{2L}$, $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$:

$$\mathbf{g}_t^\top(\mathbf{z}_t - \mathbf{x}^\star) = \frac{t+1}{4L}\|\mathbf{g}_t\|^2 + \frac{L}{t+1}\left(\|\mathbf{z}_t - \mathbf{x}^\star\|^2 - \|\mathbf{z}_{t+1} - \mathbf{x}^\star\|^2\right); \tag{3.15}$$

(iii) convexity:

$$f(\mathbf{x}_t) - f(\mathbf{w}) \leq \mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{w}), \quad \mathbf{w} \in \mathbb{R}^d. \tag{3.16}$$

On top of this, we perform some simple calculations next. By definition, the potentials are

$$\begin{aligned}
\Phi(t+1) &= t(t+1)\left(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^\star)\right) + 2(t+1)\left(f(\mathbf{y}_{t+1}) - f(\mathbf{x}^\star)\right) + 2L\left\|\mathbf{z}_{t+1} - \mathbf{x}^\star\right\|^2 \\
\Phi(t) &= t(t+1)\left(f(\mathbf{y}_t\ \ ) - f(\mathbf{x}^\star)\right) + 2L\left\|\mathbf{z}_t\ \ - \mathbf{x}^\star\right\|^2
\end{aligned}$$

Now,

$$\Delta := \frac{\Phi(t+1) - \Phi(t)}{t+1}$$

can be bounded as follows.

$$\begin{aligned}
\Delta &= t\left(f(\mathbf{y}_{t+1})-f(\mathbf{y}_t)\right)+2\left(f(\mathbf{y}_{t+1})-f(\mathbf{x}^\star)\right)+\frac{2L}{t+1}\left(\|\mathbf{z}_{t+1}-\mathbf{x}^\star\|^2-\|\mathbf{z}_t-\mathbf{x}^\star\|^2\right)\\
&\overset{(3.15)}{=} t\left(f(\mathbf{y}_{t+1})-f(\mathbf{y}_t)\right)+2\left(f(\mathbf{y}_{t+1})-f(\mathbf{x}^\star)\right)+\frac{t+1}{2L}\|\mathbf{g}_t\|^2-2\mathbf{g}_t^\top(\mathbf{z}_t-\mathbf{x}^\star)\\
&\overset{(3.14)}{\leq} t\left(f(\mathbf{x}_t)-f(\mathbf{y}_t)\right)+2\left(f(\mathbf{x}_t)-f(\mathbf{x}^\star)\right)-\frac{1}{2L}\|\mathbf{g}_t\|^2-2\mathbf{g}_t^\top(\mathbf{z}_t-\mathbf{x}^\star)\\
&\leq t\left(f(\mathbf{x}_t)-f(\mathbf{y}_t)\right)+2\left(f(\mathbf{x}_t)-f(\mathbf{x}^\star)\right)-2\mathbf{g}_t^\top(\mathbf{z}_t-\mathbf{x}^\star)\\
&\overset{(3.16)}{\leq} t\mathbf{g}_t^\top(\mathbf{x}_t-\mathbf{y}_t)+2\mathbf{g}_t^\top(\mathbf{x}_t-\mathbf{x}^\star)-2\mathbf{g}_t^\top(\mathbf{z}_t-\mathbf{x}^\star)\\
&= \mathbf{g}_t^\top((t+2)\mathbf{x}_t-t\mathbf{y}_t-2\mathbf{z}_t)\\
&\overset{(3.13)}{=} \mathbf{g}_t^\top\mathbf{0}=0.
\end{aligned}$$

Hence, we indeed have $\Phi(t+1)\leq\Phi(t)$. □

3.7 Interlude

Let us get back to the supermodel $f(x)=x^2$ (that is smooth with parameter $L=2$, as we observed before). According to Theorem 3.8, gradient descent (3.1) with stepsize $\gamma=1/2$ satisfies

$$f(x_T)\leq\frac{1}{T}x_0^2. \tag{3.17}$$

Here we used that the minimizer is $x^\star=0$. Let us check how good this bound really is. For our concrete function and concrete stepsize, (3.1) reads as

$$x_{t+1}=x_t-\frac{1}{2}\nabla f(x_t)=x_t-x_t=0,$$

so we are always done after one step! But we will see in the next section that this is only because the function is particularly beautiful, and on top of that, we have picked the best possible smoothness parameter. To simulate a more realistic situation here, let us assume that we have not looked at the supermodel too closely and found it to be smooth with parameter $L=4$ only (which is a suboptimal but still valid parameter). In this case, $\gamma=1/4$ and (3.1) becomes

$$x_{t+1}=x_t-\frac{1}{4}\nabla f(x_t)=x_t-\frac{x_t}{2}=\frac{x_t}{2}.$$

So, we in fact have

$$f(x_T) = f\left(\frac{x_0}{2^T}\right) = \frac{1}{2^{2T}} x_0^2. \tag{3.18}$$

This is still vastly better than the bound of (3.17)! While (3.17) requires $T \approx x_0^2/\varepsilon$ to achieve $f(x_T) \leq \varepsilon$, (3.18) requires only

$$T \approx \frac{1}{2} \log\left(\frac{x_0^2}{\varepsilon}\right),$$

which is an exponential improvement in the number of steps.

3.8 Smooth and strongly convex functions: $\mathcal{O}(\log(1/\varepsilon))$ steps

The supermodel function $f(x) = x^2$ is not only smooth ("not too curved") but also *strongly convex* ("not too flat"). It will turn out that this is the crucial ingredient that makes gradient descent fast.

Definition 3.10. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be a convex and differentiable function, $X \subseteq \mathbf{dom}(f)$ convex and $\mu \in \mathbb{R}_+, \mu > 0$. Function f is called* strongly convex *(with parameter μ) over X if*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X. \tag{3.19}$$

If $X = \mathbf{dom}(f)$, f is simply called strongly convex.

While smoothness according to (3.8) says that for any $\mathbf{x} \in X$, the graph of f is *below* a *not-too-steep* tangential paraboloid at $(\mathbf{x}, f(\mathbf{x}))$, strong convexity means that the graph of f is *above* a *not-too-flat* tangential paraboloid at $(\mathbf{x}, f(\mathbf{x}))$. The graph of a smooth *and* strongly convex function is therefore at every point wedged between two paraboloids; see Figure 3.3.

We can also interpret (3.19) as a strengthening of convexity. In the form of (3.7), convexity reads as

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathbf{dom}(f),$$

and therefore says that every convex function satisfies (3.19) with $\mu = 0$.

In the spirit of Lemma 3.3 for smooth functions, we can characterize strong convexity via convexity of another function.

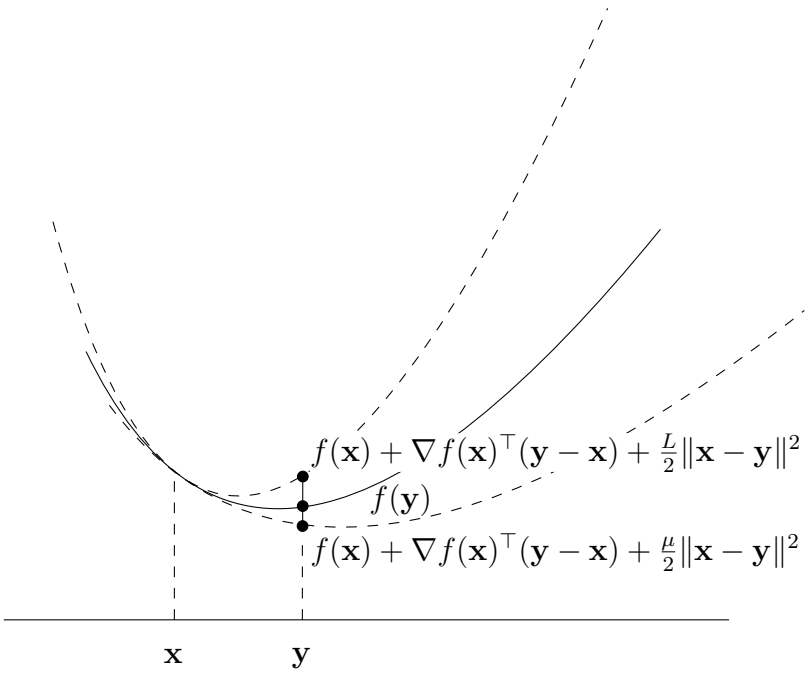


Figure 3.3: A smooth and strongly convex function

Lemma 3.11 (Exercise 28)**.** *Suppose that* $\mathbf{dom}(f)$ *is open and convex, and that* $f : \mathbf{dom}(f) \to \mathbb{R}$ *is differentiable. Let* $\mu \in \mathbb{R}_+$*. Then the following two statements are equivalent.*

(i) f *is strongly convex with parameter* μ*.*

(ii) g *defined by* $g(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2}\mathbf{x}^\top\mathbf{x}$ *is convex over* $\mathbf{dom}(g) := \mathbf{dom}(f)$*.*

Lemma 3.12 (Exercise 29)**.** *If* $f : \mathbb{R}^d \to \mathbb{R}$ *is strongly convex with parameter* $\mu > 0$*, then* f *is strictly convex and has a unique global minimum.*

The supermodel $f(x) = x^2$ is particularly beautiful since it is both smooth and strongly convex with the same parameter $L = \mu = 2$ (going through the calculations in Exercise 22 will reveal this). We can easily characterize the class of particularly beautiful functions. These are exactly the ones whose sublevel sets are ℓ_2-balls.

Lemma 3.13 (Exercise 30)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be strongly convex with parameter $\mu > 0$ and smooth with parameter μ. Prove that f is of the form*

$$f(\mathbf{x}) = \frac{\mu}{2}\|\mathbf{x} - \mathbf{b}\|^2 + c,$$

where $\mathbf{b} \in \mathbb{R}^d, c \in \mathbb{R}$.

Once we have a unique global minimum $\mathbf{x}^\star$, we can attempt to prove that $\lim_{t\to\infty} \mathbf{x}_t = \mathbf{x}^\star$ in gradient descent. We start from the vanilla analysis (3.4) and plug in the lower bound $\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) = \nabla f(\mathbf{x}_t)^\top(\mathbf{x}_t - \mathbf{x}^\star) \geq f(\mathbf{x}_t) - f(\mathbf{x}^\star) + \frac{\mu}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2$ resulting from strong convexity. We get

$$f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \frac{1}{2\gamma}\left(\gamma^2\|\nabla f(\mathbf{x}_t)\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2\right) - \frac{\mu}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2. \tag{3.20}$$

Rewriting this yields a bound on $\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2$ in terms of $\|\mathbf{x}_t - \mathbf{x}^\star\|^2$, along with some "noise" that we still need to take care of:

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \leq 2\gamma(f(\mathbf{x}^\star) - f(\mathbf{x}_t)) + \gamma^2\|\nabla f(\mathbf{x}_t)\|^2 + (1 - \mu\gamma)\|\mathbf{x}_t - \mathbf{x}^\star\|^2. \tag{3.21}$$

Theorem 3.14. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable. Suppose that f is smooth with parameter L according to (4.5) and strongly convex with parameter $\mu > 0$ according to (4.9). Exercise 33 asks you to prove that there is a unique global minimum $\mathbf{x}^\star$ of f. Choosing*

$$\gamma := \frac{1}{L},$$

gradient descent (3.1) with arbitrary $\mathbf{x}_0$ satisfies the following two properties.

(i) Squared distances to $\mathbf{x}^\star$ are geometrically decreasing:

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \leq \left(1 - \frac{\mu}{L}\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2, \quad t \geq 0.$$

(ii) The absolute error after T iterations is exponentially small in T:

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^T\|\mathbf{x}_0 - \mathbf{x}^\star\|^2, \quad T > 0.$$

Proof. For (i), we show that the noise in (3.21) disappears. By sufficient decrease (Lemma 3.7), we know that

$$f(\mathbf{x}^\star) - f(\mathbf{x}_t) \leq f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq -\frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2,$$

and hence the noise can be bounded as follows, using $\gamma = 1/L$, multiplying by 2γ and rearranging the terms, we get:

$$2\gamma\left(f(\mathbf{x}^\star) - f(\mathbf{x}_t)\right) + \gamma^2\|\nabla f(\mathbf{x}_t)\|^2 \leq 0,$$

Hence, (3.21) actually yields

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \leq (1 - \mu\gamma)\|\mathbf{x}_t - \mathbf{x}^\star\|^2 = \left(1 - \frac{\mu}{L}\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2$$

and

$$\|\mathbf{x}_T - \mathbf{x}^\star\|^2 \leq \left(1 - \frac{\mu}{L}\right)^T \|\mathbf{x}_0 - \mathbf{x}^\star\|^2.$$

The bound in (ii) follows from smoothness (3.8), using $\nabla f(\mathbf{x}^\star) = \mathbf{0}$ (Lemma 2.23):

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \nabla f(\mathbf{x}^\star)^\top(\mathbf{x}_T - \mathbf{x}^\star) + \frac{L}{2}\|\mathbf{x}_T - \mathbf{x}^\star\|^2 = \frac{L}{2}\|\mathbf{x}_T - \mathbf{x}^\star\|^2.$$

□

From this, we can derivate a rate in terms of the number of steps required (T). Using the inequality $\ln(1 + x) \leq x$, it follows that after

$$T \geq \frac{L}{\mu}\ln\left(\frac{R^2 L}{2\varepsilon}\right),$$

iterations, we reach absolute error at most ε.

3.9 Exercises

Exercise 20. *Let $\mathbf{c} \in \mathbb{R}^d$. Prove that the spectral norm of $\mathbf{c}^\top$ equals the Euclidean norm of $\mathbf{c}$, meaning that*

$$\max_{\mathbf{x}\neq\mathbf{0}} \frac{|\mathbf{c}^\top\mathbf{x}|}{\|\mathbf{x}\|} = \|\mathbf{c}\|.$$

Exercise 21. *Prove Lemma 3.3! (Alternative characterization of smoothness)*

Exercise 22. *Prove Lemma 3.4: The quadratic function $f(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$, Q symmetric, is smooth with parameter $2\|Q\|$.*

Exercise 23. *Consider the function $f(x) = |x|^{3/2}$ for $x \in \mathbb{R}$.*

(i) Prove that f is strictly convex and differentiable, with a unique global minimum $x^\star = 0$.

(ii) Prove that for every fixed stepsize γ in gradient descent (3.1) applied to f, there exists x_0 for which $f(x_1) > f(x_0)$.

(iii) Prove that f is not smooth.

(iv) Let $X \subseteq \mathbb{R}$ be a closed convex set such that $0 \in X$ and $X \neq \{0\}$. Prove that f is not smooth over X.

Exercise 24. *In order to obtain average error at most ε in Theorem 3.1, we need to choose iteration number and stepsize as*

$$T \geq \left(\frac{RB}{\varepsilon}\right)^2, \quad \gamma := \frac{R}{B\sqrt{T}}.$$

If R or B are unknown, we cannot do this.

Suppose now that we know R but not B. This means, we know a concrete number R such that $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$; we also know that there exists a number B such that $\|\nabla f(\mathbf{x})\| \leq B$ for all $\mathbf{x}$, but we don't know a concrete such number.

Develop an algorithm that—not knowing B—finds a vector $\mathbf{x}$ such that $f(\mathbf{x}) - f(\mathbf{x}^\star) < \varepsilon$, using at most

$$\mathcal{O}\left(\left(\frac{RB}{\varepsilon}\right)^2\right)$$

many gradient descent steps!

Exercise 25. *Prove Lemma 3.6! (Operations which preserve smoothness)*

Exercise 26. *In order to obtain average error at most ε in Theorem 3.8, we need to choose*

$$\gamma := \frac{1}{L}, \quad T \geq \frac{R^2 L}{2\varepsilon},$$

if $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$. If L is unknown, we cannot do this.

Now suppose that we know R but not L. This means, we know a concrete number R such that $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$; we also know that there exists a number L such that f is smooth with parameter L, but we don't know a concrete such number.

Develop an algorithm that—not knowing L—finds a vector $\mathbf{x}$ such that $f(\mathbf{x}) - f(\mathbf{x}^\star) < \varepsilon$, using at most

$$\mathcal{O}\left(\frac{R^2 L}{2\varepsilon}\right)$$

many gradient descent steps!

Exercise 27. *Let $a \in \mathbb{R}$. Prove that $f(x) = x^4$ is smooth over $X = (-a, a)$ and determine a concrete smoothness parameter L.*

Exercise 28. *Prove Lemma 3.11! (Alternative characterization of strong convexity)*

Exercise 29. *Prove Lemma 3.12! (Strongly convex functions have unique global minimum)*

Exercise 30. *Prove Lemma 3.13! (Strongly convex and smooth functions)*

Chapter 4

Projected Gradient Descent

Contents

4.1 The Algorithm

Another way to control gradients in (3.5) is to minimize f over a closed convex subset $X \subseteq \mathbb{R}^d$. For example, we may have a constrained optimization problem to begin with (for example the LASSO in Section 2.6.2), or we happen to know some region X containing a global minimum $\mathbf{x}^\star$, so that we can restrict our search to that region. In this case, gradient descent also works, but we need an additional *projection step*. After all, it can happen that some iteration of (3.1) takes us "into the wild" (out of X) where we have no business to do. *Projected* gradient descent is the following modification. We choose $\mathbf{x}_0 \in X$ arbitrary and for $t \geq 0$ define

$$\begin{aligned} \mathbf{y}_{t+1} &:= \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t), && (4.1) \\ \mathbf{x}_{t+1} &:= \Pi_X(\mathbf{y}_{t+1}) := \operatorname*{argmin}_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{y}_{t+1}\|^2. && (4.2) \end{aligned}$$

This means, after each iteration, we project the obtained iterate $\mathbf{y}_{t+1}$ back to X. This may be very easy (think of X as the unit ball in which case we just have to scale $\mathbf{y}_{t+1}$ down to length 1 if it is longer). But it may also be very difficult. In general, computing $\Pi_X(\mathbf{y}_{t+1})$ means to solve an auxiliary convex constrained minimization problem in each step! Here, we are just assuming that we can do this. The projection is well-defined: the squared distance function $d_{\mathbf{y}}(\mathbf{x}) := \|\mathbf{x} - \mathbf{y}\|^2$ is strongly convex, and hence, a unique minimum over the nonempty closed and convex set X exists by Exercise 33.

We note that finding an initial $\mathbf{x}_0 \in X$ also reduces to projection (of $\mathbf{0}$, for example) onto X.

We will frequently need the following

Fact 4.1. *Let $X \subseteq \mathbb{R}^d$ be closed and convex, $\mathbf{x} \in X, \mathbf{y} \in \mathbb{R}^d$. Then*

(i) $(\mathbf{x} - \Pi_X(\mathbf{y}))^\top (\mathbf{y} - \Pi_X(\mathbf{y})) \leq 0$.

(ii) $\|\mathbf{x} - \Pi_X(\mathbf{y})\|^2 + \|\mathbf{y} - \Pi_X(\mathbf{y})\|^2 \leq \|\mathbf{x} - \mathbf{y}\|^2$.

Part (i) says that the vectors $\mathbf{x} - \Pi_X(\mathbf{y})$ and $\mathbf{y} - \Pi_X(\mathbf{y})$ form an obtuse angle, and (ii) equivalently says that the square of the long side $\mathbf{x} - \mathbf{y}$ in the triangle formed by the three points is at least the sum of squares of the two short sides; see Figure 4.1.

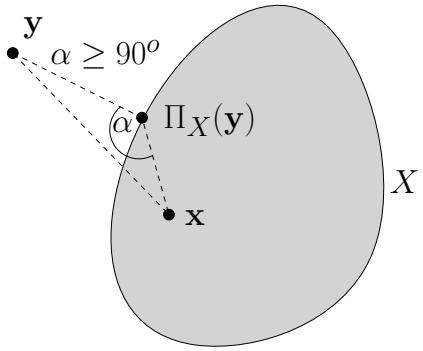


Figure 4.1: Illustration of Fact 4.1

Proof. $\Pi_X(\mathbf{y})$ is by definition a minimizer of the (differentiable) convex function $d_{\mathbf{y}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|^2$ over X, and (i) is just the equivalent optimality condition of Lemma 2.28. We need X to be closed in the first place in order to ensure that we can project onto X (see Exercise 33 applied with $d_{\mathbf{y}}(\mathbf{x})$). Indeed, for example, the number 1 has no closest point in the set $[-\infty, 0) \in \mathbb{R}$. Part (ii) follows from (i) via the (by now well-known) equation $2\mathbf{v}^\top \mathbf{w} = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - \|\mathbf{v} - \mathbf{w}\|^2$. □

Exercise 31 asks you to prove that if $\mathbf{x}_{t+1} = \mathbf{x}_t$ in projected gradient descent (i.e. we project back to the previous iterate), then $\mathbf{x}_t$ is a minimizer of f over X.

4.2 Bounded gradients: $\mathcal{O}(1/\varepsilon^2)$ steps

As in the unconstrained case, let us first assume that gradients are bounded by a constant B—this time over X. This implies that f is B-Lipschitz over X (see Theorem 2.10), but the converse may not hold.

If we minimize f over a closed and *bounded* (= compact) convex set X, we get the existence of a minimizer and a bound R for the initial distance to it for free; assuming that f is *continuously* differentiable, we also have a bound B for the gradient norms over X. This is because then $\mathbf{x} \mapsto \|\nabla f(\mathbf{x})\|$ is a continuous function that attains a maximum over X. In this case, our vanilla analysis yields a much more useful result than the one in Theorem 3.1, with the same stepsize and the same number of steps.

Theorem 4.2. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be convex and differentiable, $X \subseteq \mathbf{dom}(f)$ closed and convex, $\mathbf{x}^\star$ a minimizer of f over X; furthermore, suppose that $\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq R$, and that $\|\nabla f(\mathbf{x})\| \leq B$ for all $\mathbf{x} \in X$. Choosing the constant stepsize*

$$\gamma := \frac{R}{B\sqrt{T}},$$

projected gradient descent (4.1) with $\mathbf{x}_0 \in X$ yields

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{RB}{\sqrt{T}}.$$

Proof. The only required changes to the vanilla analysis are that in steps (3.3) and (3.4), $\mathbf{x}_{t+1}$ needs to be replaced by $\mathbf{y}_{t+1}$ as this is the real next (non-projected) gradient descent iterate after these steps; we therefore get

$$\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) = \frac{1}{2\gamma}\left(\gamma^2\|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|^2\right). \tag{4.3}$$

From Fact 4.1 (ii) (with $\mathbf{x} = \mathbf{x}^\star, \mathbf{y} = \mathbf{y}_{t+1}$), we obtain $\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \leq \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|^2$, hence we get

$$\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) \leq \frac{1}{2\gamma}\left(\gamma^2\|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2\right) \tag{4.4}$$

and return to the previous vanilla analysis for the remainder of the proof. □

4.3 Smooth convex functions: $\mathcal{O}(1/\varepsilon)$ steps

We recall from Definition 3.2 that f that is smooth over X if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X. \tag{4.5}$$

To minimize f over X, we use projected gradient descent again. The runtime turns out to be the same as in the unconstrained case. Again, we have sufficient decrease. This is not obvious from the following lemma, but you are asked to prove it in Exercise 32.

Lemma 4.3. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be differentiable and smooth with parameter L over a closed and convex set $X \subseteq \mathbf{dom}(f)$, according to (4.5). Choosing stepsize*

$$\gamma := \frac{1}{L},$$

projected gradient descent (4.1) with arbitrary $\mathbf{x}_0 \in X$ satisfies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2, \quad t \geq 0.$$

More specifically, this already holds if f is smooth with parameter L over the line segment connecting $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$.

Proof. We proceed similar to the proof of the "unconstrained" sufficient decrease Lemma 3.7, except that we now need to deal with projected gradient descent. We again start from smoothness but then use $\mathbf{y}_{t+1} = \mathbf{x}_t - \nabla f(\mathbf{x}_t)/L$, followed by the usual equation $2\mathbf{v}^\top\mathbf{w} = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - \|\mathbf{v}-\mathbf{w}\|^2$:

$$\begin{aligned}
f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\
&= f(\mathbf{x}_t) - L(\mathbf{y}_{t+1} - \mathbf{x}_t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{L}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\
&= f(\mathbf{x}_t) - \frac{L}{2}\left(\|\mathbf{y}_{t+1} - \mathbf{x}_t\|^2 + \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2\right) \\
&\quad + \frac{L}{2}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 \\
&= f(\mathbf{x}_t) - \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_t\|^2 + \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2 \\
&= f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2.
\end{aligned}$$

□

Theorem 4.4. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be convex and differentiable. Let $X \subseteq \mathbf{dom}(f)$ be a closed convex set, and assume that there is a minimizer $\mathbf{x}^\star$ of f over X; furthermore, suppose that f is smooth over X with parameter L according to (4.5). Choosing stepsize*

$$\gamma := \frac{1}{L},$$

projected gradient descent (4.1) with $\mathbf{x}_0 \in X$ satisfies

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{L}{2T}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2, \quad T > 0.$$

Proof. The plan is as in the proof of Theorem 3.8 to use the inequality

$$\frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 \leq f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) + \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2 \tag{4.6}$$

resulting from sufficient decrease (Lemma 4.3) to bound the squared gradient $\|\mathbf{g}_t\|^2 = \|\nabla f(\mathbf{x}_t)\|^2$ in the vanilla analysis. Unfortunately, (4.6) has an extra term compared to what we got in the unconstrained case. But we can compensate for this in the vanilla analysis itself. Let us go back to its "constrained" version (4.3), featuring $\mathbf{y}_{t+1}$ instead of $\mathbf{x}_{t+1}$:

$$\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) = \frac{1}{2\gamma}\left(\gamma^2\|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|^2\right).$$

Previously, we applied $\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \leq \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|^2$ (Fact 4.1(ii)) to get back on the unconstrained vanilla track. But in doing so, we dropped a term that now becomes useful. Indeed, Fact 4.1(ii) actually yields $\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 + \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2 \leq \|\mathbf{y}_{t+1} - \mathbf{x}^\star\|^2$, so that we get the following upper bound for $\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star)$:

$$\frac{1}{2\gamma}\left(\gamma^2\|\mathbf{g}_t\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2\right). \tag{4.7}$$

Using $f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq \mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star)$ from convexity, we have (with $\gamma = 1/L$) that

$$\begin{aligned}
\sum_{t=0}^{T-1}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) &\leq \sum_{t=0}^{T-1}\mathbf{g}_t^\top(\mathbf{x}_t - \mathbf{x}^\star) \qquad (4.8)\\
&\leq \frac{1}{2L}\sum_{t=0}^{T-1}\|\mathbf{g}_t\|^2 + \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2 - \frac{L}{2}\sum_{t=0}^{T-1}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2.
\end{aligned}$$

To bound the sum of the squared gradients, we use (4.6):

$$\begin{aligned}
\frac{1}{2L}\sum_{t=0}^{T-1}\|\mathbf{g}_t\|^2 &\leq \sum_{t=0}^{T-1}\left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) + \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2\right)\\
&= f(\mathbf{x}_0) - f(\mathbf{x}_T) + \frac{L}{2}\sum_{t=0}^{T-1}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2.
\end{aligned}$$

Plugging this into (4.8), the extra terms cancel, and we arrive—as in the unconstrained case—at

$$\sum_{t=1}^{T} (f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \leq \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2.$$

The statement follows as in the proof of Theorem 3.8 from the fact that due to sufficient decrease (Exercise 32), the last iterate is the best one. □

4.4 Smooth and strongly convex functions: $\mathcal{O}(\log(1/\varepsilon))$ steps

Assuming that f is smooth *and* strongly convex over a set X, we can also prove fast convergence of projected gradient descent. This does not require any new ideas, we have seen all the ingredients before.

We recall from Definition 3.10 that f is strongly convex with parameter $\mu > 0$ over X if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X. \tag{4.9}$$

Theorem 4.5. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be convex and differentiable. Let $X \subseteq \mathbf{dom}(f)$ be a nonempty closed and convex set and suppose that f is smooth over X with parameter L according to (4.5) and strongly convex over X with parameter $\mu > 0$ according to (4.9). Exercise 33 asks you to prove that there is a unique minimizer $\mathbf{x}^\star$ of f over X. Choosing*

$$\gamma := \frac{1}{L},$$

projected gradient descent (4.1) with arbitrary $\mathbf{x}_0$ satisfies the following two properties.

(i) Squared distances to $\mathbf{x}^\star$ are geometrically decreasing:

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 \leq \left(1 - \frac{\mu}{L}\right)\|\mathbf{x}_t - \mathbf{x}^\star\|^2, \quad t \geq 0.$$

(ii) The absolute error after T iterations is exponentially small in T:

$$\begin{aligned} f(\mathbf{x}_T) - f(\mathbf{x}^\star) &\leq \|\nabla f(\mathbf{x}^\star)\| \left(1 - \frac{\mu}{L}\right)^{T/2} \|\mathbf{x}_0 - \mathbf{x}^\star\| \\ &+ \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^{T} \|\mathbf{x}_0 - \mathbf{x}^\star\|^2, \quad T > 0. \end{aligned}$$

We note that this is *almost* the same result as in Theorem 3.14 for the unconstrained case; in fact, the result in part (i) is identical, but in part (ii), we get an additional term. This is due to the fact that in the constrained case, we cannot argue that $\nabla f(\mathbf{x}^\star) = \mathbf{0}$. In fact, this additional term is the dominating one, once the error becomes small. It has the effect that the required number of steps to reach error at most ε will roughly double, in comparison to the bound of Theorem 3.14.

Proof. In the strongly convex case, the "constrained" vanilla bound (4.7)

$$\frac{1}{2\gamma}\left(\gamma^2\|\nabla f(\mathbf{x}_t)\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2\right)$$

on $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$ can be strengthened to

$$\frac{1}{2\gamma}\left(\gamma^2\|\nabla f(\mathbf{x}_t)\|^2 + \|\mathbf{x}_t - \mathbf{x}^\star\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2\right) - \frac{\mu}{2}\|\mathbf{x}_t - \mathbf{x}^\star\|^2 \tag{4.10}$$

Now we proceed as in the proof of Theorem 3.14 and rewrite the latter bound into a bound on $\|\mathbf{x}_{t+1} - \mathbf{x}^\star\|^2$ that is

$$2\gamma(f(\mathbf{x}^\star) - f(\mathbf{x}_t)) + \gamma^2\|\nabla f(\mathbf{x}_t)\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2 + (1 - \mu\gamma)\|\mathbf{x}_t - \mathbf{x}^\star\|^2,$$

so we have geometric decrease in squared distance to $\mathbf{x}^\star$, up to some noise. Again, we show that by sufficient decrease, the noise in this bound disappears. From Lemma 4.3, we know that

$$f(\mathbf{x}^\star) - f(\mathbf{x}_t) \leq f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq -\frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 + \frac{L}{2}\|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2,$$

and using this, the noise can be bounded. Multiplying the previous inequality by $2/L$, and rearranging the terms we get:

$$\frac{2}{L}\left(f(\mathbf{x}^\star) - f(\mathbf{x}_t)\right) + \frac{1}{L^2}\|\nabla f(\mathbf{x}_t)\|^2 - \|\mathbf{y}_{t+1} - \mathbf{x}_{t+1}\|^2 \leq 0.$$

With $\gamma = 1/L$, this exactly shows that the noise is nonpositive. This yields (i). The bound in (ii) follows from smoothness (3.8):

$$\begin{aligned}
f(\mathbf{x}_T) - f(\mathbf{x}^\star) &\leq \nabla f(\mathbf{x}^\star)^\top(\mathbf{x}_T - \mathbf{x}^\star) + \frac{L}{2}\|\mathbf{x}^\star - \mathbf{x}_T\|^2 \\
&\leq \|\nabla f(\mathbf{x}^\star)\|\,\|\mathbf{x}_T - \mathbf{x}^\star\| + \frac{L}{2}\|\mathbf{x}^\star - \mathbf{x}_T\|^2 \text{ (Cauchy-Schwarz)} \\
&\leq \|\nabla f(\mathbf{x}^\star)\|\left(1 - \frac{\mu}{L}\right)^{T/2}\|\mathbf{x}_0 - \mathbf{x}^\star\| + \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^{T}\|\mathbf{x}_0 - \mathbf{x}^\star\|^2.
\end{aligned}$$

□

4.5 Projecting onto ℓ_1-balls

Problems that are ℓ_1-regularized appear among the most commonly used models in machine learning and signal processing, and we have already discussed the Lasso as an important example of that class. We will now address how to perform projected gradient as an efficient optimization for ℓ_1-constrained problems. Let

$$X = B_1(R) := \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 = \sum_{i=1}^{d} |x_i| \leq R \right\}$$

be the ℓ_1-ball of radius $R > 0$ around $\mathbf{0}$, i.e., the set of all points with 1-norm at most R. Our goal is to compute $\Pi_X(\mathbf{v})$ for a given vector $\mathbf{v}$, i.e. the projection of $\mathbf{v}$ onto X; see Figure 4.2.

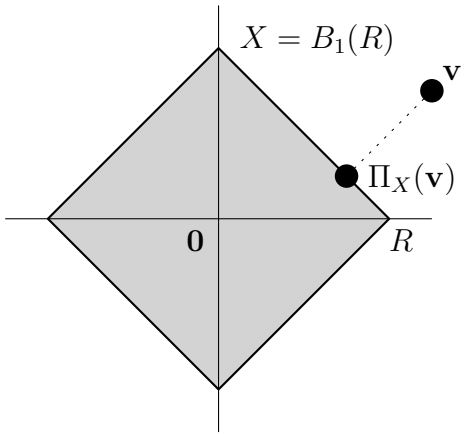


Figure 4.2: Projecting onto an ℓ_1-ball

At first sight, this may look like a rather complicated task. Geometrically, X is a *cross polytope* (square for $d = 2$, octahedron for $d = 3$), and as such it has 2^d many facets. But we can start with some basic simplifying observations.

Fact 4.6. *We may assume without loss of generality that (i) $R = 1$, (ii) $v_i \geq 0$ for all i, and (iii) $\sum_{i=1}^{d} v_i > 1$.*

Proof. If we project $\mathbf{v}/R$ onto $B_1(1)$, we obtain $\Pi_X(\mathbf{v})/R$ (just scale Figure 4.2), so we can restrict to the case $R = 1$. For (ii), we observe that

simultaneously flipping the signs of a fixed subset of coordinates in both $\mathbf{v}$ and $\mathbf{x} \in X$ yields vectors $\mathbf{v}'$ and $\mathbf{x}' \in X$ such that $\|\mathbf{x} - \mathbf{v}\| = \|\mathbf{x}' - \mathbf{v}'\|$; thus, $\mathbf{x}$ minimizes the distance to $\mathbf{v}$ if and only if $\mathbf{x}'$ minimizes the distance to $\mathbf{v}'$. Hence, it suffices to compute $\Pi_X(\mathbf{v})$ for vectors with nonnegative entries. If $\sum_{i=1}^d v_i \leq 1$, we have $\Pi_X(\mathbf{v}) = \mathbf{v}$ and are done, so the interesting case is (iii). □

Fact 4.7. *Under the assumptions of Fact 4.6,* $\mathbf{x} = \Pi_X(\mathbf{v})$ *satisfies* $x_i \geq 0$ *for all* i *and* $\sum_{i=1}^d x_i = 1$.

Proof. If $x_i < 0$ for some i, then $(-x_i - v_i)^2 \leq (x_i - v_i)^2$ (since $v_i \geq 0$), so flipping the i-th sign in $\mathbf{x}$ would yield another vector in X at least as close to $\mathbf{v}$ as $\mathbf{x}$, but such a vector cannot exist by strict convexity of the squared distance. And if $\sum_{i=1}^d x_i < 1$, then $\mathbf{x}' = \mathbf{x} + \lambda(\mathbf{v} - \mathbf{x}) \in X$ for some small positive λ, with $\|\mathbf{x}' - \mathbf{v}\| = (1 - \lambda)\|\mathbf{x} - \mathbf{v}\|$, again contradicting the optimality of $\mathbf{x}$. □

Corollary 4.8. *Under the assumptions of Fact 4.6,*

$$\Pi_X(\mathbf{v}) = \operatorname*{argmin}_{\mathbf{x} \in \Delta_d} \|\mathbf{x} - \mathbf{v}\|^2,$$

where

$$\Delta_d := \left\{ \mathbf{x} \in \mathbb{R}^d : \sum_{i=1}^d x_i = 1, x_i \geq 0 \; \forall i \right\}$$

is the standard simplex.

This means, we have reduced the projection onto an ℓ_1-ball to the projection onto the standard simplex; see Figure 4.3.

To address the latter task, we make another assumption that can be established by suitably permuting the entries of $\mathbf{v}$ (which just permutes the entries of its projection onto Δ_d in the same way).

Fact 4.9. *We may assume without loss of generality that* $v_1 \geq v_2 \geq \cdots \geq v_d$.

Lemma 4.10. *Let* $\mathbf{x}^\star := \operatorname{argmin}_{\mathbf{x} \in \Delta_d} \|\mathbf{x} - \mathbf{v}\|^2$. *Under the assumption of Fact 4.9, there exists (a unique)* $p \in \{1, \ldots, d\}$ *such that*

$$\begin{aligned} x_i^\star &> 0, \quad i \leq p, \\ x_i^\star &= 0, \quad i > p. \end{aligned}$$

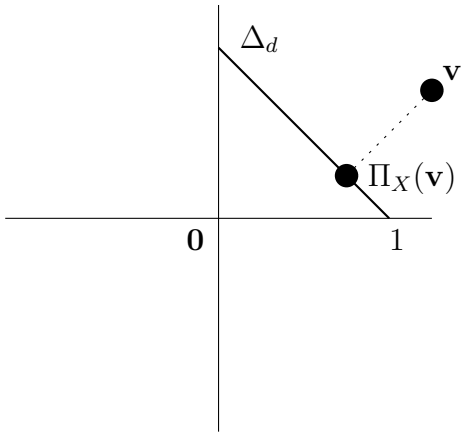


Figure 4.3: Projecting onto the standard simplex

Proof. We are using the optimality criterion of Lemma 2.28:

$$\nabla d_{\mathbf{v}}(\mathbf{x}^\star)^\top(\mathbf{x}-\mathbf{x}^\star) = 2(\mathbf{x}^\star-\mathbf{v})^\top(\mathbf{x}-\mathbf{x}^\star) \geq 0, \quad \mathbf{x}\in\Delta_d, \tag{4.11}$$

where $d_{\mathbf{v}}(\mathbf{z}) := \|\mathbf{z}-\mathbf{v}\|^2$ is the squared distance to $\mathbf{v}$.

Because $\sum_{i=1}^d x_i^\star = 1$, there is at least one positive entry in $\mathbf{x}^\star$. It remains to show that we cannot have $x_i^\star = 0$ and $x_{i+1}^\star > 0$. Indeed, in this situation, we could decrease $x_{i+1}^\star$ by some small positive ε and simultaneously increase $x_i^\star$ to ε to obtain a vector $\mathbf{x}\in\Delta_d$ such that

$$(\mathbf{x}^\star-\mathbf{v})^\top(\mathbf{x}-\mathbf{x}^\star) = (0-v_i)\varepsilon - (x_{i+1}^\star - v_{i+1})\varepsilon = \varepsilon(\underbrace{v_{i+1}-v_i}_{\leq 0} - \underbrace{x_{i+1}^\star}_{>0}) < 0,$$

contradicting the optimality (4.11). □

But we can say even more about $\mathbf{x}^\star$.

Lemma 4.11. *Under the assumption of Fact 4.9, and with p as in Lemma 4.10,*

$$x_i^\star = v_i - \Theta_p, \quad i\leq p,$$

where

$$\Theta_p = \frac{1}{p}\Big(\sum_{i=1}^p v_i - 1\Big).$$

Proof. Again, we argue by contradiction. If not all $x_i^\star - v_i, i \le p$ have the same value $-\Theta_p$, then we have $x_i^\star - v_i < x_j^\star - v_j$ for some $i, j \le p$. As before, we can then decrease $x_j^\star > 0$ by some small positive ε and simultaneously increase $x_i^\star$ by ε to obtain $\mathbf{x} \in \Delta_d$ such that

$$(\mathbf{x}^\star - \mathbf{v})^\top (\mathbf{x} - \mathbf{x}^\star) = (x_i^\star - v_i)\varepsilon - (x_j^\star - v_j)\varepsilon = \varepsilon \underbrace{((x_i^\star - v_i) - (x_j^\star - v_j))}_{<0} < 0,$$

again contradicting (4.11). The expression for Θ_p is then obtained from

$$1 = \sum_{i=1}^{p} x_i^\star = \sum_{i=1}^{p} (v_i - \Theta_p) = \sum_{i=1}^{p} v_i - p\Theta_p.$$

□

Let us summarize the situation: we now have d candidates for $\mathbf{x}^\star$, namely the vectors

$$\mathbf{x}^\star(p) := (v_1 - \Theta_p, \dots, v_p - \Theta_p, 0, \dots, 0), \quad p \in \{1, \dots, d\}, \tag{4.12}$$

and we just need to find the right one. In order for candidate $\mathbf{x}^\star(p)$ to comply with Lemma 4.10, we must have

$$v_p - \Theta_p > 0, \tag{4.13}$$

and this actually ensures $\mathbf{x}^\star(p)_i > 0$ for all $i \le p$ by the assumption of Fact 4.9 and therefore $\mathbf{x}^\star(p) \in \Delta_d$. But there could still be several values of p satisfying (4.13). Among them, we simply pick the one for which $\mathbf{x}^\star(p)$ minimizes the distance to $\mathbf{v}$. It is not hard to see that this can be done in time $\mathcal{O}(d \log d)$, by first sorting v and then carefully updating the values Θ_p and $\|\mathbf{x}^\star(p) - \mathbf{v}\|^2$ as we vary p to check all candidates.

But actually, there is an even simpler criterion that saves us from comparing distances.

Lemma 4.12. *Under the assumption of Fact 4.9, with $\mathbf{x}^\star(p)$ as in (4.12), and with*

$$p^\star := \max\Big\{p \in \{1, \dots, d\} : v_p - \frac{1}{p}\Big(\sum_{i=1}^{p} v_i - 1\Big) > 0\Big\},$$

it holds that

$$\operatorname*{argmin}_{\mathbf{x} \in \Delta_d} \|\mathbf{x} - \mathbf{v}\|^2 = \mathbf{x}^\star(p^\star).$$

The proof is Exercise 34. Together with our previous reductions, we obtain the following result.

Theorem 4.13. *Let $\mathbf{v} \in \mathbb{R}^d$, $R \in \mathbb{R}_+$, $X = B_1(R)$ the ℓ_1-ball around $\mathbf{0}$ of radius R. The projection*

$$\Pi_X(\mathbf{v}) = \operatorname*{argmin}_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{v}\|^2$$

of $\mathbf{v}$ onto $B_1(R)$ can be computed in time $\mathcal{O}(d \log d)$.

This can be improved to time $\mathcal{O}(d)$, based on the observation that a given p can be compared to the value $p^\star$ in Lemma 4.12 in linear time, without the need to presort $\mathbf{v}$ [DSSSC08].

4.6 Exercises

Exercise 31. *Consider the projected gradient descent algorithm as in (4.1) and (4.2), with a convex differentiable function f. Suppose that for some iteration t, $\mathbf{x}_{t+1} = \mathbf{x}_t$. Prove that in this case, $\mathbf{x}_t$ is a minimizer of f over the closed and convex set X!*

Exercise 32. *Prove that in Theorem 4.4 (i),*

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t).$$

Exercise 33. *Let $X \subseteq \mathbb{R}^d$ be a nonempty closed and convex set, and let f be strongly convex over X. Prove that f has a unique minimizer $\mathbf{x}^\star$ over X! In particular, for $X = \mathbb{R}^d$, we obtain the existence of a unique global minimum.*

Exercise 34. *Prove Lemma 4.12!*

Hint: *It is useful to prove that with $\mathbf{x}^\star(p)$ as in (4.12) and satisfying (4.13),*

$$\mathbf{x}^\star(p) = \operatorname{argmin}\{\|\mathbf{x} - \mathbf{v}\| : \sum_{i=1}^{d} x_i = 1, x_{p+1} = \cdots = x_d = 0\}.$$

Chapter 5

Coordinate Descent

Contents

5.1 Overview

In large-scale learning, an issue with the gradient descent algorithms discussed in Chapter 3 is that in every iteration, we need to compute the full gradient $\nabla f(\mathbf{x}_t)$ in order to obtain the next iterate $\mathbf{x}_{t+1}$. If the number of variables d is large, this can be very costly. The idea of coordinate descent is to update only *one* coordinate of $\mathbf{x}_t$ at a time, and to do this, we only need to compute *one* coordinate of $\nabla f(\mathbf{x}_t)$ (one partial derivative). We expect this to be by a factor of d faster than computation of the full gradient and update of the full iterate.

But we also expect to pay a price for this in terms of a higher number of iterations. In this chapter, we will analyze a number of coordinate descent variants on smooth and strongly convex functions. It turns out that in the worst case, the number of iterations will increase by a factor of d, so nothing is gained (but also nothing is lost).

But under suitable additional assumptions about the function f, coordinate descent variants can actually lead to provable speedups. In practice, coordinate descent algorithms are popular due to their simplicity and often good performance.

Much of this chapter's material is from Karimi at al. [KNS16] and Nutini et al. [NSL+15]. As a warm-up, we return to gradient descent.

5.2 Alternative analysis of gradient descent

We have analyzed gradient descent on smooth and strongly convex functions before (Section 3.8) and in particular proved that the sequence of iterates converges to the unique global minimum $\mathbf{x}^\star$. Here we go a different route. We will only prove that the sequence of function values converges to the optimal function value. To do so, we do not need strong convexity but only the *Polyak-Łojasiewicz inequality*, a consequence of strong convexity that we derive next. This alternative simple anlysis of gradient descent will also pave the way for our later analysis of coordinate descent.

5.2.1 The Polyak-Łojasiewicz inequality

Definition 5.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function with a global minimum $\mathbf{x}^\star$. We say that f satisfies the* Polyak-Łojasiewicz inequality *(PL in-*

equality) if the following holds for some $\mu > 0$:

$$\frac{1}{2}\|\nabla f(\mathbf{x})\|^2 \geq \mu(f(\mathbf{x}) - f(\mathbf{x}^\star)), \quad \forall\, \mathbf{x} \in \mathbb{R}^d. \tag{5.1}$$

The inequality was proposed by Polyak in 1963, and also by Łojasiewicz in the same year; see Karimi et al. and the references therein [KNS16]. It says that the squared gradient norm at every point $\mathbf{x}$ is at least proportional to the error in objective function value at $\mathbf{x}$. It also directly implies that every critical point (a point where $\nabla f(\mathbf{x}) = \mathbf{0}$) is a minimizer of f.

The interesting result for us is that strong convexity over $\mathbb{R}^d$ implies the PL inequality.

Lemma 5.2 (Strong Convexity $\Rightarrow$ PL inequality)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and strongly convex with parameter $\mu > 0$ (in particular, a global minimum $\mathbf{x}^\star$ exists by Lemma 3.12). Then f satisfies the PL inequality for the same μ.*

Proof. Using strong convexity, we get

$$\begin{aligned} f(\mathbf{x}^\star) &\geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{x}^\star - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{x}^\star - \mathbf{x}\|^2 \\ &\geq f(\mathbf{x}) + \min_{\mathbf{y}} \left(\nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|^2\right) \\ &= f(\mathbf{x}) - \frac{1}{2\mu}\|\nabla f(\mathbf{x})\|^2. \end{aligned}$$

The latter equation results from solving a convex minimization problem in $\mathbf{y}$ by finding a critical point (Lemma 2.22). The PL inequality follows. □

The PL inequality is a strictly weaker condition than strong convexity. For example, consider $f(x_1, x_2) = x_1^2$ which is not strongly convex: every point $(0, x_2)$ is a global minimum. But f still satisfies the PL inequality, since it behaves like the strongly convex function $x \to x^2$ in (5.1).

There are even nonconvex functions satisfying the PL inequality (Exercise 35).

5.2.2 Analysis

We can now easily analyze gradient descent on smooth functions that in addition satisfy the PL inequality. By Exercise 35, this result also covers some nonconvex optimization problems.

Theorem 5.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum $\mathbf{x}^\star$. Suppose that f is smooth with parameter L according to (4.5) and satisfies the PL inequality (5.1) with parameter $\mu > 0$. Choosing stepsize*

$$\gamma = \frac{1}{L},$$

gradient descent (3.1) with arbitrary $\mathbf{x}_0$ satisfies

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \left(1 - \frac{\mu}{L}\right)^T (f(\mathbf{x}_0) - f(\mathbf{x}^\star)), \quad T > 0.$$

Proof. For all t, we have

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2 && \text{(sufficient decrease, Lemma 3.7)} \\ &\leq f(\mathbf{x}_t) - \frac{\mu}{L}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) && \text{(PL inequality (5.1))}. \end{aligned}$$

If we subtract $f(\mathbf{x}^\star)$ on both sides, we get

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}^\star) \leq \left(1 - \frac{\mu}{L}\right)(f(\mathbf{x}_t) - f(\mathbf{x}^\star)),$$

and the statement follows. □

5.3 Coordinate-wise smoothness

To analyze coordinate descent, we work with *coordinate-wise smoothness*.

Definition 5.4. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable, and $\mathcal{L} = (L_1, L_2, \ldots, L_d) \in \mathbb{R}^d_+$. Function f is called* coordinate-wise smooth *(with parameter $\mathcal{L}$) if for every coordinate $i = 1, 2, \ldots, d$,*

$$f(\mathbf{x} + \lambda \mathbf{e}_i) \leq f(\mathbf{x}) + \lambda \nabla_i f(\mathbf{x}) + \frac{L_i}{2}\lambda^2 \quad \forall \mathbf{x} \in \mathbb{R}^d, \lambda \in \mathbb{R}, . \tag{5.2}$$

If $L_i = L$ for all i, f is said to be coordinate-wise smooth with parameter L.

Let's compare this to our standard definition of smoothness in Definition 3.2. It is easy to see that if f is smooth with parameter L, then f is coordinate-wise smooth with parameter L. Indeed, (5.2) then coincides

with the regular smoothness inequality (3.8), when applied to vectors $\mathbf{y}$ of the form $\mathbf{y} = \mathbf{x} + \lambda \mathbf{e}_i$.

But we may be able to say more. For example, $f(x_1, x_2) = x_1^2 + 10x_2^2$ is smooth with parameter $L = 20$ (due to the $10x_2^2$ term, no smaller value will do), but f is coordinate-wise smooth with parameter $\mathcal{L} = (2, 20)$. So coordinate-wise smoothness allows us to obtain a more fine-grained picture of f than smoothness.

There are even cases where the best possible smoothness parameter is L, but we can choose coordinate-wise smoothness parameters L_i (significantly) smaller than L for all i. Consider $f(x_1, x_2) = x_1^2 + x_2^2 + Mx_1x_2$ for a constant $M > 0$. For $\mathbf{y} = (y, y)$ and $\mathbf{x} = \mathbf{0}$, smoothness requires that $(M + 2)y^2 = f(\mathbf{y}) \leq \frac{L}{2}\|\mathbf{y}\|^2 = Ly^2$, so we need smoothness parameter $L \geq (M + 2)$.

On the other hand, f is coordinate-wise smooth with $\mathcal{L} = (2, 2)$: fixing one cordinate, we obtain a univariate function of the form $x^2 + ax + b$. This is smooth with parameter 2 (use Lemma 3.6 (i) along with the fact that affine functions are smooth with parameter 0).

5.4 Coordinate descent algorithms

Coordinate descent methods generate a sequence $\{\mathbf{x}_t\}_{t\geq 0}$ of iterates. In iteration t, they do the following:

$$\begin{aligned} &\text{choose an active coordinate } i \in [d] \\ &\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_i \nabla_i f(\mathbf{x}_t)\mathbf{e}_i. \end{aligned} \tag{5.3}$$

Here, $\mathbf{e}_i$ denotes the i-th unit basis vector in $\mathbb{R}^d$, and λ_i is a suitable stepsize for the selected coordinate i. We will focus on the gradient-based choice of the stepsize as

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_i \nabla_i f(\mathbf{x}_t)\, \mathbf{e}_i\,, \tag{5.4}$$

Here, $\nabla_i f(\mathbf{x})$ denotes the i-th entry of the gradient $\nabla f(\mathbf{x})$.

In the coordinate-wise smooth case, we obtain a variant of sufficient decrease for coordinate descent.

Lemma 5.5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and coordinate-wise smooth with parameter $\mathcal{L} = (L_1, L_2, \ldots, L_d)$ according to (5.2). With active coordinate i in*

iteration t and stepsize

$$\gamma_i = \frac{1}{L_i},$$

coordinate descent (5.4) satisfies

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L_i}|\nabla_i f(\mathbf{x}_t)|^2.$$

Proof. We apply the coordinate-wise smoothness condition (5.2) with $\lambda = -\nabla_i f(\mathbf{x}_t)/L_i$, for which we have $\mathbf{x}_{t+1} = \mathbf{x}_t + \lambda \mathbf{e}_i$. Hence

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) + \lambda \nabla_i f(\mathbf{x}_t) + \frac{L_i}{2}\lambda^2 \\ &= f(\mathbf{x}_t) - \frac{1}{L_i}|\nabla_i f(\mathbf{x}_t)|^2 + \frac{1}{2L_i}|\nabla_i f(\mathbf{x}_t)|^2 \\ &= f(\mathbf{x}_t) - \frac{1}{2L_i}|\nabla_i f(\mathbf{x}_t)|^2. \end{aligned}$$

□

In the next two sections, we consider randomized variants of coordinate descent that pick the coordinate to consider in a given step at random (from some distribution). Using elementary techniques, we will be able to bound the *expected* number of iterations. It requires more elaborate techniques to prove tail estimates of the form that with high probability, a certain number of steps will not be exceeded [Nes12].

5.4.1 Randomized coordinate descent

In *randomized coordinate descent*, the active coordinate in step t is chosen uniformly at random from the set $[d]$:

$$\begin{aligned} &\text{sample } i \in [d] \text{ uniformly at random} \\ &\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_i \nabla_i f(\mathbf{x}_t)\mathbf{e}_i. \end{aligned} \tag{5.5}$$

Nesterov shows that randomized coordinate descent is at least as fast as gradient descent on smooth functions, if we assume that it is d times cheaper to update one coordinate than the full iterate [Nes12].

If we additionally assume the PL inequality, we can obtain fast convergence as follows.

Theorem 5.6. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum $\mathbf{x}^\star$. Suppose that f is coordinate-wise smooth with parameter L according to Definition 5.4 and satisfies the PL inequality (5.1) with parameter $\mu > 0$. Choosing stepsize*

$$\gamma_i = \frac{1}{L},$$

randomized coordinate descent (5.5) with arbitrary $\mathbf{x}_0$ satisfies

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^\star)] \leq \left(1 - \frac{\mu}{dL}\right)^T (f(\mathbf{x}_0) - f(\mathbf{x}^\star)), \quad T > 0.$$

Comparing this to the result for gradient descent in Theorem 5.3, the number of iterations to reach optimization error at most ε is by a factor of d higher. To see this, note that (for μ/L small)

$$\left(1 - \frac{\mu}{L}\right) \approx \left(1 - \frac{\mu}{dL}\right)^d.$$

This means, while each iteration of coordinate descent is by a factor of d cheaper, the number of iterations is by a factor of d higher, so we have a zero-sum game here. But in the next section, we will refine the analysis and show that there are cases where coordinate descent will actually be faster. But first, let's prove Theorem 5.6.

Proof. By definition, f is coordinate-wise smooth with $(L, L, \ldots, L)$, so sufficient decrease according to Lemma 5.5 yields

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}|\nabla_i f(\mathbf{x}_t)|^2.$$

By taking the expectation of both sides with respect to the choice of i, we have

$$\begin{aligned}
\mathbb{E}\left[f(\mathbf{x}_{t+1})|\mathbf{x}_t\right] &\leq f(\mathbf{x}_t) - \frac{1}{2L}\sum_{i=1}^d \frac{1}{d}|\nabla_i f(\mathbf{x}_t)|^2 \\
&= f(\mathbf{x}_t) - \frac{1}{2dL}\|\nabla f(\mathbf{x}_t)\|^2 \\
&\leq f(\mathbf{x}_t) - \frac{\mu}{dL}(f(\mathbf{x}_t) - f(\mathbf{x}^\star)) \quad \text{(PL inequality (5.1))}.
\end{aligned}$$

In the second line, we conveniently used the fact that the squared Euclidean norm is additive. Subtracting $f(\mathbf{x}^\star)$ from both sides, we therefore obtain

$$\mathbb{E}[f(\mathbf{x}_{t+1}) - f(\mathbf{x}^\star)|\mathbf{x}_t] \leq \left(1 - \frac{\mu}{dL}\right)(f(\mathbf{x}_t) - f(\mathbf{x}^\star)).$$

Taking expectations (over $\mathbf{x}_t$), we obtain

$$\mathbb{E}[f(\mathbf{x}_{t+1}) - f(\mathbf{x}^\star)] \leq \left(1 - \frac{\mu}{dL}\right)\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}^\star)].$$

The statement follows. □

In the proof, we have used conditional expectations: $\mathbb{E}\left[f(\mathbf{x}_{t+1})|\mathbf{x}_t\right]$ is a random variable whose expectation is $\mathbb{E}\left[f(\mathbf{x}_{t+1})\right]$.

5.4.2 Importance Sampling

Uniformly random selection of the active coordinate is not the best choice when the coordinate-wise smoothness parameters L_i differ. In this case, it makes sense to sample proportional to the L_i's as suggested by Nesterov [Nes12]. This is coordinate descent with *importance sampling*:

$$\begin{aligned} &\text{sample } i \in [d] \text{ with probability } \frac{L_i}{\sum_{j=1}^d L_j} \\ &\mathbf{x}_{t+1} := \mathbf{x}_t - \frac{1}{L_i}\nabla_i f(\mathbf{x}_t)\mathbf{e}_i. \end{aligned} \tag{5.6}$$

Here is the result.

Theorem 5.7. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum $\mathbf{x}^\star$. Suppose that f is coordinate-wise smooth with parameter $\mathcal{L} = (L_1, L_2, \ldots, L_d)$ according to (5.2) and satisfies the PL inequality (5.1) with parameter $\mu > 0$. Let*

$$\bar{L} = \frac{1}{d}\sum_{i=1}^d L_i$$

be the average of all coordinate-wise smoothness constants. Then coordinate descent with importance sampling (5.6) and arbitrary $\mathbf{x}_0$ satisfies

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^\star)] \leq \left(1 - \frac{\mu}{d\bar{L}}\right)^T (f(\mathbf{x}_0) - f(\mathbf{x}^\star)), \quad T > 0.$$

The proof of the theorem is Exercise 36. We note that $\bar{L}$ can be much smaller than the value $L = \max_{i=1}^{d} L_i$ that appears in Theorem 5.6, so coordinate descent with importance sampling is potentially much faster than randomized coordinate descent. In the worst-case (all L_i are equal), both algorithms are the same.

5.4.3 Steepest coordinate descent

In contrast to random coordinate descent, *steepest coordinate descent* (or greedy coordinate descent) chooses the active coordinate according to

$$\begin{aligned} &\text{choose } i = \operatorname*{argmax}_{i \in [d]} |\nabla_i f(\mathbf{x}_t)| \\ &\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_i \nabla_i f(\mathbf{x}_t) \mathbf{e}_i. \end{aligned} \tag{5.7}$$

This is a deterministic algorithm and also called the *Gauss-Southwell* rule.

It is easy to show that the same convergence rate that we have obtained for random coordinate descent in Theorem 5.6 also holds for steepest coordinate descent. To see this, the only ingredient we need is the fact that

$$\max_i |\nabla_i f(\mathbf{x})|^2 \geq \frac{1}{d} \sum_{i=1}^{d} |\nabla_i f(\mathbf{x})|^2 ,$$

and since we now have a deterministic algorithm, there is no need to take expectations in the proof.

Corollary 5.8. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum $\mathbf{x}^\star$. Suppose that f is coordinate-wise smooth with parameter L according to Definition 5.4 and satisfies the PL inequality (5.1) with parameter $\mu > 0$. Choosing stepsize*

$$\gamma_i = \frac{1}{L},$$

steepest coordinate descent (5.7) with arbitrary $\mathbf{x}_0$ satisfies

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \left(1 - \frac{\mu}{dL}\right)^T (f(\mathbf{x}_0) - f(\mathbf{x}^\star)), \quad T > 0.$$

This result is a bit disappointing: individual iterations seem to be as costly as in gradient descent, but the number of iterations is by factor of d larger. This comparison with Theorem 5.3 is not fully fair, though, since in contrast to gradient descent, steepest coordinate descent requires only coordinate-wise smoothness, and as we have seen in Section 5.3, this can be better than global smoothness. But steepest coordinate descent also cannot compete with randomized gradient descent (same number of iterations, but higher cost per iteration). However, we show next that the algorithm allows for a speedup in certain cases; also, it may be possible to efficiently maintain the maximum absolute gradient value throughout the iterations, so that evaluation of the full gradient can be avoided.

Strong convexity with respect to ℓ_1-norm. It was shown by Nutini et al. [NSL+15] that a better convergence result can be obtained for strongly convex functions, when strong convexity is measured with respect to ℓ_1-norm instead of the standard Euclidean norm, i.e.

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu_1}{2} \|\mathbf{y} - \mathbf{x}\|_1^2, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{5.8}$$

Due to $\|\mathbf{y} - \mathbf{x}\|_1 \geq \|\mathbf{y} - \mathbf{x}\|$, f is then also strongly convex with $\mu = \mu_1$ in the usual sense. On the other hand, if f is μ-strongly convex in the usual sense, then f satisfies (5.8) with $\mu_1 = \mu/d$, due to $\|\mathbf{y} - \mathbf{x}\| \geq \|\mathbf{y} - \mathbf{x}\|_1 / \sqrt{d}$.

Hence, μ_1 may be up to factor of d smaller than μ, and if this happens, (5.8) will not lead to a speedup of the algorithm. But isn't μ_1 necessarily smaller than μ by a factor of d ? After all, there are always $\mathbf{x}, \mathbf{y}$ such that $\|\mathbf{y} - \mathbf{x}\| = \|\mathbf{y} - \mathbf{x}\|_1 / \sqrt{d}$. But if for those worst-case $\mathbf{x}, \mathbf{y}$, the inequality of strong convexity holds with $\mu' > \mu$, we can achieve $\mu_1 > \mu/d$. As an example for this scenario, Nutini et al. [NSL+15, Appendix C of arXiv version] compute the best parameters μ, μ_1 of strong convexity for a convex function of the form $f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^d L_i x_i$ and show that μ_1 can be significiantly larger than μ/d.

The proof of convergence under (5.8) is similar to the one of Theorem 5.6, after proving the following lemma: if f is strongly convex with respect to ℓ_1-norm, it satisfies the PL inequality with respect to ℓ_∞-norm. The proof is Exercise 38 and follows the same strategy as the earlier Lemma 5.2 for the Euclidean norm. While this requires only elementary calculations, it does not reveal the deeper reason why ℓ_1-norm in strong convexity leads

to ℓ_∞-norm in the PL inequality. This has to do with convex conjugates, but we will not go into it here.

Lemma 5.9 (Exercise 38)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and strongly convex with parameter $\mu_1 > 0$ w.r.t. ℓ_1-norm as in (5.8). (In particular, f is μ_1-strongly convex w.r.t. Euclidean norm, so a global minimum $\mathbf{x}^\star$ exists by Lemma 3.12.) Then f satisfies the PL inequality w.r.t. ℓ_∞-norm with the same μ_1:*

$$\tfrac{1}{2}\|\nabla f(\mathbf{x})\|_\infty^2 \geq \mu_1(f(\mathbf{x}) - f(\mathbf{x}^\star)), \quad \forall \mathbf{x} \in \mathbb{R}^d. \tag{5.9}$$

Theorem 5.10. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum $\mathbf{x}^\star$. Suppose that f is coordinate-wise smooth with parameter L according to Definition 5.4 and satisfies the PL inequality (5.9) with parameter $\mu_1 > 0$. Choosing stepsize*

$$\gamma_i = \frac{1}{L},$$

steepest coordinate descent (5.7) with arbitrary $\mathbf{x}_0$ satisfies

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \left(1 - \frac{\mu_1}{L}\right)^T (f(\mathbf{x}_0) - f(\mathbf{x}^\star)), \quad T > 0.$$

Proof. By definition, f is coordinate-wise smooth with $(L, L, \ldots, L)$, so sufficient decrease according to Lemma 5.5 yields

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}|\nabla_i f(\mathbf{x}_t)|^2 = f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|_\infty^2,$$

by definition of steepest gradient descent. Using the PL inequality (5.9), we further get

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{\mu_1}{L}(f(\mathbf{x}_t) - f(\mathbf{x}^\star).$$

Now we proceed as in the alternative analysis of gradient descent: Subtracting $f(\mathbf{x}^\star)$ from both sides, we obtain

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}^\star) \leq \left(1 - \frac{\mu_1}{L}\right)(f(\mathbf{x}_t) - f(\mathbf{x}^\star)),$$

and the statement follows. □

5.4.4 Greedy coordinate descent

This is a variant that does not even require f to be differentiable. In each iteration, we make the step that maximizes the progress in the chosen coordinate. This requires to perform a *line search* by solving a 1-dimensional optimization problem:

$$\begin{aligned} &\text{choose } i \in [d] \\ &\mathbf{x}_{t+1} := \operatorname*{argmin}_{\lambda \in \mathbb{R}} f(\mathbf{x}_t + \lambda \mathbf{e}_i) \end{aligned} \tag{5.10}$$

There are cases where the line search can exactly be done analytically, or approximately by some other means. In the differentiable case, we can take any of the previously studied coordinate descent variants and replace some of its steps by greedy steps if it turns out that we can perform line search along the selected coodinate. This will not compromise the convergence analysis, as stepwise progress can only be better.

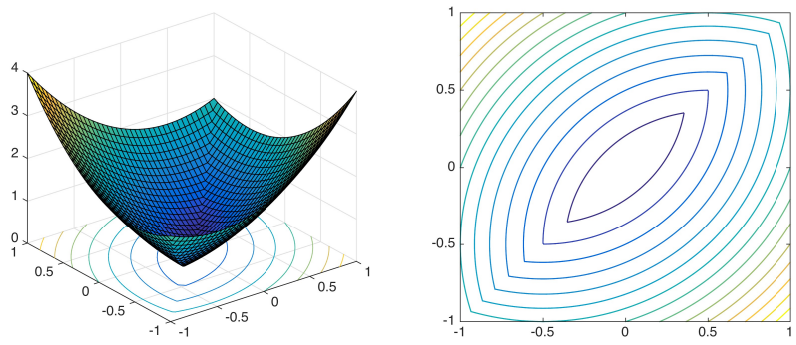


Figure 5.1: The non-differentiable function $f(\mathbf{x}) := \|\mathbf{x}\|^2 + |x_1 - x_2|$. The global minimum is $(0, 0)$, but greedy coordinate descent cannot escape any point $(x, x), |x| \leq 1/2$. *Figure by Alp Yurtsever & Volkan Cevher, EPFL*

Some care is in order when applying the greedy variant in the nondifferentiable case for which the previous variants don't work. The algorithm can get stuck in non-optimal points, as for example in the objective function of Figure 5.1. But not all hope is lost. There are relevant cases where this scenario does not happen, as we show next.

Theorem 5.11. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be of the form*

$$f(\mathbf{x}) := g(\mathbf{x}) + h(\mathbf{x}) \quad \text{with } h(\mathbf{x}) = \sum_i h_i(x_i), \quad \mathbf{x} \in \mathbb{R}^d, \tag{5.11}$$

with g convex and differentiable, and the h_i convex.

Let $\mathbf{x} \in \mathbb{R}^d$ be a point such that greedy coordinate descent cannot make progress in any coordinate. Then $\mathbf{x}$ is a global minimum of f.

A function h as in the theorem is called *separable*. Figure 5.2 illustrates the theorem.

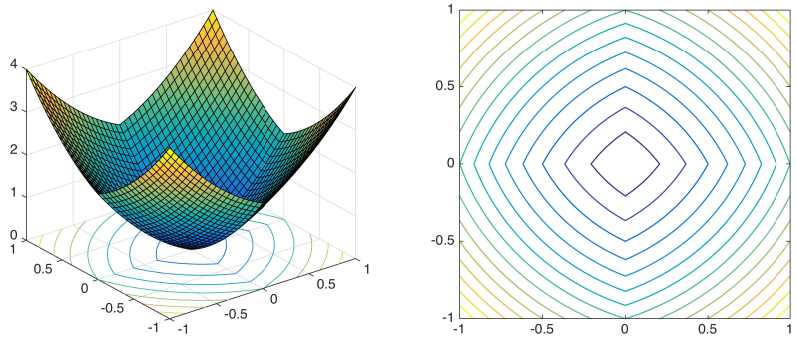


Figure 5.2: The function $f(\mathbf{x}) := \|\mathbf{x}\|^2 + \|\mathbf{x}\|_1$. Greedy coordinate descent cannot get stuck. *Figure by Alp Yurtsever & Volkan Cevher, EPFL*

Proof. We follow Ryan Tibshirani's lecture.[1]. Let $\mathbf{y} \in \mathbb{R}^d$. Using the first-order characaterization of convexity for g and the definition of h, we obtain

$$\begin{aligned} f(\mathbf{y}) &= g(\mathbf{y}) + h(\mathbf{y}) \\ &\geq g(\mathbf{x}) + \nabla g(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + h(\mathbf{x}) + \sum_{i=1}^{d} (h_i(y_i) - h_i(x_i)) \\ &= f(\mathbf{x}) + \sum_{i=1}^{d} (\nabla_i g(\mathbf{x})(y_i - x_i) + h_i(y_i) - h_i(x_i)) \geq f(\mathbf{x}), \end{aligned}$$

using that $\nabla_i g(\mathbf{x})(y_i - x_i) + h_i(y_i) - h_i(x_i) \geq 0$ for all i (Exercise 39). □

[1] https://www.stat.cmu.edu/~ryantibs/convexopt-S15/lectures/22-coord-desc.pdf

One very important class of applications here are objective functions of the form

$$f(\mathbf{x}) + \lambda\|\mathbf{x}\|_1,$$

where f is convex and smooth, and $h(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ is a (separable) ℓ_1-regularization term. The LASSO (Section 2.6) in its regularized form gives rise to a concrete such case:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|^2 + \lambda\|\mathbf{x}\|_1 \,. \tag{5.12}$$

Whether greedy coordinate descent actually converges on functions as in Theorem 5.11 is a different question; this was answered in the affirmative by Tseng under mild regularity conditions on g, and under using the cyclic order of coordinates throughout the iterations [Tse01].

5.5 Summary

Coordinate descent methods are used widely in machine learning applications. Variants of coordinate methods form the state of the art for the class of *generalized linear models*, including linear classifiers and regression models, as long as separable convex regularizers are used (e.g. ℓ_1-norm or squared ℓ_2-norm).

The following table summarizes the converegence bounds of coordinate descent algorithms on coordinate-wise smooth and strongly convex functions (we only use the PL inequality, a consequence of strong convexity). The Bound column contains the factor by which the error is guaranteed to decrease in every step.

Algorithm	PL norm	Smoothness	Bound	Result
Randomized	ℓ_2	L	$1 - \frac{\mu}{dL}$	Thm. 5.6
Importance sampling	ℓ_2	$(L_1, L_2, \ldots, L_d)$	$1 - \frac{\mu}{d\bar{L}}$	Thm. 5.7
Steepest	ℓ_2	L	$1 - \frac{\mu}{dL}$	Cor. 5.8
Steeper (than Steepest)	ℓ_1	L	$1 - \frac{\mu_1}{L}$	Thm. 5.10

In the worst case, all algorithms have a Bound of $1 - \frac{\mu}{dL}$ and therefore need d times more iterations than gradient descent. This can fully be compensated if iterations are d times cheaper.

In the best case, Steeper (than Steepest) matches the performance of gradient descent in terms of iteration count. The algorithm is therefore an attractive choice for problems where we can obtain (or maintain) the steepest coordinate of the gradient efficiently. This includes several practical case, for example when the gradients are sparse, e.g. because the original data is sparse.

Importance sampling is attractive when most coordinate-wise smoothness parameters L_i are much smaller than the maximum. In the best case, it can be d times faster than gradient descent. On the downside, applying the method requires to know all the L_i. In the other methods, an upper bound on all L_i is sufficient in order to run the algorithm.

5.6 Exercises

Exercise 35. *Provide an example of a nonconvex function that satisfies the PL inequality 5.1!*

Exercise 36 (Importance Sampling)**.** *Prove Theorem 5.7! Can you come up with an example from machine learning where $\bar{L} \ll L = \max_{i=1}^{d} L_i$?*

Exercise 37. *Derive the solution to exact coordinate minimization for the Lasso problem* (5.12)*, for the i-th coordinate. Write A_{-i} for the $n \times (d-1)$ matrix obtained by removing the i-th column from A, and same for the vector $\mathbf{x}_{-i}$ with one entry removed accordingly.*

Exercise 38. *Prove Lemma 5.9, proceeding as in the proof of Lemma 5.2!*

Exercise 39. *Let f be as in Theorem 5.11 and $\mathbf{x} \in \mathbb{R}^d$ such that $f(\mathbf{x} + \lambda \mathbf{e}_i) \geq f(\mathbf{x})$ for all λ and all i. Prove that $\nabla_i g(\mathbf{x})(y_i - x_i) + h_i(y_i) - h_i(x_i) \geq 0$ for all $\mathbf{y} \in \mathbb{R}^d$ and all $i \in [d]$.*

Chapter 6

Nonconvex functions

Contents

So far, all convergence results that we have given for variants of gradient descent have been for convex functions. And there is a good reason for this: on nonconvex functions, gradient descent can in general not be expected to come close (in distance or function value) to the global minimum $\mathbf{x}^\star$, even if there is one.

As an example, consider the nonconvex function from Figure 2.4 (left). Figure 6.1 shows what happens if we start gradient descent somewhere "to the right", with a not too large stepsize so that we do not overshoot. For any sufficiently large T, the iterate $\mathbf{x}_T$ will be close to the local minimum $\mathbf{y}^\star$, but not to the global minimum $\mathbf{x}^\star$.

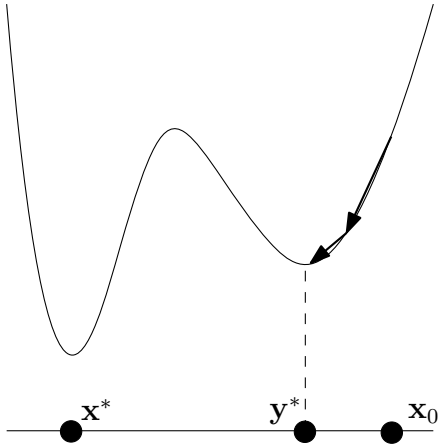


Figure 6.1: Gradient descent may get stuck in a local minimum $\mathbf{y}^\star \neq \mathbf{x}^\star$

Even if the global minimum is the unique local minimum, gradient descent is not guaranteed to get there, as it may also get stuck in a saddle point, or even fail to reach anything at all; see Figure 6.2.

In practice, variants of gradient descent are often observed to perform well even on nonconvex functions, but theoretical explanations for this are mostly missing.

In this chapter, we show that under favorable conditions, we can still say something useful about the behavior of gradient descent, even on nonconvex functions.

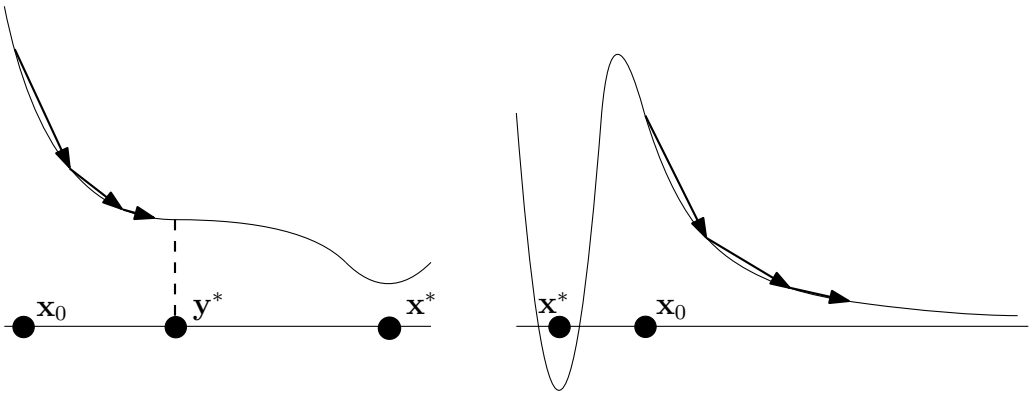


Figure 6.2: Gradient descent may get stuck in a flat region (saddle point) $\mathbf{y}^\star$ (left), or reach neither a local minimum nor a saddle point (right).

6.1 Smooth functions

A particularly low hanging fruit is the analysis of gradient descent on smooth (but not necessarily convex) functions. We recall from Definition 3.2 that a differentiable function $f : \mathbf{dom}(f) \to \mathbb{R}$ is smooth with parameter $L \in \mathbb{R}_+$ over a convex set $X \subseteq \mathbf{dom}(f)$ if

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

This means that at every point $\mathbf{x} \in X$, the graph of f is below a not-too-steep tangential paraboloid, and this may happen even if the function is not convex; see Figure 6.3.

There is a class of arbitrarily smooth nonconvex functions, namely the differentiable *concave* functions. A function f is called concave if $-f$ is convex. Hence, for all $\mathbf{x}$, the graph of a differentiable concave function is *below* the tangent hyperplane at $\mathbf{x}$, hence f is smooth with parameter $L = 0$; see Figure 6.4.

However, from our optimization point of view, concave functions are boring, since they have no global minimum (at least in the unconstrained setting that we are treating here). Gradient descent will then simply "run off to infinity".

We will therefore consider smooth functions that have a global minimum $\mathbf{x}^\star$. Are there even such functions that are not convex? Actually,

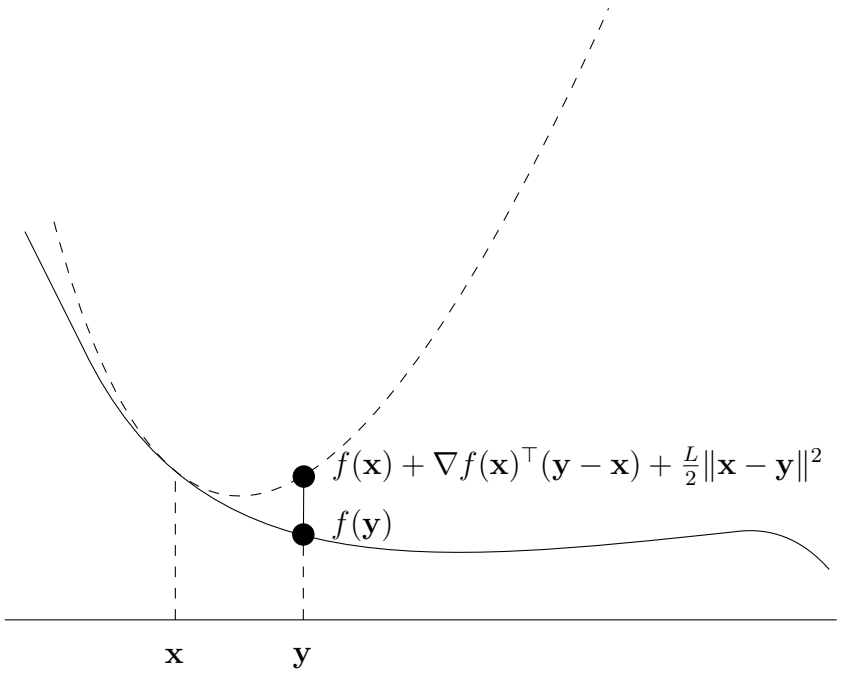


Figure 6.3: A smooth and nonconvex function

many. As we show next, any twice differentiable function with bounded Hessians over some convex set X is smooth over X. A concrete example of a smooth function that is not convex but has a global minimum (actually, many), is $f(x) = \sin(x)$.

Lemma 6.1. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be twice differentiable, with $X \subseteq \mathbf{dom}(f)$ a convex set, and $\|\nabla^2 f(\mathbf{x})\| \leq L$ for all $\mathbf{x} \in X$, where $\|\cdot\|$ is again spectral norm. Then f is smooth with parameter L over X.*

Proof. By Theorem 2.10 (applied to the gradient function ∇f), bounded Hessians imply Lipschitz continuity of the gradient,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|, \quad \mathbf{x}, \mathbf{y} \in X. \tag{6.1}$$

We show that this in turn implies smoothness. This is in fact the easy direction of Lemma 3.5 (in the twice differentiable case).

For any fixed $\mathbf{x}, \mathbf{y} \in X$, we use the (by now) very familar function $h : \mathbf{dom}(h) \to \mathbb{R}^d$ over a suitable open domain $I \supset [0, 1]$, given by

$$h(t) = f\big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\big), \quad t \in I,$$

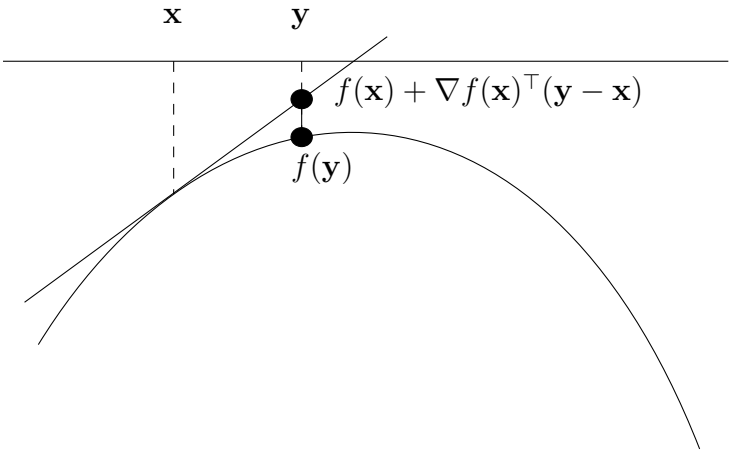


Figure 6.4: A concave function and the first-order characterization of concavity: $f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x})$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$

for which we have shown in (2.1) that

$$h'(t) = \nabla f\big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\big)^\top(\mathbf{y} - \mathbf{x}), \quad t \in I.$$

As f is twice differentiable, ∇f and hence also h' are actually continuous, so we can apply the fundamental theorem of calculus (in the second line of the lengthy but easy derivation below). We compute

$$\begin{aligned}
& f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \\
= \; & h(1) - h(0) - \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \\
= \; & \int_0^1 h'(t)dt - \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \\
= \; & \int_0^1 \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))^\top(\mathbf{y} - \mathbf{x})dt - \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) \\
= \; & \int_0^1 \big(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))^\top(\mathbf{y} - \mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x})\big)dt \\
= \; & \int_0^1 \big(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x})\big)^\top(\mathbf{y} - \mathbf{x})dt.
\end{aligned}$$

So far, we had only equalities, now we start estimating:

$$\begin{aligned}
& f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \\
= & \int_0^1 \big(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x})\big)^\top (\mathbf{y} - \mathbf{x}) dt \\
\leq & \int_0^1 \big|\big(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x})\big)^\top (\mathbf{y} - \mathbf{x})\big| dt \\
\leq & \int_0^1 \big\|\big(\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x})\big)\big\| \, \|(\mathbf{y} - \mathbf{x})\| \, dt \quad \text{(Cauchy-Schwarz)} \\
\leq & \int_0^1 L \, \|t(\mathbf{y} - \mathbf{x})\| \, \|(\mathbf{y} - \mathbf{x})\| \, dt \quad \text{(Lipschitz continuous gradients)} \\
= & \int_0^1 Lt \, \|\mathbf{x} - \mathbf{y}\|^2 \, dt \\
= & \frac{L}{2} \, \|\mathbf{x} - \mathbf{y}\|^2 .
\end{aligned}$$

This is smoothness over X according to Definition 3.2. □

For twice differentiable *convex* functions, the converse is also (almost) true. If f is smooth over an *open* convex subset $X \subseteq \mathbf{dom}(f)$, it has bounded Hessians over X (Exercise 40). Convexity is needed here since e.g. concave functions are smooth with parameter $L = 0$ but generally have unbounded Hessians. It is also not hard to understand why openness is necessary in general. Indeed, for a point $\mathbf{x}$ on the boundary of X, the smoothness condition does not give us any information about nearby points not in X. As a consequence, even at points with large Hessians, f might look smooth inside X. As a simple example, consider $f(x_1, x_2) = x_1^2 + Mx_2^2$ with $M \in \mathbb{R}_+$ large. The function f is smooth with $L = 2$ over $X = \{(x_1, x_2) : x_2 = 0\}$: indeed, over this set, f looks just like the supermodel. But for all $\mathbf{x}$, we have $\|\nabla^2 f(\mathbf{x})\| = 2M$.

Now we get back to gradient descent on smooth functions with a global minimum. The punchline is so unspectacular that there is no harm in spoiling it already now: What we can prove is that $\|\nabla f(\mathbf{x}_t)\|^2$ converges to 0 at the same rate as $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$ converges to 0 in the convex case. Naturally, $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$ itself is not guaranteed to converge in the nonconvex case, for example if $\mathbf{x}_t$ converges to a local minimum that is not global, as in Figure 6.1.

It is tempting to interpret convergence of $\|\nabla f(\mathbf{x}_t)\|^2$ to 0 as convergence to a *critical point* of f (a point where the gradient vanishes). But this interpretation is not fully accurate in general, as Figure 6.2 (right) shows: The algorithm may enter a region where f asymptotically approaches some value, without reaching it (think of the rightmost piece of the function in the figure as $f(x) = e^{-x}$). In this case, the gradient converges to 0, but the iterates are nowhere near a critical point.

Theorem 6.2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum $\mathbf{x}^\star$; furthermore, suppose that f is smooth with parameter L according to Definition 3.2. Choosing stepsize*

$$\gamma := \frac{1}{L},$$

gradient descent (3.1) yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{2L}{T}\big(f(\mathbf{x}_0) - f(\mathbf{x}^\star)\big), \quad T > 0.$$

In particular, $\|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{2L}{T}\big(f(\mathbf{x}_0) - f(\mathbf{x}^\star)\big)$ for some $t \in \{0, \dots, T-1\}$. And also, $\lim_{t\to\infty} \|\nabla f(\mathbf{x}_t)\|^2 = 0$ (Exercise 41).

Proof. We recall that sufficient decrease (Lemma 3.7) does not require convexity, and this gives

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0.$$

Rewriting this into a bound on the gradient yields

$$\|\nabla f(\mathbf{x}_t)\|^2 \leq 2L\big(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\big).$$

Hence, we get a telescoping sum

$$\sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}_t)\|^2 \leq 2L\big(f(\mathbf{x}_0) - f(\mathbf{x}_T)\big) \leq 2L\big(f(\mathbf{x}_0) - f(\mathbf{x}^\star)\big).$$

The statement follows. □

In the smooth setting, gradient descent has another interesting property: with stepsize $1/L$, it cannot overshoot. By this, we mean that it cannot pass a critical point (in particular, not the global minimum) when moving from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. Equivalently, with a smaller stepsize, no critical point can be reached. With stepsize $1/L$, it is possible to reach a critical point, as we have demonstrated for the supermodel function $f(x) = x^2$ in Section 3.7.

Lemma 6.3 (Exercise 42)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable; let $\mathbf{x} \in \mathbb{R}^d$ such that $\nabla f(\mathbf{x}) \neq \mathbf{0}$, i.e. $\mathbf{x}$ is not a critical point. Suppose that f is smooth with parameter L over the line segment connecting $\mathbf{x}$ and $\mathbf{x}' = \mathbf{x} - \gamma \nabla f(\mathbf{x})$, where $\gamma = 1/L' < 1/L$. Then $\mathbf{x}'$ is also not a critical point.*

Figure 6.5 illustrates the situation.

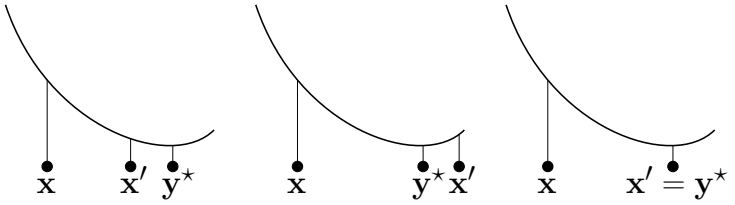


Figure 6.5: Gradient descent on smooth functions: When moving from $\mathbf{x}$ to $\mathbf{x}' = \mathbf{x} - \gamma \nabla f(\mathbf{x})$ with $\gamma < 1/L$, $\mathbf{x}'$ will not be a critical point (left); equivalently, with $\gamma = 1/L$, we cannot overshoot, i.e. pass a critical point (middle); with $\gamma = 1/L$, we may exactly reach a critical point (right).

6.2 Trajectory analysis

Even if the "landscape" (graph) of a nonconvex function has local minima, saddle points, and flat parts, it is sometimes possible to prove that gradient descent avoids these bad spots and still converges to a global minimum. For this, one needs a good starting point and some theoretical understanding of what happens when we start there—this is trajectory analysis.

In 2018, results along these lines have appeared that prove convergence of gradient descent to a global minimum in training deep *linear* linear networks, under suitable conditions. In this section, we will study a vastly

simplified setting that allows us to show the main ideas (and limitations) behind one particular trajectory analysis [ACGH18].

In our simplified setting, we will look at the task of minimizing a concrete and very simple nonconvex function. This function turns out be smooth *along the trajectories* that we analyze, and this is one important ingredient. However, smoothness alone does not suffice to prove convergence to the global minimum, let alone fast convergence: As we have seen in the last section, we can in general only guarantee that the gradient norms converge to 0, and at a rather slow rate. To get beyond this, we will need to exploit additional properties of the function under consideration.

6.2.1 Deep linear neural networks

Let us go back to the problem of learning linear models as discussed in Section 2.6.2, using the example of Master's admission. We had n inputs $\mathbf{x}_1, \dots, \mathbf{x}_n$, where each input $\mathbf{x}_i \in \mathbb{R}^d$ consisted of d input variables; and we had n outputs $y_1, \dots, y_n \in \mathbb{R}$. Then we made the hypothesis that (after centering), output values depend (approximately) linearly on the input,

$$y_i \approx \mathbf{w}^\top x_i,$$

for a weight vector $\mathbf{w} = (w_1, \dots, w_d) \in \mathbb{R}^d$ to be learned.

Now we consider the more general case where there is not just one output $y_i \in \mathbb{R}$ as response to the i-th input, but m outputs $\mathbf{y}_i \in \mathbb{R}^m$. In this case, the linear hypothesis becomes

$$\mathbf{y}_i \approx W\mathbf{x}_i,$$

for a weight matrix $W \in \mathbb{R}^{m \times d}$ to be learned. The matrix that best fits this hypothesis on the given observations is the least-squares matrix

$$W^\star = \operatorname*{argmin}_{W \in \mathbb{R}^{m \times d}} \sum_{i=1}^{n} \|W\mathbf{x}_i - \mathbf{y}_i\|^2 .$$

If we let $X \in \mathbb{R}^{d \times n}$ be the matrix whose columns are the $\mathbf{x}_i$ and $Y \in \mathbb{R}^{m \times n}$ the matrix whose columns are the $\mathbf{y}_i$, we can equivalently write this as

$$W^\star = \operatorname*{argmin}_{W \in \mathbb{R}^{m \times d}} \|WX - Y\|_F^2, \tag{6.2}$$

where $\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$ is the *Frobenius norm* of a matrix A.

Finding W^* (the global minimum of a convex quadratic function) is a simple task that boils down to solving a system of linear equations; see also Section 2.4.2. A fancy way of saying this is that we are training a linear neural network with one layer, see Figure 6.6 (left).

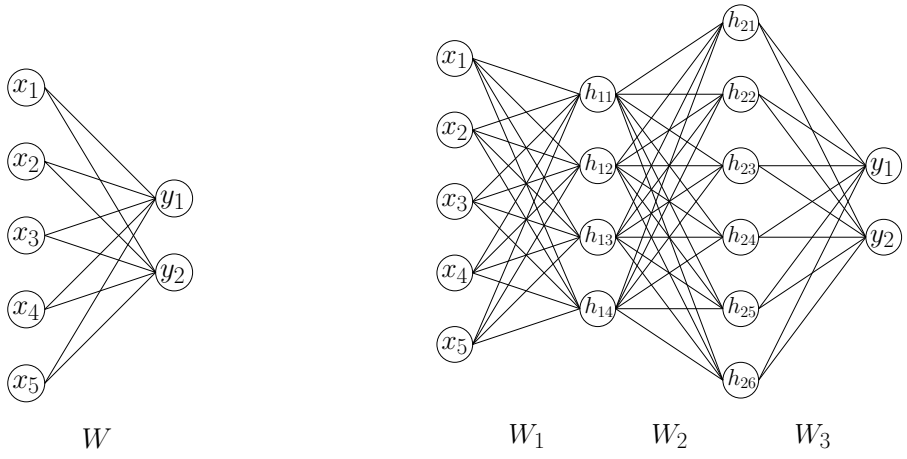


Figure 6.6: Left: A linear neural network over d input variables $\mathbf{x} = (x_1, \ldots, x_d)$ and m output variables $\mathbf{y} = (y_1, \ldots, y_m)$. The edge connecting input variable x_j with output variable y_i has a weight w_{ij} (to be learned), and all weights together form a weight matrix $W \in \mathbb{R}^{m \times d}$. Given the weights, the network computes the linear transformation $\mathbf{y} = W\mathbf{x}$ between inputs and outputs. Right: a deep linear neural network of depth 3 with weight matrices W_1, W_2, W_3. Given the weights, the network computes the linear transformation $\mathbf{y} = W_3 W_2 W_1 \mathbf{x}$.

But what if we have ℓ layers (Figure 6.6 (right)? Training such a network corresponds to minimizing

$$\|W_\ell W_{\ell-1} \cdots W_1 X - Y\|_F^2,$$

over ℓ weight matrices $W_1, \ldots, W_\ell$ to be learned. In case of linear neural networks, there is no benefit in adding layers, as any linear transformation $\mathbf{x} \mapsto W_\ell W_{\ell-1} \cdots W_1 X$ can of course be represented as $\mathbf{x} \mapsto WX$ with $W := W_{\ell-1} \cdots W_1$. But from a theoretical point of view, a deep linear neural network gives us a simple playground in which we can try to understand why training deep neural networks with gradient descent works,

despite the fact that the objective function is no longer convex. The hope is that such an understanding can ultimately lead to an analyis of gradient descent (or other suitable methods) for "real" (meaning non-linear) deep neural networks.

In the next section, we will discuss the case where all matrices are 1×1, so they are just numbers. This is arguably a toy example in our already simple playground. Still, it gives rise to a nontrivial nonconvex function, and the analysis of gradient descent on it will require similar ingredients as the one on general deep linear neural networks [ACGH18].

6.2.2 A simple nonconvex function

The function (that we consider fixed throughout the section) is $f : \mathbb{R}^d \to \mathbb{R}$ defined by

$$f(\mathbf{x}) := \frac{1}{2}\left(\prod_{k=1}^{d} x_k - 1\right)^2, \tag{6.3}$$

As d is fixed, we will abbreviate $\prod_{k=1}^{d} x_k$ by $\prod_k x_k$ throughout. Minimizing this function corresponds to training a deep linear neural network with d layers, one neuron per layer, with just one training input $x = 1$ and a corresponding output $y = 1$. Figure 6.7 visualizes the function f for $d = 2$.

First of all, the function f does have global minima, as it is nonnegative, and value 0 can be achieved (in many ways). Hence, we immediately know how to minimize this (for example, set $x_k = 1$ for all k). The question is whether gradient descent also knows, and if so, how we prove this.

Let us start by computing the gradient. We have

$$\nabla f(\mathbf{x}) = \left(\prod_k x_k - 1\right)\left(\prod_{k\neq 1} x_k\ ,\ \ldots\ ,\ \prod_{k\neq d} x_k\right)^\top. \tag{6.4}$$

What are the critical points, the ones where $\nabla f(\mathbf{x})$ vanishes? This happens when $\prod_k x_k = 1$ in which case we have a global minimum (level 0 in Figure 6.7). But there are other critical points. Whenever at least *two* of the x_k are zero, the gradient also vanishes, and the value of f is $1/2$ at such a point (point $\mathbf{0}$ in Figure 6.7). This already shows that the function cannot be convex, as for convex functions, every critical point is a global minimum (Lemma 2.22). It is easy to see that every non-optimal critical point must have two or more zeros.

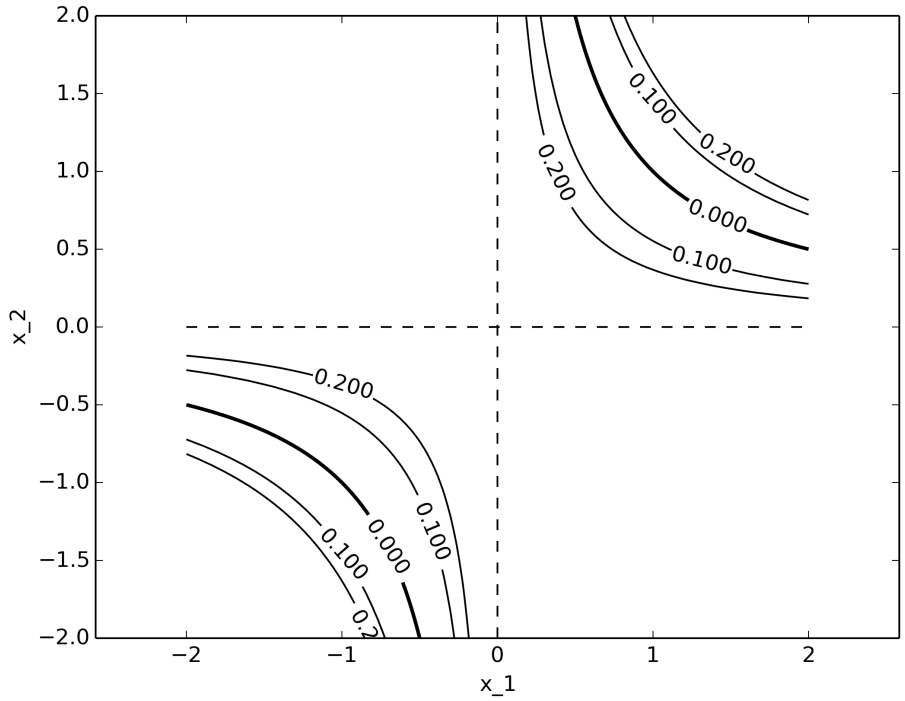


Figure 6.7: Levels sets of $f(x_1, x_2) = \frac{1}{2}(x_1 x_2 - 1)^2$

In fact, all critical points except the global minima are saddle points. This is because at any such point $\mathbf{x}$, we can slightly perturb the (two or more) zero entries in such a way that the product of all entries becomes either positive or negative, so that the function value either decreases or increases.

Figure 6.8 visualizes (scaled) negative gradients of f for $d = 2$; these are the directions in which gradient descent would move from the tails of the respective arrows. The figure already indicates that it is difficult to avoid convergence to a global minimum, but it is possible (see Exercise 44).

We now want to show that for any dimension d, and from *anywhere* in $X = \{\mathbf{x} : \mathbf{x} > \mathbf{0}, \prod_k \mathbf{x}_k \leq 1\}$, gradient descent will converge to a global minimum. Unfortunately, our function f is not smooth over X. For the analysis, we will therefore show that f is smooth along the trajectory of

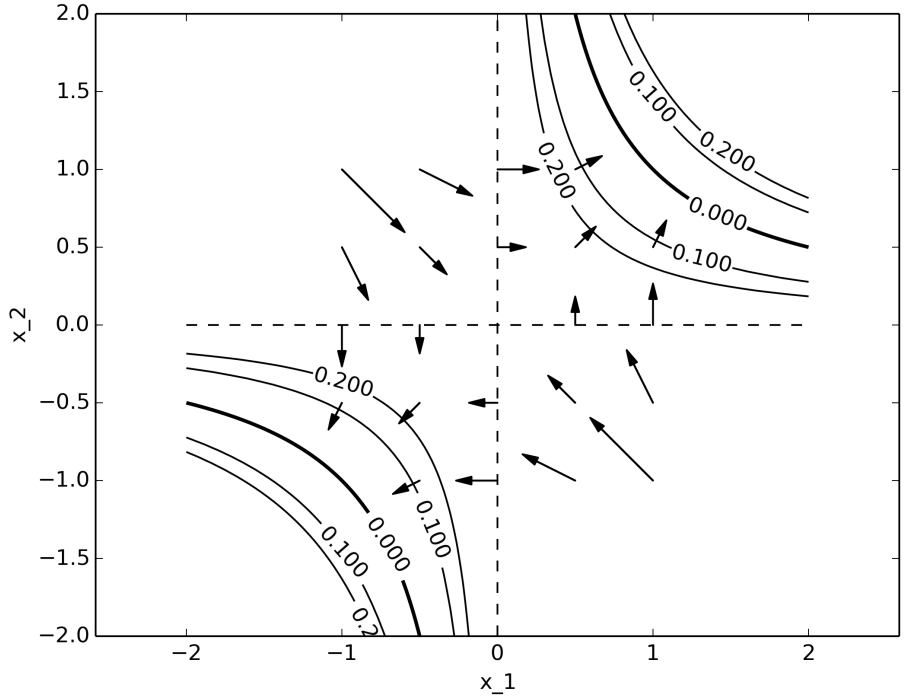


Figure 6.8: Scaled negative gradients of $f(x_1, x_2) = \frac{1}{2}(x_1 x_2 - 1)^2$

gradient descent for suitable L, so that we get sufficient decrease

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{2L}\|\nabla f(\mathbf{x}_t)\|^2, \quad t \geq 0$$

by Lemma 3.7.

This already shows that gradient descent cannot converge to a saddle point: all these have (at least two) zero entries and therefore function value $1/2$. But for starting point $\mathbf{x}_0 \in X$, we have $f(\mathbf{x}_0) < 1/2$, so we can never reach a saddle while decreasing f.

But doesn't this mean that we necessarily *have* to converge to a global minimum? No, because the sublevel sets of f are unbounded, so it could in principle happen that gradient descent runs off to infinity while constantly improving $f(\mathbf{x}_t)$ (an example is gradient descent on $f(x) = e^{-x}$). Or some

other bad behavior occurs (we haven't characterized what can go wrong). So there is still something to prove.

How about convergence from other starting points? For $\mathbf{x} > \mathbf{0}, \prod_k \mathbf{x}_k \geq 1$, we also get convergence (Exercise 43). But there are also starting points from which gradient descent will not converge to a global minimum (Exercise 44).

The following simple lemma is the key to showing that gradient descent behaves nicely in our case.

Definition 6.4. *Let $\mathbf{x} > \mathbf{0}$ (componentwise) , and let $c \geq 1$ be a real number.* $\mathbf{x}$ *is called* c-balanced *if $x_i \leq c x_j$ for all $1 \leq i, j \leq d$.*

In fact, any initial iterate $\mathbf{x}_0 > \mathbf{0}$ is c-balanced for some (possibly large) c.

Lemma 6.5. *Let $\mathbf{x} > \mathbf{0}$ be c-balanced with $\prod_k x_k \leq 1$. Then for any stepsize $\gamma > 0$, $\mathbf{x}' := \mathbf{x} - \gamma \nabla f(\mathbf{x})$ satisfies $\mathbf{x}' \geq \mathbf{x}$ (componentwise) and is also c-balanced.*

If $c = 1$ (all entries of $\mathbf{x}$ are equal), this is easy to see since then also all entries of $\nabla f(\mathbf{x})$ in (6.4) are equal. Later we will show that for suitable step size, we also maintain that $\prod_k x'_k \leq 1$, so that gradient descent only goes through balanced iterates.

Proof. Set $\Delta := -\gamma(\prod_k x_k - 1)(\prod_k x_k) \geq 0$. Then the gradient descent update assumes the form

$$x'_k = x_k + \frac{\Delta}{x_k} \geq x_k, \quad k = 1, \ldots, d.$$

For i, j, we have $x_i \leq c x_j$ and $x_j \leq c x_i$ ($\Leftrightarrow 1/x_i \leq c/x_j$). We therefore get

$$x'_i = x_i + \frac{\Delta}{x_i} \leq c x_j + \frac{\Delta c}{x_j} = c x'_j.$$

□

6.2.3 Smoothness along the trajectory

It will turn out that our function f—despite not being globally smooth—is smooth over the trajectory of gradient descent, assuming that we start with $\mathbf{x}_0 > \mathbf{0}, \prod_k (\mathbf{x}_0)_k < 1$. We will derive this from bounded Hessians. Let us therefore start by computing the Hessian matrix $\nabla^2 f(\mathbf{x})$, where by

definition, $\nabla^2 f(\mathbf{x})_{ij}$ is the j-th partial derivative of the i-th entry of $\nabla f(\mathbf{x})$. This i-th entry is

$$(\nabla f)_i = \left(\prod_k x_k - 1\right) \prod_{k \neq i} x_k$$

and its j-th partial derivative is therefore

$$\nabla^2 f(\mathbf{x})_{ij} = \begin{cases} \left(\prod_{k \neq i} x_i\right)^2, & j = i \\ 2 \prod_{k \neq i} x_k \prod_{k \neq j} x_k - \prod_{k \neq i,j} x_k, & j \neq i \end{cases}$$

This looks promising: if $\prod_k x_k \leq 1$, then we would also expect that the products $\prod_{k \neq i} x_k$ and $\prod_{k \neq i,j} x_k$ are small, in which case all entries of the Hessian are small, giving us a bound on $\|\nabla f^2 \mathbf{x}\|$ that we need to establish smoothness of f. However, for general $\mathbf{x}$, this fails. If $\mathbf{x}$ contains entries close to 0, it may happen that some terms $\prod_{k \neq i} x_k$ and $\prod_{k \neq i,j} x_k$ are actually very large.

What comes to our rescue is again c-balancedness.

Lemma 6.6. *Suppose that $\mathbf{x} > \mathbf{0}$ is c-balanced (Definition 6.4). Then for any $I \subseteq \{1, \ldots, d\}$, we have*

$$\left(\frac{1}{c}\right)^{|I|} \left(\prod_k x_k\right)^{1-|I|/d} \quad \leq \quad \prod_{k \notin I} x_k \quad \leq \quad c^{|I|} \left(\prod_k x_k\right)^{1-|I|/d}.$$

Proof. For any i, we have $x_i^d \geq (1/c)^d \prod_k x_k$ by balancedness, hence $x_i \geq (1/c)(\prod_k x_k)^{1/d}$. It follows that

$$\prod_{k \notin I} x_k = \frac{\prod_k x_k}{\prod_{i \in I} x_i} \leq \frac{\prod_k x_k}{(1/c)^{|I|} (\prod_k x_k)^{|I|/d}} = c^{|I|} \left(\prod_k x_k\right)^{1-|I|/d}.$$

The lower bound follows in the same way from $x_i^d \leq c^d \prod_k x_k$. □

This lets us bound the Hessians of c-balanced points.

Lemma 6.7. *Let $\mathbf{x} > \mathbf{0}$ be c-balanced with $\prod_k x_k \leq 1$. Then*

$$\left\|\nabla^2 f(\mathbf{x})\right\| \leq \left\|\nabla^2 f(\mathbf{x})\right\|_F \leq 3dc^2.$$

where $\|\cdot\|_F$ is the Frobenius norm and $\|\cdot\|$ the spectral norm.

Proof. The fact that $\|A\| \leq \|A\|_F$ is Exercise 45. To bound the Frobenius norm, we use the previous lemma to compute

$$\left|\nabla^2 f(\mathbf{x})_{ii}\right| = \left|\left(\prod_{k\neq i} x_i\right)^2\right| \leq c^2$$

and for $i \neq j$,

$$\left|\nabla^2 f(\mathbf{x})_{ij}\right| \leq \left|2\prod_{k\neq i} x_k \prod_{k\neq j} x_k\right| + \left|\prod_{k\neq i,j} x_k\right| \leq 3c^2.$$

Hence, $\|\nabla^2 f(\mathbf{x})\|_F^2 \leq 9d^2c^4$. Taking square roots, the statement follows. □

This now implies smoothness of f along the whole trajectory of gradient descent, under the usual "smooth stepsize" $\gamma = 1/L = 1/3dc^2$.

Lemma 6.8. *Let $\mathbf{x} > \mathbf{0}$ be c-balanced with $\prod_k x_k < 1$, $L = 3dc^2$. Let $\gamma := 1/L$. We already know from Lemma 6.5 that*

$$\mathbf{x}' := \mathbf{x} - \gamma\nabla f(\mathbf{x}) \geq \mathbf{x}$$

is c-balanced. Furthermore, (and this is the statement of the lemma), f is smooth with parameter L over the line segment connecting $\mathbf{x}$ and $\mathbf{x}'$. Lemma 6.3 (no overshooting) also yields $\prod_k x'_k \leq 1$.

Proof. Image traveling from $\mathbf{x}$ to $\mathbf{x}'$ along the line segment. As long as the product of all variables remains bounded by 1, Hessians remain bounded by Lemma 6.7, and f is smooth over the part of the segment traveled so far, by Lemma 6.1. So f can only fail to be smooth over the whole segment when there is $\mathbf{y} \neq \mathbf{x}'$ on the segment such that $\prod_k y_k = 1$. Consider the first such $\mathbf{y}$. Note that f is still smooth with parameter L over the segment connecting $\mathbf{x}$ and $\mathbf{y}$. Also, $\nabla f(\mathbf{x}) \neq \mathbf{0}$ (due to $\mathbf{x} > \mathbf{0}, \prod_k x_k < 1$), so $\mathbf{x}$ is not a critical point, and $\mathbf{y}$ results from $\mathbf{x}$ by a gradient descent step with stepsize $< 1/L$ (stepsize $1/L$ takes us to $\mathbf{x}'$). Hence, $\mathbf{y}$ is also not a critical point by Lemma 6.3, and we can't have $\prod_k y_k = 1$.

Consequently, f is smooth over the whole line segment connecting $\mathbf{x}$ and $\mathbf{x}'$. □

6.2.4 Convergence

Theorem 6.9. *Let $c \geq 1$ and $\delta > 0$ such that $\mathbf{x}_0 > \mathbf{0}$ is c-balanced with $\delta \leq \prod_k (\mathbf{x}_0)_k < 1$. Choosing stepsize*

$$\gamma = \frac{1}{3dc^2},$$

gradient descent satisfies

$$f(\mathbf{x}_T) \leq \left(1 - \frac{\delta^2}{3c^4}\right)^T f(\mathbf{x}_0), \quad T \geq 0.$$

This means that the loss indeed converges to its optimal value 0, and does so with a fast exponential error decrease. Exercise 46 asks you to prove that also the iterates themselves converge (to an optimal solution), so gradient descent will not run off to infinity.

Proof. For each $t \geq 0$, f is smooth over $\operatorname{conv}(\{\mathbf{x}_t, \mathbf{x}_{t+1}\})$ with parameter $L = 3dc^2$, hence Lemma 3.7 yields sufficient decrease:

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{6dc^2} \left\|\nabla f(\mathbf{x}_t)\right\|^2. \tag{6.5}$$

For every c-balanced $\mathbf{x}$ with $\delta \leq \prod_k x_k \leq 1$, we have

$$\begin{aligned}
\left\|\nabla f(\mathbf{x})\right\|^2 &= 2f(\mathbf{x}) \sum_{i=1}^{d} \left(\prod_{k \neq i} x_k\right)^2 \\
&\geq 2f(\mathbf{x}) \frac{d}{c^2} \left(\prod_k x_k\right)^{2-2/d} \qquad \text{(Lemma 6.6)} \\
&\geq 2f(\mathbf{x}) \frac{d}{c^2} \left(\prod_k x_k\right)^2 \\
&\geq 2f(\mathbf{x}) \frac{d}{c^2} \delta^2.
\end{aligned}$$

Then, (6.5) further yields

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{1}{6dc^2} 2f(\mathbf{x}_t) \frac{d}{c^2} \delta^2 = f(\mathbf{x}_t) \left(1 - \frac{\delta^2}{3c^4}\right),$$

proving the theorem. □

This looks great: just as for strongly convex functions, we seem to have fast convergence since the function value goes down by a constant factor in each step. There is a catch, though. To see this, consider the starting solution $\mathbf{x}_0 = (1/2, \ldots, 1/2)$. This is c-balanced with $c = 1$, but the δ that we get is $1/2^d$. Hence, the "constant factor" is

$$\left(1 - \frac{1}{3 \cdot 4^d}\right),$$

and we need $T \approx 4^d$ to reduce the initial error by a constant factor not depending on d.

Indeed, for this starting value $\mathbf{x}_0$, the gradient is exponentially small, so we are crawling towards the optimum at exponentially small speed. In order to get polynomial-time convergence, we need to start with a δ that decays at most polynomially with d. For large d, this requires us to start very close to optimality. As a concrete example, let us try to achieve a constant δ (not depending on d) with a 1-balanced solution of the form $x_i = (1 - b/d)$ for all i. For this, we need that

$$\left(1 - \frac{b}{d}\right)^d \approx e^{-b} = \Omega(1),$$

and this requires $b = O(1)$. Hence, we need to start at distance $O(1/\sqrt{d})$ from the optimal solution $(1, \ldots, 1)$.

The problem is due to constant stepsize. Indeed, f is locally much smoother at small $\mathbf{x}_0$ than Lemma 6.8 predicts, so we could afford much larger steps in the beginning. The lemma covers the "worst case" when we are close to optimality already.

So could we improve using a time-varying stepsize? The question is moot: if we know the function f under consideration, we do not need to run any optimization in the first place. The question we were trying to address is whether and how a *standard* gradient descent algorithm is able to optimize nonconvex functions as well. Above, we have given a (partially satisfactory) answer for a concrete function: yes, it can, but at a very slow rate, if d is large and the starting point not close to optimality yet.

6.3 Exercises

Exercise 40. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be convex and twice differentiable, with $X \subseteq \mathbf{dom}(f)$ an open convex set, and suppose that f is smooth with parameter L over X. Prove that under these conditions, $\|\nabla^2 f(\mathbf{x})\| \leq L$ for all $\mathbf{x} \in X$, where $\|\cdot\|$ is the spectral norm.*

Exercise 41. *Prove that the statement of Theorem 6.2 implies that*

$$\lim_{t\to\infty} \|\nabla f(\mathbf{x}_t)\|^2 = 0.$$

Exercise 42. *Prove Lemma 6.3 (gradient descent does not overshoot on smooth functions).*

Exercise 43. *Consider the function $f(\mathbf{x}) = \frac{1}{2}\left(\prod_{k=1}^{d} x_k - 1\right)^2$. Prove that for any starting point $\mathbf{x}_0 \in X = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{x} > \mathbf{0}, \prod_k \mathbf{x}_k \geq 1\}$ and any $\varepsilon > 0$, gradient descent attains $f(\mathbf{x}_T) \leq \varepsilon$ for some iteration T.*

Exercise 44. *Consider the function $f(\mathbf{x}) = \frac{1}{2}\left(\prod_{k=1}^{d} x_k - 1\right)^2$. Prove that for even dimension $d \geq 2$, there is a point $\mathbf{x}_0$ (not a critical point) such that gradient descent does not converge to a global minimum when started at $\mathbf{x}_0$, regardless of step size(s).*

Exercise 45. *Prove that for any matrix A, $\|A\| \leq \|A\|_F$, where $\|\cdot\|$ is the spectral norm and $\|\cdot\|_F$ the Frobenius norm.*

Exercise 46. *Prove that the sequence $(\mathbf{x}_T)_{T\geq 0}$ of iterates in Theorem 6.9 converges to a an optimal solution $\mathbf{x}^\star$.*

Chapter 7

The Frank-Wolfe Algorithm

Contents

7.1 Overview

As constrained optimization problems do appear often in practice, we will give them a second look here. We again consider problems of the form

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in X, \end{array} \tag{7.1}$$

which we have introduced already in Section 2.4.3.

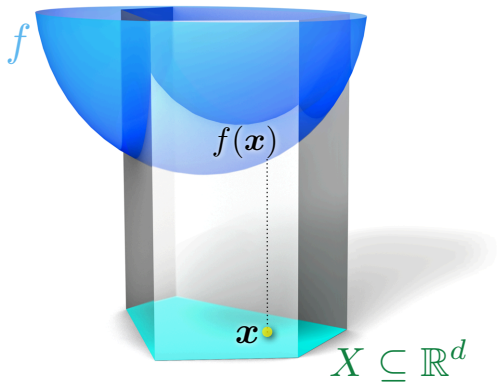


Figure 7.1: A constrained optimization problem in dimension $d = 2$.

The only algorithm we have discussed for this case was projected gradient descent in Chapter 4. This comes with a clear downside that projections onto a set X can sometimes be very complex to compute, even in cases when the set is convex. Would it still be possible to solve constrained optimization problems using a gradient-based algorithm, but without any projection steps?

From a different perspective, coordinate descent, as we have discussed in Chapter 5, had the attractive advantage that it only modified one coordinate in every step, keeping all others unchanged. Yet, it is not applicable in the general constrained case, as we can not easily know when a coordinate step would exit the constraint set X (except in easy cases when X is defined as a product of intervals). Is there a coordinate-like algorithm also for general constraint sets X?

It turns out the answer to both previous questions is yes. The algorithm was discovered by Marguerite Frank and Philip Wolfe in 1956 [FW56],

giving rise to the name of the method. Historically, the motivation for the method was different from the two aspects mentioned above. After the second world war, linear programming (that is to minimize a linear function over set of linear constraints) had significant impact for many industrial applications (e.g. in logistics). Given these successes with linear objectives, Marguerite Frank and Philip Wolfe studied if similar methods could be generalized to non-linear objectives (including quadratic as well as general objectives), that is problems of the form (7.1).

7.2 The Algorithm

Similar to projected gradient descent, the Frank-Wolfe algorithm uses a nontrivial primitive. Here, it is the *linear minimization oracle* (LMO). For the feasible region $X \subseteq \mathbb{R}^d$ and an arbitrary vector $\mathbf{g} \in \mathbb{R}^d$ (which we can think of as an optimization direction),

$$\mathrm{LMO}_X(\mathbf{g}) := \operatorname*{argmin}_{\mathbf{z} \in X} \mathbf{g}^\top \mathbf{z} \tag{7.2}$$

is any minimizer of the linear function $\mathbf{g}^\top \mathbf{z}$ over X. We will assume that a minimizer exists whenever we apply the oracle. If X is closed and bounded, this is guaranteed.

The Frank-Wolfe algorithm proceeds iteratively, starting from an initial feasible point $\mathbf{x}_0 \in X$, using a (time-dependent) stepsize $\gamma_t \in [0, 1]$.

$$\mathbf{s} := \mathrm{LMO}_X(\nabla f(\mathbf{x}_t)), \tag{7.3}$$

$$\mathbf{x}_{t+1} := (1 - \gamma_t)\mathbf{x}_t + \gamma_t \mathbf{s}, \tag{7.4}$$

We immediately see that the algorithm reduces non-linear constrained optimization to linear optimization over the same set X: It is able to solve general non-linear constrained optimization problems (7.1), by only solving a simpler linear constrained optimization over the same set X in each iteration — that is the call to the linear minimization oracle LMO_X (7.2).

But which linear problem is actually helpful to solve in each step — that is which direction should we give to the linear oracle LMO_X? The Frank-Wolfe algorithm uses the gradient $\mathbf{g} = \nabla f(\mathbf{x}_t)$. The rationale is that the gradient defines the best linear approximation of f at $\mathbf{x}_t$. In each step, the algorithm minimizes this linear approximation over the set X

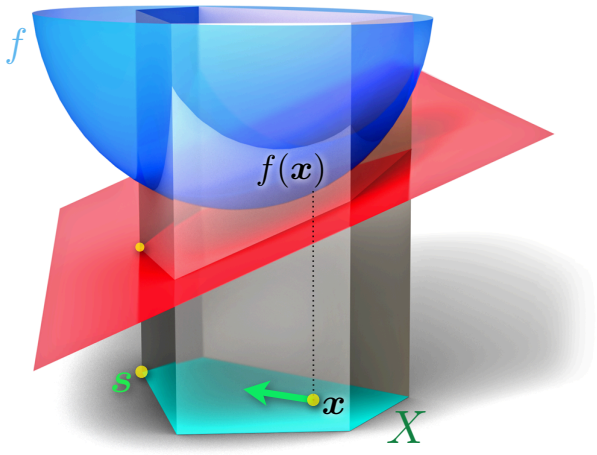


Figure 7.2: Illustration of a Frank-Wolfe step from an iterate $\mathbf{x}$.

and makes a step into the direction of the minimizer; see Figure 7.2.

We identify several attractive properties of this algorithm:

- Iterates are *always feasible*, if the constraint set X is convex. In other words, $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t \in X$. This follows thanks to the definition of the linear minimization oracle returning a point $\mathbf{s}$ within X, and the fact that the next iterate $\mathbf{x}_{t+1}$ is on the line segment $[\mathbf{s}, \mathbf{x}_t]$, for $\gamma_t \in [0, 1]$. This requires that the stepsize in each iteration is chosen in $0 \leq \gamma_t \leq 1$. We postpone the further discussion of the stepsizes to later when we give the convergence analysis.

- The algorithm is *projection-free*. As we are going to see later, depending on the geometry of the constraint set X, the subproblem LMO_X is often easier to solve than a projection onto the same set X. Intuitively, this the case because LMO_X is only a linear problem, while a projection operation is a quadratic optimization problem.

- The iterates always have a simple *sparse representation*: $\mathbf{x}_t$ is always a convex combination of the initial iterate and the minimizers $\mathbf{s}$ used so far. We will come back to this point in Section 7.6 below.

7.3 On linear minimization oracles

The algorithm is particularly useful for cases when the constraint set X can be described as a convex hull of a finite or otherwise "nice" set of points $\mathcal{A}$, formally $\mathrm{conv}(\mathcal{A}) = X$. We call $\mathcal{A}$ the *atoms* describing the constraint set.

In this case, a solution to the linear subproblem LMO_X defined in (7.2) is always attained by an atom $\mathbf{a} \in \mathcal{A}$. Indeed, every $\mathbf{s} \in \mathrm{conv}(X)$ is a convex combination $\mathbf{s} = \sum_{i=1}^{n} \lambda_i \mathbf{a}_i$ of finitely many atoms ($\sum_{i=1}^{n} \lambda_i = 1$, all λ_i nonnegative). It follows that for every $\mathbf{g}$, there is always an atom such that $\mathbf{g}^\top \mathbf{s} \geq \mathbf{a}_i^\top \mathbf{g}$. Hence, if $\mathbf{s}$ minimizes $\mathbf{g}^\top \mathbf{z}$, then there is also an atomic minimizer.

This allows us to significantly reduce the candidate solutions for the step directions used by the Frank-Wolfe algorithm. (Note that subproblem (7.2) might still have optimal solutions which are not atoms, but there is always at least one atomic solution $\mathrm{LMO}_X(\mathbf{g}) \in \mathcal{A}$).

The set $\mathcal{A} = X$ is a valid (but not too useful) set of atoms. The "optimal" set of atoms is the set of *extreme* points. A point $\mathbf{x} \in X$ is extreme if $\mathbf{x} \notin \mathrm{conv}(X \setminus \{\mathbf{x}\})$. Such an extreme point must be in every set of atoms, but not every atom must be extreme. All that we require for $\mathcal{A}$ to be a set of atoms is that $\mathrm{conv}(\mathcal{A}) = X$.

We give two interesting examples next.

7.3.1 LASSO and the ℓ_1-ball

The LASSO problem in its standard (primal) form is given as

$$\min_{\mathbf{x} \in \mathbb{R}^d} \|A\mathbf{x} - \mathbf{b}\|^2 \text{ subject to } \|\mathbf{x}\|_1 \leq 1 \tag{7.5}$$

Here we observe that the constraint set $X = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 \leq 1\}$ is the unit ℓ_1-ball, the convex hull of the unit basis vectors: $X = \mathrm{conv}(\{\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_d\})$.

Linear problems over the unit ℓ_1-ball are easy to solve: For any direction $\mathbf{g}$, the minimizer can be chosen as one of the atoms (the unit basis

vectors and their negatives):

$$\begin{aligned}
\mathrm{LMO}_X(\mathbf{g}) &= \operatorname*{argmin}_{\mathbf{z}\in X} \mathbf{z}^\top \mathbf{g} \\
&= \operatorname*{argmin}_{\mathbf{z}\in\{\pm\mathbf{e}_1,\ldots,\pm\mathbf{e}_n\}} \mathbf{z}^\top \mathbf{g} \qquad (7.6)\\
&= -\mathrm{sgn}(g_i)\mathbf{e}_i \text{ with } i := \operatorname*{argmax}_{i\in[d]} |g_i| \qquad (7.7)
\end{aligned}$$

So we only have to look at the vector $\mathbf{g}$ and identify its largest coordinate (in absolute value). This operation is of course significantly more efficient than projection onto an ℓ_1-ball. The latter we have analyzed in Section 4.5 and have shown a more sophisticated algorithm that still did not have runtime linear in the dimension.

7.3.2 Semidefinite Programming and the Spectahedron

Hazan's algorithm [Haz08] is an application of the Frank-Wolfe algorithm to semidefinite programming. We use the notation of Gärtner and Matoušek [GM12, Chapter 5]. In Hazan's algorithm, each LMO assumes the form

$$\begin{array}{ll} \text{argmin} & G \bullet Z \\ \text{subject to} & \mathrm{Tr}(Z) = 1 \\ & Z \succeq 0. \end{array} \qquad (7.8)$$

Here, the feasible region X is the *spectahedron*, the set of all (symmetric) positive semidefinite matrices $Z \in \mathbb{R}^{d\times d}$ of trace 1, and G is a symmetric matrix. For two square matrices A and B, the notation $A \bullet B$ stands for their "scalar product" $\sum_{i,j} a_{ij}b_{ij}$, so $G \bullet Z$ is the matrix analog of $\mathbf{g}^\top\mathbf{z}$. In fact, (7.8) is a semidefinite program itself, but of a simple form that allows an explicit solution, as we show next.

The atoms of the spectahedron turn out to be the matrices of the form $\mathbf{z}\mathbf{z}^\top$ with $\mathbf{z} \in \mathbb{R}^d$, $\|\mathbf{z}\| = 1$ (these are positive semidefinite of trace 1). It remains to show that every positive semidefinite matrix of trace 1 is a convex combination of suitable atoms. To see this, we diagonalize such a matrix Z as $Z = TDT^\top$ where T is orthogonal and D is diagonal, again of trace 1. The diagonal elements $\lambda_1, \ldots, \lambda_d$ are the (nonnegative) eigenvalues of Z, summing up to the trace 1. Let $\mathbf{a}_i$ be the i-th column of T. As T is orthogonal, we have $\|\mathbf{a}_i\| = 1$. It follows that $Z = \sum_{i=1}^d \lambda_i \mathbf{a}_i\mathbf{a}_i^\top$ is the de-

sired convex combination of atoms. We remark that $\mathbf{a}_i$ is a (unit length) eigenvector of Z w.r.t. eigenvalue λ_i.

Lemma 7.1. *Let λ_1 be the smallest eigenvalue of G, and let $\mathbf{s}_1$ be a corresponding eigenvector of unit length. Then we can choose $\mathrm{LMO}_X(G) = \mathbf{s}_1\mathbf{s}_1^\top$.*

Proof. Since it is sufficient to minimize over atoms, we have

$$\min_{\mathrm{Tr}(Z)=1, Z\succeq 0} G \bullet Z = \min_{\|\mathbf{z}\|=1} G \bullet \mathbf{z}\mathbf{z}^\top = \min_{\|\mathbf{z}\|=1} \mathbf{z}^\top G \mathbf{z} = \lambda_1.$$

The second equality follows from $G \bullet \mathbf{z}\mathbf{z}^\top = \mathbf{z}^\top G \mathbf{z}$ for all $\mathbf{z}$ (simple rewriting), and the last equality is a standard result from linear algebra that can be proved via elementary calculations, involving diagonalization of G.

Now, $\mathbf{s}_1$ is easily seen to attain the last minimum, hence $\mathbf{s}_1\mathbf{s}_1^\top$ attains the first minimum, and $\mathrm{LMO}_X(G) = \mathbf{s}_1\mathbf{s}_1^\top$ follows. □

7.4 Duality gap — A certificate for optimization quality

A rather unexpected side benefit of the linear minimization oracle is that it can be used as a *certificate of the optimization quality* at our current iterate. Even if the true optimum value $f(\mathbf{x}^\star)$ of the problem is unknown, the point $\mathbf{s}$ returned by $\mathrm{LMO}_X(\nabla f(\mathbf{x}_t))$ lets us compute an upper bound on the *optimality gap* $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$.

Given $\mathbf{x} \in X$, we define the *duality gap* (also known as the Hearn gap) at $\mathbf{x}$ as

$$g(\mathbf{x}) := \nabla f(\mathbf{x})^\top(\mathbf{x} - \mathbf{s}) \quad \text{for} \quad \mathbf{s} := \mathrm{LMO}_X(\nabla f(\mathbf{x})). \tag{7.9}$$

Note that $g(\mathbf{x})$ is well-defined since it only depends on the minimum value $\nabla f(\mathbf{x})^\top \mathbf{s}$ of $\mathrm{LMO}_X(\nabla f(\mathbf{x}))$, but not on the concrete minimizer $\mathbf{s}$ of which there may be many. The duality gap $g(\mathbf{x})$ can be interpreted as the optimality gap $\nabla f(\mathbf{x})^\top \mathbf{x} - \nabla f(\mathbf{x})^\top \mathbf{s}$ of the linear subproblem. In particular, $g(\mathbf{x}) \geq 0$; see Figure 7.3.

Lemma 7.2. *Suppose that the constrained minimization problem (7.1) has a minimizer $\mathbf{x}^\star$. Let $\mathbf{x} \in X$. Then*

$$g(\mathbf{x}) \geq f(\mathbf{x}) - f(\mathbf{x}^\star),$$

meaning that the duality gap is an upper bound for the optimality gap.

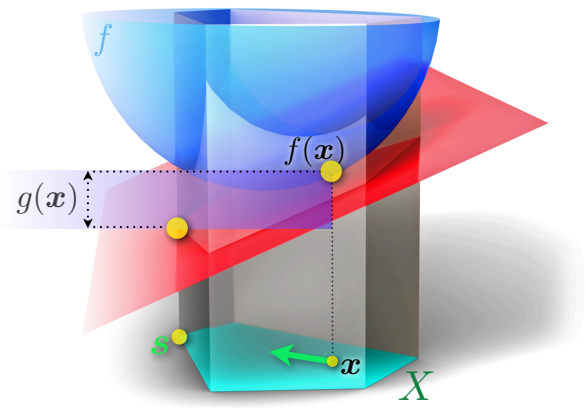


Figure 7.3: Illustration of the duality gap at iterate $\mathbf{x}$.

Proof. Using that $\mathbf{s}$ minimizes $\nabla f(\mathbf{x})^\top \mathbf{z}$ over X, we argue that

$$\begin{aligned} g(\mathbf{x}) &= \nabla f(\mathbf{x})^\top (\mathbf{x} - \mathbf{s}) \\ &\geq \nabla f(\mathbf{x})^\top (\mathbf{x} - \mathbf{x}^\star) \\ &\geq f(\mathbf{x}) - f(\mathbf{x}^\star) \end{aligned} \tag{7.10}$$

where in the last inequality we have used the first-order characterization of convexity of f (Lemma 2.16). □

So the duality gap $g(\mathbf{x}_t)$—a value which is available for every iteration of the Frank-Wolfe algorithm—always gives us a guaranteed upper bound on the unknown error $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$. This contrasts unconstrained optimization, where we don't have any such certificate in general.

We argue that it is also a useful upper bound. At any optimal point $\mathbf{x}^\star$ of the constrained optimization problem, the gap vanishes, i.e. $g(\mathbf{x}^\star) = 0$. This follows from the optimality conditions for constrained convex optimization, given in Lemma 2.28, stating that $\nabla f(\mathbf{x}^\star)^\top (\mathbf{x} - \mathbf{x}^\star) \geq 0 \quad \forall \mathbf{x} \in X$.

7.5 Convergence in $\mathcal{O}(1/\varepsilon)$ steps

We first address the standard way of choosing the stepzise in the Frank-Wolfe algorithm. We need to assume that the function f is smooth, but unlike for gradient descent, the stepsize can be chosen independently from the smoothness parameter.

7.5.1 Convergence analysis for $\gamma_t = 2/(t+2)$

Theorem 7.3. *Consider the constrained minimization problem (7.1) where $f : \mathbb{R}^d \to \mathbb{R}$ is convex and smooth with parameter L, and X is convex, closed and bounded (in particular, a minimizer $\mathbf{x}^\star$ of f over X exists, and all linear minimization oracles have minimizers). With any $\mathbf{x}_0 \in X$, and with stepsizes $\gamma_t = 2/(t+2)$, the Frank-Wolfe algorithm yields*

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{2L \operatorname{diam}(X)^2}{T+1}, \quad T \geq 1,$$

where $\operatorname{diam}(X) := \max_{\mathbf{x},\mathbf{y} \in X} \|\mathbf{x} - \mathbf{y}\|$ is the diameter of X (which exists since X is closed and bounded).

The following descent lemma forms the core of the convergence proof:

Lemma 7.4. *For a step $\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma_t(\mathbf{s} - \mathbf{x}_t)$ with stepsize $\gamma_t \in [0,1]$, it holds that*

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \gamma_t g(\mathbf{x}_t) + \gamma_t^2 \frac{L}{2} \|\mathbf{s} - \mathbf{x}_t\|^2,$$

where $\mathbf{s} = \operatorname{LMO}_X(\nabla f(\mathbf{x}_t))$.

Proof. From the definition of smoothness of f, we have

$$\begin{aligned} f(\mathbf{x}_{t+1}) &= f(\mathbf{x}_t + \gamma_t(\mathbf{s} - \mathbf{x}_t)) \\ &\leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top \gamma_t(\mathbf{s} - \mathbf{x}_t) + \gamma_t^2 \frac{L}{2} \|\mathbf{s} - \mathbf{x}_t\|^2 \qquad (7.11) \\ &= f(\mathbf{x}_t) - \gamma_t g(\mathbf{x}_t) + \gamma_t^2 \frac{L}{2} \|\mathbf{s} - \mathbf{x}_t\|^2, \end{aligned}$$

using the definition (7.9) of the duality gap. □

Proof of Theorem 7.3. Writing $h(\mathbf{x}) := f(\mathbf{x}) - f(\mathbf{x}^\star)$ for the (unknown) optimization gap at point $\mathbf{x}$, und using the certificate property (7.10) of the duality gap, that is $h(\mathbf{x}) \leq g(\mathbf{x})$, Lemma 7.4 implies that

$$\begin{aligned} h(\mathbf{x}_{t+1}) &\leq h(\mathbf{x}_t) - \gamma_t g(\mathbf{x}_t) + \gamma_t^2 \frac{L}{2} \|\mathbf{s} - \mathbf{x}_t\|^2 \\ &\leq h(\mathbf{x}_t) - \gamma_t h(\mathbf{x}_t) + \gamma_t^2 \frac{L}{2} \|\mathbf{s} - \mathbf{x}_t\|^2 \\ &= (1 - \gamma_t) h(\mathbf{x}_t) + \gamma_t^2 \frac{L}{2} \|\mathbf{s} - \mathbf{x}_t\|^2 \\ &\leq (1 - \gamma_t) h(\mathbf{x}_t) + \gamma_t^2 C, \qquad (7.12) \end{aligned}$$

where $C := \frac{L}{2}\operatorname{diam}(X)^2$.

The convergence proof finishes by induction. Exercise 47 asks you to prove that for $\gamma_t = \frac{2}{t+2}$, we obtain

$$h(\mathbf{x}_t) \leq \frac{4C}{t+1}, \quad t \geq 1.$$

□

7.5.2 Stepsize variants

The previous runtime analysis also holds for two alternative stepsizes. In practice, convergence might even be faster with these alternatives, since they are trying to optimize progress, in two different ways. For both alternative stepsizes, we will establish inequality (7.12) with the standard stepsize $\gamma_t = 2/(t+2) =: \mu_t$ from which $h(\mathbf{x}_t) \leq 4C/(t+1)$ follows.

Line search stepsize. Here, $\gamma_t \in [0,1]$ is chosen such that the progress in f-value (and hence also in h-value) is maximized,

$$\gamma_t := \operatorname*{argmin}_{\gamma \in [0,1]} f\big((1-\gamma)\mathbf{x}_t + \gamma \mathbf{s}\big).$$

Let $\mathbf{y}_{t+1}$ be the iterate obtained from $\mathbf{x}_t$ with the standard stepsize μ_t. From (7.12) and the definition of γ_t, we obtain the desired inequality

$$h(\mathbf{x}_{t+1}) \leq h(\mathbf{y}_{t+1}) \leq (1-\mu_t)h(\mathbf{x}_t) + \mu_t^2 C. \tag{7.13}$$

Gap-based stepsize. This chooses γ_t such that the right-hand side in the first line of (7.12) is minimized. A simple calculation shows that this results in

$$\gamma_t := \min\left(\frac{g(\mathbf{x}_t)}{L\left\|\mathbf{s}-\mathbf{x}_t\right\|^2}, 1\right).$$

Now we establish (7.13) as follows:

$$\begin{aligned}
h(\mathbf{x}_{t+1}) &\leq h(\mathbf{x}_t) - \gamma_t g(\mathbf{x}_t) + \gamma_t^2 \frac{L}{2}\left\|\mathbf{s}-\mathbf{x}_t\right\|^2 \\
&\leq h(\mathbf{x}_t) - \mu_t g(\mathbf{x}_t) + \mu_t^2 \frac{L}{2}\left\|\mathbf{s}-\mathbf{x}_t\right\|^2 \\
&\leq h(\mathbf{x}_t) - \mu_t h(\mathbf{x}_t) + \mu_t^2 \frac{L}{2}\left\|\mathbf{s}-\mathbf{x}_t\right\|^2 \\
&\leq (1-\mu_t)h(\mathbf{x}_t) + \mu_t^2 C.
\end{aligned}$$

Directly plugging in the definition of γ_t yields

$$h(\mathbf{x}_{t+1}) \leq \begin{cases} h(\mathbf{x}_t)\left(1 - \frac{\gamma_t}{2}\right), & \gamma_t < 1, \\ h(\mathbf{x}_t), & \gamma_t = 1, \end{cases}$$

So we make progress in every iteration under the gap-based stepsize (this is not guaranteed under the standard stepsize), but faster convergence is not implied.

7.5.3 Affine invariance

The convergence bound on the Frank-Wolfe method that we have developed in Theorem 7.3,

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{2L \operatorname{diam}(X)^2}{T+1},$$

is in some sense bad. Consider the problem of minimizing $f(x_1, x_2) = x_1^2 + x_2^2$ over the unit square $X = \{(x_1, x_2) : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$. The function f (the two-dimensional supermodel) is smooth with $L = 2$, and $\operatorname{diam}(X)^2 = 2$. Next consider $f'(x_1, x_2) = x_1^2 + (10x_2)^2$ over the rectangle $X' = \{(x_1, x_2) : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1/10\}$. The function f' is smooth with $L' = 200$, and $\operatorname{diam}(X')^2 = 1 + 1/100$. Hence, our convergence analysis seems to suggest that the error after T steps of the Frank-Wolfe algorithm on f' over X' is roughly 100 times larger than on f over X.

In reality, however, there is no such difference. The reason is that the two problems (f, X) and (f', X') are equivalent under rescaling of variable x_2, and the Frank-Wolfe algorithm is invariant under this and more generally all affine transformations of space. Figure 7.4 depicts the two problems (f, X) and (f', X') from our example above.

To argue about the affine invariance formally, we call two problems (f, X) and (f', X') *affinely equivalent* if $f'(\mathbf{x}) = f(A\mathbf{x}+\mathbf{b})$ for some invertible matrix A and some vector $\mathbf{b}$, and $X' = \{A^{-1}(\mathbf{x} - \mathbf{b}) : \mathbf{x} \in X\}$. The equivalence is that $\mathbf{x} \in X$ with function value $f(\mathbf{x})$ if and only if $\mathbf{x}' = A^{-1}(\mathbf{x}-\mathbf{b}) \in X'$ with the same function value $f'(\mathbf{x}') = f(AA^{-1}(\mathbf{x} - \mathbf{b}) + \mathbf{b}) = f(\mathbf{x})$. We call $\mathbf{x}'$ the vector corresponding to $\mathbf{x}$. In Figure 7.4, we have

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix}, \quad \mathbf{b} = \mathbf{0}.$$

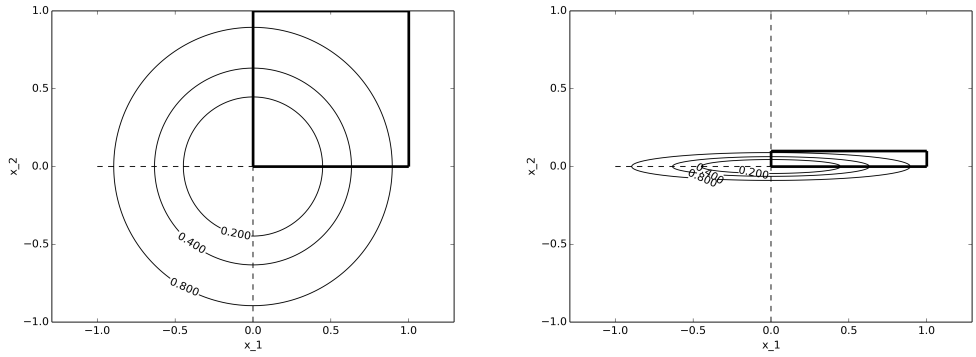


Figure 7.4: Two optimization problems (f, X) and (f', X') that are equivalent under an affine transformation.

By the chain rule, we get

$$\nabla f'(\mathbf{x}') = A^\top \nabla f(A\mathbf{x}' + \mathbf{b}) = A^\top \nabla f(\mathbf{x}). \tag{7.14}$$

Now consider performing an iteration of the Frank-Wolfe algorithm

(a) on (f, X), starting from some iterate $\mathbf{x}$, and

(b) on (f', X'), starting from the corresponding iterate $\mathbf{x}'$,

in both cases with the same stepsize. Because of

$$\nabla f'(\mathbf{x}')^\top \mathbf{z}' \overset{(7.14)}{=} \nabla f(\mathbf{x})^\top A A^{-1}(\mathbf{z} - \mathbf{b}) = \nabla f(\mathbf{x})^\top \mathbf{z} - c,$$

where c is some constant, the linear minimization oracle in (b) returns the step direction $\mathbf{s}' = A^{-1}(\mathbf{s} - \mathbf{b}) \in X'$ corresponding to the step direction $\mathbf{s} \in X$ in (a). It follows that also the next iterates in (a) and (b) will correspond to each other and have the same function values. In particular, after any number of steps, both (a) and (b) will incur the same optimization error.

7.5.4 The curvature constant

It follows from the above discussion that a good analysis of the Frank-Wolfe algorithm should provide a bound that is invariant under affine

transformations, unlike the bound of Theorem 7.3. For this, we define a *curvature constant* of the constrained optimization problem (7.1). The quantity serves as a combined notion of complexity of both the objective function f and the constraint set X:

$$C_{(f,X)} := \sup_{\substack{\mathbf{x},\mathbf{s}\in X,\gamma\in(0,1]\\ \mathbf{y}=(1-\gamma)\mathbf{x}+\gamma\mathbf{s}}} \frac{1}{\gamma^2}\big(f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{y}-\mathbf{x})\big). \tag{7.15}$$

To gain an understanding of this quantity, note that $d(\mathbf{y}) := f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{y}-\mathbf{x})$ is the pointwise vertical distance between the graph of f and its linear approximation at $\mathbf{x}$. By convexity, $\mathbf{d}(\mathbf{y}) \geq 0$ for all $\mathbf{y} \in X$. For $\mathbf{y}$ resulting from $\mathbf{x}$ by a Frank-Wolfe step with stepsize γ, we normalize the vertical distance with γ^2 (a natural choice if we think of f as being smooth), and take the supremum over all possible such normalized vertical distances.

We will see that the convergence rate of the Frank-Wolfe algorithm can be described purely in terms of this quantity, without resorting to any smoothness constants L or diameters $\operatorname{diam}(X)$. As we have already seen, the latter two quantities are not affine invariant.

In a similar way as we have done it for the algorithm itself, we can prove that the curvature constant $C_{(f,X)}$ is affine invariant. Hence, here is the envisioned good analysis of the Frank-Wolfe algorithm.

Theorem 7.5. *Consider the constrained minimization problem (7.1) where $f : \mathbb{R}^d \to \mathbb{R}$ is convex, and X is convex, closed and bounded. Let $C_{(f,X)}$ be the curvature constant (7.15) of f over X. With any $\mathbf{x}_0 \in X$, and with stepsizes $\gamma_t = 2/(t+2)$, the Frank-Wolfe algorithm yields*

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{4C_{(f,X)}}{T+1}, \quad T \geq 1.$$

Proof. The crucial step is to prove the following version of (7.11):

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \nabla f(\mathbf{x}_t)^\top \gamma_t(\mathbf{x}_t - \mathbf{s}) + \gamma_t^2 C_{(f,X)}. \tag{7.16}$$

After this, we can follow the remainder of the proof of Theorem 7.3, with $C_{(f,X)}$ instead of $C = \frac{L}{2}\operatorname{diam}(X)^2$. To show (7.16), we use

$$\mathbf{x} := \mathbf{x}_t, \quad \mathbf{y} := \mathbf{x}_{t+1} = (1-\gamma_t)\mathbf{x}_t + \gamma_t\mathbf{s}, \quad \mathbf{y} - \mathbf{x} = -\gamma_t(\mathbf{x} - \mathbf{s}),$$

and rewrite the definition of the curvature constant (7.15) to get

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \gamma_t^2 C_{(f,X)}.$$

Plugging in the previous definitions of $\mathbf{x}$ and $\mathbf{y}$, (7.16) follows. □

One might suspect this affine independent bound to be worse than the best bound obtainable from Theorem 7.3 after an affine transformation. As we show next, this not the case: when f is twice differentiable, $C_{(f,X)}$ is always bounded by the constant $C = \frac{L}{2}\operatorname{diam}(X)^2$ that determines the convergence rate in Theorem 7.3.

Lemma 7.6 (Exercise 48)**.** *Let f be a convex function which is smooth with parameter L over X. Then*

$$C_{(f,X)} \leq \frac{L}{2}\operatorname{diam}(X)^2 .$$

7.5.5 Convergence in duality gap

The following result shows that the duality gap converges as well, essentially at the same rate as the primal error. .

Theorem 7.7. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and* smooth *with parameter L, and $\mathbf{x}_0 \in X$, $T \geq 2$. Then choosing any of the stepsizes in Section 7.5.2, the Frank-Wolfe algorithm yields at $t, 1 \leq t \leq T$ such that*

$$g(\mathbf{x}_t) \leq \frac{27/2 \cdot C_{(f,X)}}{T+1}$$

Still, compared to our previous theorem, the convergence of the gap here is a stronger and more useful result, because $g(\mathbf{x}_t)$ is easy to compute in any iteration of the Frank-Wolfe algorithm, and as we have seen in (7.10) serves as an upper bound (certificate) to the unknown primal error, that is $f(\mathbf{x}_t) - f(\mathbf{x}^\star) \leq g(\mathbf{x}_t)$.

The proof of the theorem is left as Exercise 49, and is difficult. The argument leverages that not all gaps can be small, and will again crucially rely on the descent Lemma 7.4.

7.6 Sparsity, extensions and use cases

A very important feature of the Frank-Wolfe algorithm has been pointed out before, but we would like to make it explicit here. Consider the convergence bound of Theorem 7.5,

$$f(\mathbf{x}_T) - f(\mathbf{x}^\star) \leq \frac{4C_{(f,X)}}{T+1}, \quad T \geq 1.$$

This means that $O(1/\varepsilon)$ many iterations are sufficent to obtain optimality gap at most ε. At this time, the current solution is a convex combination of $\mathbf{x}_0$ and $O(1/\varepsilon)$ many atoms of the constraint set X. Thinking of ε as a constant (such as 0.01), this means that *constantly* many atoms are sufficient in order to get an almost optimal solution. This is quite remarkable, and it connects to the notion of *coresets* in computational geometry. A coreset is a small subsets of a given set of objects that is representative (with respect to some measure) for the set of all objects. Some algorithms for finding small coresets are inspired by the Frank-Wolfe algorithm [Cla10].

The algorithm and analysis above can be extended to several settings, including

- *Approximate* LMO, that is we can allow a linear minimization oracle which is not exact but is of a certain additive or multiplicative approximation quality for the subproblem (7.2). Convergence bounds are essentially as in the exact case [Jag13].
- *Randomized* LMO, that is that the LMO_X solves the linear minimization oracle only over a random subset of X. Convergence in $O(1/\varepsilon)$ steps still holds [KPd18].
- *Stochastic* LMO, that is LMO_X is fed with a stochastic gradient instead of the true gradient [HL20].
- *Unconstrained problems*. This is achieved by considering growing versions of a constraint set X. For instance when X is an ℓ_1-norm ball, the algorithm will become similar to popular steepest coordinate methods as we have discussed in Section 5.4.3. In this case, the resulting algorithms are also known as matching-pursuit, and are widely used in the literature on sparse recovery of a signal, also known as compressed sensing. For more details, we refer the reader to [LKTJ17].

The Frank-Wolfe algorithm and its variants have many popular use-cases. The most attractive uses are for constraint sets X where a projection step bears significantly more computational cost compared to solving a linear problem over X. Some examples of such sets include:

- *Lasso* and other L1-constrained problems, as discussed in Section 7.3.1.
- *Matrix Completion*. For several low-rank approximation problems, including matrix completion as in recommender systems, the Frank-Wolfe algorithm is a very scalable algorithm, and has much lower iteration cost compared to projected gradient descent. For a more formal treatment, see Exercise 50.
- Relaxation of *combinatorial problems*, where we would like to optimize over a discrete set $\mathcal{A}$ (e.g. matchings, network flows etc). In this case, the Frank-Wolfe algorithm is often used together with early stopping, in order to achieve a good iterate $\mathbf{x}_t$ being a combination of at most t of the original points $\mathcal{A}$.

Many of these applications can also be written as constraint sets of the form $X := \mathrm{conv}(\mathcal{A})$ for some set of atoms $\mathcal{A}$, as illustrated in the following table:

Examples	$\mathcal{A}$	$\lvert\mathcal{A}\rvert$	dim.	$\mathrm{LMO}_X(\mathbf{g})$
L1-ball	$\{\pm\mathbf{e}_i\}$	$2d$	d	$\pm\mathbf{e}_i$ with $\mathrm{argmax}_i \lvert g_i\rvert$
Simplex	$\{\mathbf{e}_i\}$	d	d	$\mathbf{e}_i$ with $\mathrm{argmin}_i g_i$
Spectahedron	$\{\mathbf{x}\mathbf{x}^\top, \lVert\mathbf{x}\rVert = 1\}$	∞	d^2	$\mathrm{argmin}_{\lVert\mathbf{x}\rVert=1} \mathbf{x}^\top G\mathbf{x}$
Norms	$\{\mathbf{x}, \lVert\mathbf{x}\rVert \le 1\}$	∞	d	$\mathrm{argmin}_{\mathbf{s},\lVert\mathbf{s}\rVert\le 1} \langle\mathbf{s},\mathbf{g}\rangle$
Nuclear norm	$\{Y, \lVert Y\rVert_* \le 1\}$	∞	d^2	..
Wavelets	..	∞	∞	..

7.7 Exercises

Exercise 47 (Induction for the Frank-Wolfe convergence analysis)**.** *Given some constant $C > 0$ and a sequence of real values $h_0, h_1, \ldots$ satisfying* (7.12)*, i.e.*

$$h_{t+1} \le (1-\gamma_t)h_t + \gamma_t^2 C \qquad \textit{for } t = 0, 1, \ldots$$

for $\gamma = \frac{2}{t+2}$, prove that

$$h_t \leq \frac{4C}{t+1} \qquad \textit{for } t \geq 1.$$

Exercise 48 (Relating Curvature and Smoothness)**.** *Prove Lemma 7.6:*

Exercise 49 (Duality gap convergence for the Frank-Wolfe algorithm)**.** *Prove Theorem 7.7 on the convergence of the duality gap (which is an upper bound to the primal error $f(\mathbf{x}_t) - f(\mathbf{x}^\star)$. The proof will again crucially rely on the descent Lemma 7.4.*

Exercise 50 (Frank-Wolfe for Matrix completion)**.** *Consider the matrix completion problem, that is to find a matrix Y solving*

$$\min_{Y \in X \subseteq \mathbb{R}^{n \times m}} \sum_{(i,j) \in \Omega} (Z_{ij} - Y_{ij})^2$$

where the optimization domain X is the set of matrices in the unit ball of the trace norm (or nuclear norm), which is defined the convex hull of the rank-1 matrices

$$X := conv(\mathcal{A}) \;\; \textit{with} \;\; \mathcal{A} := \left\{ \mathbf{u}\mathbf{v}^\top \;\middle|\; \begin{smallmatrix} \mathbf{u} \in \mathbb{R}^n, \, \|\mathbf{u}\|_2 = 1 \\ \mathbf{v} \in \mathbb{R}^m, \, \|\mathbf{v}\|_2 = 1 \end{smallmatrix} \right\}.$$

Here $\Omega \subseteq [n] \times [m]$ is the set of observed entries from a given data matrix Z (collecting the ratings given by users to items for example).

1. *Derive the LMO_X for this set X for a gradient at iterate $Y \in \mathbb{R}^{n \times m}$.*
2. *Derive the* projection *step onto X. How do the LMO_X and the projection step compare, in terms of computational cost?*

Chapter 8

Newton's Method

Contents

8.1 1-dimensional case

The Newton method (or Newton-Raphson method, invented by Sir Isaac Newton and formalized by Joseph Raphson) is an iterative method for finding a zero of a differentiable univariate function $f : \mathbb{R} \to \mathbb{R}$. Starting from some number x_0, it computes

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}, \quad t \geq 0. \tag{8.1}$$

Figure 8.1 shows what happens. x_{t+1} is the point where the tangent line to the graph of f at $(x_t, f(x_t))$ intersects the x-axis. In formulas, x_{t+1} is the solution of the linear equation

$$f(x_t) + f'(x_t)(x - x_t) = 0,$$

and this yields the update formula (8.1).

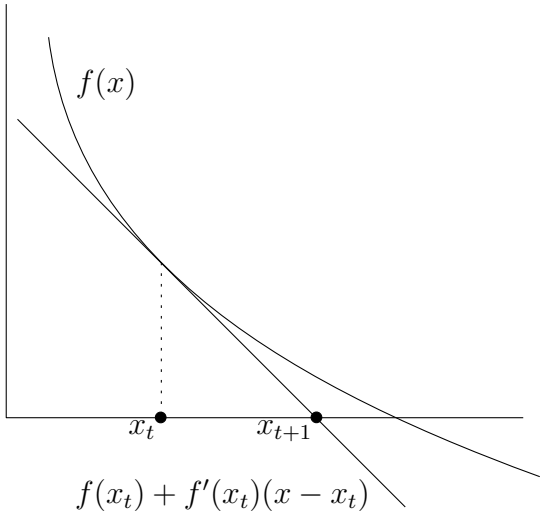


Figure 8.1: One step of Newton's method

The Newton step (8.1) obviously fails if $f'(x_t) = 0$ and may get out of control if $|f'(x_t)|$ is very small. Any theoretical analysis will have to make suitable assumptions to avoid this. But before going into this, we look at Newton's method in a benign case.

Let $f(x) = x^2 - R$, where $R \in \mathbb{R}_+$. f has two zeros, $\sqrt{R}$ and $-\sqrt{R}$. Starting for example at $x_0 = R$, we hope to converge to $\sqrt{R}$ quickly. In this case, (8.1) becomes

$$x_{t+1} = x_t - \frac{x_t^2 - R}{2x_t} = \frac{1}{2}\left(x_t + \frac{R}{x_t}\right). \tag{8.2}$$

This is in fact the *Babylonian method* to compute square roots, and here we see that it is just a special case of Newton's method.

Can we prove that we indeed quickly converge to $\sqrt{R}$? What we immediately see from (8.2) is that all iterates will be positive and hence

$$x_{t+1} = \frac{1}{2}\left(x_t + \frac{R}{x_t}\right) \geq \frac{x_t}{2}.$$

So we cannot be too fast. Suppose $R \geq 1$. In order to even get $x_t < 2\sqrt{R}$, we need at least $T \geq \log(R)/2$ steps. It turns out that the Babylonian method starts taking off only when $x_t - \sqrt{R} < 1/2$, say (Exercise 51 asks you to prove that it takes $\mathcal{O}(\log R)$ steps to get there).

To watch takeoff, let us now suppose that $x_0 - \sqrt{R} < 1/2$, so we are starting close to $\sqrt{R}$ already. We rewrite (8.2) as

$$x_{t+1} - \sqrt{R} = \frac{x_t}{2} + \frac{R}{2x_t} - \sqrt{R} = \frac{1}{2x_t}\left(x_t - \sqrt{R}\right)^2. \tag{8.3}$$

Assuming for now that $R \geq 1/4$, all iterates have value at least $\sqrt{R} \geq 1/2$, hence we get

$$x_{t+1} - \sqrt{R} \leq \left(x_t - \sqrt{R}\right)^2.$$

This means that the error goes to 0 *quadratically*, and

$$x_T - \sqrt{R} \leq \left(x_0 - \sqrt{R}\right)^{2^T} < \left(\frac{1}{2}\right)^{2^T}, \quad T \geq 0. \tag{8.4}$$

What does this tell us? In order to get $x_T - \sqrt{R} < \varepsilon$, we only need $T = \log\log(\frac{1}{\varepsilon})$ steps! Hence, it takes a while to get to roughly $\sqrt{R}$, but from there, we achieve high accuracy very fast.

Let us do a concrete example of the practical behavior (on a computer with IEEE 754 double arithmetic). If $R = 1000$, the method takes 7 steps to get $x_7 - \sqrt{1000} < 1/2$, and then 3 more steps to get x_{13} equal to $\sqrt{1000}$ up to the machine precision (53 binary digits). In this last phase, we essentially double the number of correct digits in each iteration!

8.2 Newton's method for optimization

Suppose we want to find a global minimum $x^\star$ of a differentiable convex function $f : \mathbb{R} \to \mathbb{R}$ (assuming that a global minimum exists). Lemmata 2.22 and 2.23 guarantee that we can equivalently search for a zero of the derivative f'. To do this, we can apply Newton's method if f is *twice* differentiable; the update step then becomes

$$x_{t+1} := x_t - \frac{f'(x_t)}{f''(x_t)} = x_t - f''(x_t)^{-1} f'(x_t), \quad t \geq 0. \tag{8.5}$$

There is no reason to restrict to $d = 1$. Here is Newton's method for minimizing a convex function $f : \mathbb{R}^d \to \mathbb{R}$. We choose $\mathbf{x}_0$ arbitrarily and then iterate:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla^2 f(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t), \quad t \geq 0. \tag{8.6}$$

The update vector $\nabla^2 f(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t)$ is the result of a matrix-vector multiplication: we invert the Hessian at $\mathbf{x}_t$ and multiply the result with the gradient at $\mathbf{x}_t$. As before, this fails if the Hessian is not invertible, and may get out of control if the Hessian has small norm.

We have introduced iteration (8.6) simply as a (more or less natural) generalization of (8.5), but there's more to it. If we consider (8.6) as a special case of a general update scheme

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H(\mathbf{x}_t) \nabla f(\mathbf{x}_t),$$

where $H(\mathbf{x}_t) \in \mathbb{R}^{d \times d}$ is some matrix, then we see that also gradient descent (3.1) is of this form, with $H(\mathbf{x}_t) = \gamma I$. Hence, Newton's method can also be thought of as "adaptive gradient descent" where the adaptation is w.r.t. the local geometry of the function at $\mathbf{x}_t$. Indeed, as we show next, this allows Newton's method to converge on *all* nondegenerate quadratic functions in one step, while gradient descent only does so with the right stepsize on "beautiful" quadratic functions whose sublevel sets are Euclidean balls (Exercise 30).

Lemma 8.1. *A* nondegenerate *quadratic function is a function of the form*

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top M \mathbf{x} - \mathbf{q}^\top \mathbf{x} + c,$$

where $M \in \mathbb{R}^{d \times d}$ is an invertible symmetric matrix, $\mathbf{q} \in \mathbb{R}^d, c \in R$. Let $\mathbf{x}^\star = M^{-1}\mathbf{q}$ be the unique solution of $\nabla f(\mathbf{x}) = \mathbf{0}$ (the unique global minimum if f is convex). With any starting point $\mathbf{x}_0 \in \mathbb{R}^d$, Newton's method (8.6) yields $\mathbf{x}_1 = \mathbf{x}^\star$.

Proof. We have $\nabla f(\mathbf{x}) = M\mathbf{x} - \mathbf{q}$ (this implies $\mathbf{x}^\star = M^{-1}\mathbf{q}$) and $\nabla^2 f(\mathbf{x}) = M$. Hence,

$$\mathbf{x}_0 - \nabla^2 f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0) = \mathbf{x}_0 - M^{-1}(M\mathbf{x}_0 - \mathbf{q}) = M^{-1}\mathbf{q} = \mathbf{x}^\star.$$

□

In particular, Newton's method can solve an invertible system $M\mathbf{x} = \mathbf{q}$ of linear equations in one step. But no miracle is happening here, as this step involves the inversion of the matrix $\nabla^2 f(\mathbf{x}_0) = M$.

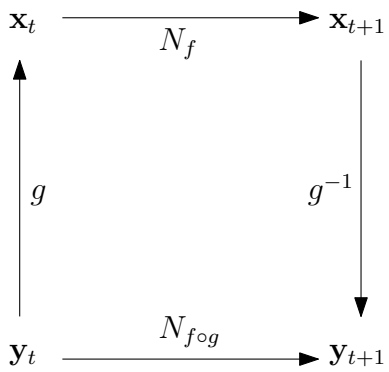
More generally, the behavior of Newton's method is affine invariant. By this, we mean that it is invariant under any invertible affine transformation, as follows:

Lemma 8.2 (Exercise 52)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be twice differentiable, $A \in \mathbb{R}^{d\times d}$ an invertible matrix, $\mathbf{b} \in \mathbb{R}^d$. Let $g : \mathbb{R}^d \to \mathbb{R}^d$ be the (bijective) affine function $g(\mathbf{y}) = A\mathbf{y} + \mathbf{b}, \mathbf{y} \in \mathbb{R}^d$. Finally, for a twice differentiable function $h : \mathbb{R}^d \to \mathbb{R}$, let $N_h : \mathbb{R}^d \to \mathbb{R}^d$ denote the Newton step for h, i.e.*

$$N_h(\mathbf{x}) := \mathbf{x} - \nabla^2 h(\mathbf{x})^{-1}\nabla h(\mathbf{x}),$$

whenever this is defined. Then we have $N_{f\circ g} = g^{-1} \circ N_f \circ g$.

This says that in order to perform a Newton step for $f \circ g$ on $\mathbf{y}_t$, we can transform $\mathbf{y}_t$ to $\mathbf{x}_t = g(\mathbf{y}_t)$, perform the Newton step for f on $\mathbf{x}$ and transform the result $\mathbf{x}_{t+1}$ back to $\mathbf{y}_{t+1} = g^{-1}(\mathbf{x}_{t+1})$. Another way of saying this is that the following diagram commutes:



Hence, while gradient descent suffers if the coordinates are at very different scales, Newton's method doesn't.

We conclude the general exposition with another interpretation of Newton's method: each step minimizes the local second-order Taylor approximation.

Lemma 8.3 (Exercise 55)**.** *Let f be convex and twice differentiable at $\mathbf{x}_t \in \mathbf{dom}(f)$, with $\nabla^2 f(\mathbf{x}_t) \succ 0$ being invertible. The vector $\mathbf{x}_{t+1}$ resulting from the Netwon step (8.6) satisfies*

$$\mathbf{x}_{t+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top(\mathbf{x}-\mathbf{x}_t) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_t)^\top \nabla^2 f(\mathbf{x}_t)(\mathbf{x}-\mathbf{x}_t).$$

8.3 Once you're close, you're there...

We will prove a result about Newton's method that may seem rather weak: under suitable conditions, and starting close to the global minimum, we will reach distance at most ε to the minimum within $\log\log(1/\varepsilon)$ steps. The weak part here is of course not the number of steps $\log\log(1/\varepsilon)$—this is much faster than anything we have seen so far—but the assumption that we are starting close to the minimum already. Under such an assumption, we say that we have a *local convergence* result.

To compensate for the above weakness to some extent, we will be able to handle non-convex functions as well. More precisely, we show that under the aforementioned suitable conditions, and starting close to a critical point, we will reach distance at most ε to the critical point within $\log\log(1/\varepsilon)$ steps. This can of course only work if the conditions ensure that we are close to only one critical point; so we have a unique critical point nearby, and Newton's method will have no choice other than to converge to it.

For convex functions, we can ask about *global convergence* results that hold for every starting point. In general, such results were unknown for Newton's method as in (8.6) until recently. Under a stability assumption on the Hessian, global convergence was shown to hold by [KSJ18]. There are some variants of Newton's method for which such results can be proved, most notably the cubic regularization variant of Nesterov and Polyak [NP06]. Weak global convergence results can be obtained by adding

a step size to (8.6) and always making only steps that decrease the function value (which may not happen under the full Newton step).

An alternative is to use gradient descent to get us sufficiently close to the global minimum, and then switch to Newton's method for the rest. In Chapter 3, we have seen that under favorable conditions, we may know when gradient descent has taken us close enough.

In practice, Newton's method is often (but not always) much faster than gradient descent in terms of the number of iterations. The price to pay is a higher iteration cost, since we need to compute (and invert) Hessians.

After this disclaimer, let us state the main result right away. We follow Vishnoi [Vis15], except that we do not require convexity.

Theorem 8.4. *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be twice differentiable with a critical point $\mathbf{x}^\star$. Suppose that there is a ball $X \subseteq \mathbf{dom}(f)$ with center $\mathbf{x}^\star$ such that the following two properties hold.*

(i) Bounded inverse Hessians: *There exists a real number $\mu > 0$ such that*

$$\|\nabla^2 f(\mathbf{x})^{-1}\| \leq \frac{1}{\mu}, \quad \forall \mathbf{x} \in X.$$

(ii) Lipschitz continuous Hessians*: There exists a real number $B \geq 0$ such that*

$$\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\| \leq B\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

In both cases, the matrix norm is the spectral norm defined in Lemma 3.6. Property (i) in particular stipulates that Hessians are invertible at all points in X.

Then, for $\mathbf{x}_t \in X$ and $\mathbf{x}_{t+1}$ resulting from the Newton step (8.6), we have

$$\|\mathbf{x}_{t+1} - \mathbf{x}^\star\| \leq \frac{B}{2\mu}\|\mathbf{x}_t - \mathbf{x}^\star\|^2.$$

As an example, let us consider a nondegenerate quadratic function f (constant Hessian $M = \nabla^2 f(\mathbf{x})$ for all $\mathbf{x}$; see Lemma 8.1). Then f has exactly one critical point $\mathbf{x}^\star$. Property (i) is satisfied with $\mu = 1/\|M^{-1}\|$ over $X = \mathbb{R}^d$; property (ii) is satisfied for $B = 0$. According to the statement of the theorem, Newton's method will thus reach $\mathbf{x}^\star$ after one step—which we already know from Lemma 8.1.

In general, there could be several critical points for which properties (i) and (ii) hold, and it may seem surprising that the theorem makes a

statement about all of them. But in fact, if $\mathbf{x}_t$ is far away from such a critical point, the statement allows $\mathbf{x}_{t+1}$ to be even further away from it; we cannot expect to make progress towards all critical points simultaneously. The theorem becomes interesting only if we are *very close* to some critical point. In this case, we will actually converge to it. In particular, this critical point is then isolated and the only one nearby, so that Newton's method cannot avoid getting there.

Corollary 8.5 (Exercise 53)**.** *With the assumptions and terminology of Theorem 8.4, and if $\mathbf{x}_0 \in X$ satisfies*

$$\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq \frac{\mu}{B},$$

then Newton's method (8.6) yields

$$\|\mathbf{x}_T - \mathbf{x}^\star\| \leq \frac{\mu}{B}\left(\frac{1}{2}\right)^{2^T-1}, \quad T \geq 0.$$

Hence, we have a bound as (8.4) for the last phase of the Babylonian method: in order to get $\|\mathbf{x}_T - \mathbf{x}^\star\| < \varepsilon$, we only need $T = \log\log(\frac{1}{\varepsilon})$ steps. But before this fast behavior kicks in, we need to be μ/B-close to $\mathbf{x}^\star$ already. The fact that $\mathbf{x}_0$ is this close to only *one* critical point necessarily follows.

An intuitive reason for a unique critical point near $\mathbf{x}_0$ (and for fast convergence to it) is that under our assumptions, the Hessians we encounter are almost constant when we are close to $\mathbf{x}^\star$. This means that locally, our function behaves almost like a nondegenerate quadratic function which has truly constant Hessians and allows Newton's method to convergence to its unique critical point in one step (Lemma 8.1).

Lemma 8.6 (Exercise 54)**.** *With the assumptions and terminology of Theorem 8.4, and if $\mathbf{x}_0 \in X$ satisfies*

$$\|\mathbf{x}_0 - \mathbf{x}^\star\| \leq \frac{\mu}{B},$$

then the Hessians in Newton's method satisfy the relative error bound

$$\frac{\|\nabla^2 f(\mathbf{x}_t) - \nabla^2 f(\mathbf{x}^\star)\|}{\|\nabla^2 f(\mathbf{x}^\star)\|} \leq \left(\frac{1}{2}\right)^{2^t-1}, \quad t \geq 0.$$

We now still owe the reader the proof of the main convergence result, Theorem 8.4:

Proof of Theorem 8.4. To simplify notation, let us abbreviate $H := \nabla^2 f$, $\mathbf{x} = \mathbf{x}_t, \mathbf{x}' = \mathbf{x}_{t+1}$. Subtracting $\mathbf{x}^\star$ from both sides of (8.6), we get

$$\begin{aligned}\mathbf{x}' - \mathbf{x}^\star &= \mathbf{x} - \mathbf{x}^\star - H(\mathbf{x})^{-1}\nabla f(\mathbf{x}) \\ &= \mathbf{x} - \mathbf{x}^\star + H(\mathbf{x})^{-1}(\nabla f(\mathbf{x}^\star) - \nabla f(\mathbf{x})) \\ &= \mathbf{x} - \mathbf{x}^\star + H(\mathbf{x})^{-1}\int_0^1 H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x}))(\mathbf{x}^\star - \mathbf{x})dt.\end{aligned}$$

The last step, which applies the fundamental theorem of calculus, needs some explanations. In fact, we have applied it to each component $h_i(t)$ of the vector-valued function $h(t) = \nabla f(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x}))$:

$$h_i(1) - h_i(0) = \int_0^1 h_i'(t), \quad i = 1, \ldots, d.$$

These d equations can be summarized as

$$\nabla f(\mathbf{x}^\star) - \nabla f(\mathbf{x}) = h(1) - h(0) = \int_0^1 h'(t),$$

where $h'(t)$ has components $h_1'(t), \ldots, h_d'(t)$, and the integral is also understood componentwise. Furthermore, as $h_i(t) = \frac{\partial f}{\partial x_i}(\mathbf{x}+t(\mathbf{x}^\star-\mathbf{x}))$, the chain rule yields $h_i'(t) = \sum_{j=1}^d \frac{\partial f}{\partial x_{ij}}(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x}))(\mathbf{x}_j^\star - \mathbf{x}_j)$. This summarizes to $h'(t) = H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x}))(\mathbf{x}^\star - \mathbf{x})$.

Also note that we are allowed to apply the fundamental theorem of calculus in the first place, since f is twice *continuously* differentiable over X (as a consequence of assuming Lipschitz continuous Hessians), so also $h'(t)$ is continuous.

After this justifying intermezzo, we further massage the expression we have obtained last. Using

$$\mathbf{x} - \mathbf{x}^\star = H(\mathbf{x})^{-1}H(\mathbf{x})(\mathbf{x} - \mathbf{x}^\star) = H(\mathbf{x})^{-1}\int_0^1 -H(\mathbf{x})(\mathbf{x}^\star - \mathbf{x})dt,$$

we can now write

$$\mathbf{x}' - \mathbf{x}^\star = H(\mathbf{x})^{-1}\int_0^1 \left(H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x})) - H(\mathbf{x})\right)(\mathbf{x}^\star - \mathbf{x})dt.$$

Taking norms, we have

$$\|\mathbf{x}' - \mathbf{x}^\star\| \leq \|H(\mathbf{x})^{-1}\| \cdot \left\| \int_0^1 \left(H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x})) - H(\mathbf{x})\right)(\mathbf{x}^\star - \mathbf{x}) dt \right\|,$$

by the properties of the spectral norm. As we also have

$$\left\| \int_0^1 \mathbf{g}(t) dt \right\| \leq \int_0^1 \|\mathbf{g}(t)\| dt$$

for any vector-valued function g (Exercise 57), we can further bound

$$\begin{aligned}
\|\mathbf{x}' - \mathbf{x}^\star\| &\leq \|H(\mathbf{x})^{-1}\| \int_0^1 \left\| \big(H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x})) - H(\mathbf{x})\big)(\mathbf{x}^\star - \mathbf{x}) \right\| dt \\
&\leq \|H(\mathbf{x})^{-1}\| \int_0^1 \left\| H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x})) - H(\mathbf{x}) \right\| \cdot \|\mathbf{x}^\star - \mathbf{x}\| dt \\
&= \|H(\mathbf{x})^{-1}\| \cdot \|\mathbf{x}^\star - \mathbf{x}\| \int_0^1 \left\| H(\mathbf{x} + t(\mathbf{x}^\star - \mathbf{x})) - H(\mathbf{x}) \right\| dt.
\end{aligned}$$

We can now use the properties (i) and (ii) (bounded inverse Hessians, Lipschitz continuous Hessians) to conclude that

$$\|\mathbf{x}' - \mathbf{x}^\star\| \leq \frac{1}{\mu} \|\mathbf{x}^\star - \mathbf{x}\| \int_0^1 B \|t(\mathbf{x}^\star - \mathbf{x})\| dt = \frac{B}{\mu} \|\mathbf{x}^\star - \mathbf{x}\|^2 \underbrace{\int_0^1 t dt}_{1/2}.$$

□

How realistic are properties (i) and (ii)? If f is twice *continuously* differentiable (meaning that the second derivative $\nabla^2 f$ is continuous), then we will always find suitable values of μ and B over a ball X with center $\mathbf{x}^\star$—provided that $\nabla^2 f(\mathbf{x}^\star) \neq 0$.

Indeed, already in the one-dimensional case, we see that under $f''(x^\star) = 0$ (vanishing second derivative at the global minimum), Newton's method will in the worst reduce the distance to $x^\star$ at most by a constant factor in each step, no matter how close to $x^\star$ we start. Exercise 56 asks you to find such an example. In such a case, we have linear convergence, but the fast quadratic convergence ($\mathcal{O}(\log\log(1/\varepsilon))$ steps cannot be proven.

One way to ensure bounded inverse Hessians is to require strong convexity over X.

Lemma 8.7 (Exercise 58). *Let $f : \mathbf{dom}(f) \to \mathbb{R}$ be twice differentiable and strongly convex with parameter μ over an open convex subset $X \subseteq \mathbf{dom}(f)$ according to Definition 3.10, meaning that*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2, \quad \forall \mathbf{x}, \mathbf{y} \in X.$$

Then $\nabla^2 f(\mathbf{x})$ is invertible and $\|\nabla^2 f(\mathbf{x})^{-1}\| \leq 1/\mu$ for all $\mathbf{x} \in X$, where $\|\cdot\|$ is the spectral norm defined in Lemma 3.6.

8.4 Exercises

Exercise 51. *Consider the Babylonian method (8.2). Prove that we get $x_T - \sqrt{R} < 1/2$ for $T = \mathcal{O}(\log R)$.*

Exercise 52. *Prove Lemma 8.2!*

Exercise 53. *Prove Corollary 8.5!*

Exercise 54. *Prove Lemma 8.6!*

Exercise 55. *Prove Lemma 8.3!*

Exercise 56. *Let $\delta > 0$ be any real number. Find an example of a convex function $f : \mathbb{R} \to \mathbb{R}$ such that (i) the unique global minimum $x^\star$ has a vanishing second derivative $f''(x^\star) = 0$, and (ii) Newton's method satisfies*

$$|x_{t+1} - x^\star| \geq (1 - \delta)|x_t - x^\star|,$$

for all $x_t \neq x^\star$.

Exercise 57. *This exercise is just meant to recall some basics around integrals. Show that for a vector-valued function $\mathbf{g} : \mathbb{R} \to \mathbb{R}^d$, the inequality*

$$\left\| \int_0^1 \mathbf{g}(t) dt \right\| \leq \int_0^1 \|\mathbf{g}(t)\| dt$$

holds, where $\|\cdot\|$ is the 2-norm (always assuming that the funtions under consideration are integrable)! You may assume (i) that integrals are linear:

$$\int_0^1 (\lambda_1 g_1(t) + \lambda_2 g_2(t)) dt = \lambda_1 \int_0^1 g_1(t) dt + \lambda_2 \int_0^1 g_2(t) dt,$$

And (ii), if $g(t) \geq 0$ for all $t \in [0, 1]$, then $\int_0^1 g(t) dt \geq 0$.

Exercise 58. *Prove Lemma 8.7! You may want to proceed in the following steps.*

(i) Prove that the function $g(\mathbf{x}) = f(\mathbf{x}) - \frac{\mu}{2}\|\mathbf{x}\|^2$ is convex over X (see also Exercise 28).

(ii) Prove that $\nabla^2 f(\mathbf{x})$ is invertible for all $\mathbf{x} \in X$.

(iii) Prove that all eigenvalues of $\nabla^2 f(\mathbf{x})^{-1}$ are positive and at most $1/\mu$.

(iv) Prove that for a symmetric matrix M, the spectral norm $\|M\|$ is the largest absolute eigenvalue.

Chapter 9

Quasi-Newton Methods

Contents

The main computational bottleneck in Newton's method (8.6) is the computation and inversion of the Hessian matrix in each step. This matrix has size $d \times d$, so it will take up to $\mathcal{O}(d^3)$ time to invert it (or to solve the system $\nabla^2 f(\mathbf{x}_t)\Delta\mathbf{x} = -\nabla f(\mathbf{x}_t)$ that gives us the next Newton step $\Delta\mathbf{x}$). Already in the 1950s, attempts were made to circumvent this costly step, the first one going back to Davidon [Dav59].

In this chapter, we will (for a change) not prove convergence results; rather, we focus on the development of Quasi-Newton methods, and how state-of-the-art methods arise from first principles. To motivate the approach, let us go back to the 1-dimensional case.

9.1 The secant method

Like Newton's method (8.1), the secant method is an iterative method for finding a zero of a univariate function. Unlike Newton's method, it does not use derivatives and hence does not require the function under consideration to be differentiable. In fact, it is (therefore) much older than Newton's method. Reversing history and starting from the Newton step

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}, \quad t \geq 0,$$

we can derive the secant method by replacing the derivative $f'(x_t)$ with its finite difference approximation

$$\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}}.$$

As we (in the differentiable case) have

$$f'(x_t) = \lim_{x \to x_t} \frac{f(x_t) - f(x)}{x_t - x},$$

we get

$$\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}} \approx f'(x_t)$$

for $|x_t - x_{t-1}|$ small. As the method proceeds, we expect consecutive iterates x_{t-1}, x_t to become closer and closer, so that the *secant step*

$$x_{t+1} := x_t - f(x_t)\frac{x_t - x_{t-1}}{f(x_t) - f(x_{t-1})}, \quad t \geq 1 \tag{9.1}$$

approximates the Newton step (*two* starting values x_0, x_1 need to be chosen here). Figure 9.1 shows what the method does: it constructs the line through the two points $(x_{t-1}, f(x_{t-1}))$ and $(x_t, f(x_t))$ on the graph of f; the next iterate x_{t+1} is where this line intersects the x-axis. Exercise 59 asks you to formally prove this.

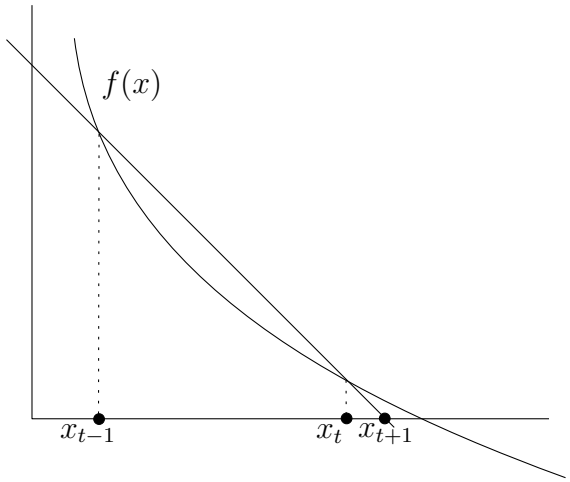


Figure 9.1: One step of the secant method

Convergence of the secant method can be analyzed, but we don't do this here. The main point for us is that we now have a *derivative-free* version of Newton's method.

When the task is to optimize a differentiable univariate function, we can apply the secant method to its derivative to obtain the secant method for optimization:

$$x_{t+1} := x_t - f'(x_t)\frac{x_t - x_{t-1}}{f'(x_t) - f'(x_{t-1})}, \quad t \geq 1. \tag{9.2}$$

This is a *second-derivative-free* version of Newton's method (8.5) for optimization. The plan is now to generalize this to higher dimensions to obtain a *Hessian-free* version of Newton's method (8.6) for optimization over $\mathbb{R}^d$.

9.2 The secant condition

Applying finite difference approximation to the second derivative of f (we're still in the 1-dimensional case), we get

$$H_t := \frac{f'(x_t) - f'(x_{t-1})}{x_t - x_{t-1}} \approx f''(x_t),$$

which we can write as

$$f'(x_t) - f'(x_{t-1}) = H_t(x_t - x_{t-1}) \approx f''(x_t)(x_t - x_{t-1}). \tag{9.3}$$

Now, while Newton's method for optimization uses the update step

$$x_{t+1} = x_t - f''(x_t)^{-1} f'(x_t), \quad t \geq 0,$$

the secant method works with the approximation $H_t \approx f''(x_t)$:

$$x_{t+1} = x_t - H_t^{-1} f'(x_t), \quad t \geq 1. \tag{9.4}$$

The fact that H_t approximates $f''(x_t)$ in the twice differentiable case was our motivation for the secant method, but in the method itself, there is no reference to f'' (which is exactly the point). All that is needed is the *secant condition* from (9.3) that defines H_t:

$$f'(x_t) - f'(x_{t-1}) = H_t(x_t - x_{t-1}). \tag{9.5}$$

This view can be generalized to higher dimensions. If $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable, (9.4) becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t), \quad t \geq 1, \tag{9.6}$$

where $H_t \in \mathbb{R}^{d \times d}$ is now supposed to be a symmetric matrix satisfying the d-dimensional secant condition

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}). \tag{9.7}$$

9.3 Quasi-Newton methods

If f is twice differentiable, the secant condition (9.7) along with the first-order Taylor approximation of $\nabla f(\mathbf{x})$ yields the d-dimensional analog of (9.3):

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}) \approx \nabla^2 f(\mathbf{x}_t)(\mathbf{x}_t - \mathbf{x}_{t-1}),$$

We might therefore hope that $H_t \approx \nabla^2 f(\mathbf{x}_t)$, and this would mean that (9.6) approximates Newton's method. Therefore, whenever we use (9.6) with a *symmetric* matrix satisfying the secant condition (9.7), we say that we have a *Quasi-Newton method*.

In the 1-dimensional case, there is only one Quasi-Newton method—the secant method (9.1). Indeed, equation (9.5) uniquely defines the number H_t in each step.

But in the d-dimensional case, the matrix H_t in the secant condition is underdetermined, starting from $d = 2$: Taking the symmetry requirement into account, (9.7) is a system of d equations in $d(d+1)/2$ unknowns, so if it is satisfiable at all, there are many solutions H_t. This raises the question of which one to choose, and how to do so efficiently; after all, we want to get some savings over Newton's method.

Newton's method is a Quasi-Newton method if and only if f is a non-degenerate quadratic function (Exercise 60). Hence, Quasi-Newton methods do not generalize Newton's method but form a family of related algorithms.

The first Quasi-Newton method was developed by William C. Davidon in 1956; he desperately needed iterations that were faster than those of Newton's method in order obtain results in the short time spans between expected failures of the room-sized computer that he used to run his computations on.

But the paper he wrote about his new method got rejected for lacking a convergence analysis, and for allegedly dubious notation. It became a very influential Technical Report in 1959 [Dav59] and was finally officially published in 1991, with a foreword giving the historical context [Dav91]. Ironically, Quasi-Newton methods are today the methods of choice in a number of relevant machine learning applications.

9.4 Greenstadt's approach

For efficieny reasons (we want to avoid matrix inversions), Quasi-Newton methods typically directly deal with the inverse matrices H_t^{-1}. Suppose that we have the iterates $\mathbf{x}_{t-1}, \mathbf{x}_t$ as well as the matrix H_{t-1}^{-1}; now we want to compute a matrix H_t^{-1} to perform the next Quasi-Newton step (9.6). How should we choose H_t^{-1}?

We draw some intuition from (the analysis of) Newton's method. Recall that we have shown $\nabla^2 f(\mathbf{x}_t)$ to fluctuate only very little in the region of extremely fast convergence (Lemma 8.6); in fact, Newton's method is optimal (one step!) when $\nabla^2 f(\mathbf{x}_t)$ is actually constant— this is the case of a quadratic function (Lemma 8.1). Hence, in a Quasi-Newton method, it also makes sense to have that $H_t \approx H_{t-1}$, or $H_t^{-1} \approx H_{t-1}^{-1}$.

Greenstadt's approach from 1970 [Gre70] is to update H_{t-1}^{-1} by an "error matrix" E_t to obtain

$$H_t^{-1} = H_{t-1}^{-1} + E_t.$$

Moreover, the errors should be as small as possible, subject to the constraints that H_t^{-1} is symmetric and satisfies the secant condition (9.7). A simple measure of error introduced by an update matrix E is its squared *Frobenius norm*

$$\|E\|_F^2 := \sum_{i=1}^{d} \sum_{j=1}^{d} e_{ij}^2.$$

Since Greenstadt considered the resulting Quasi-Newton method as "too specialized", he searched for a compromise between variability in the method and simplicity of the resulting formulas. This led him to minimize the error term

$$\|AEA^\top\|_F^2,$$

where $A \in \mathbb{R}^{d \times d}$ is some fixed invertible transformation matrix. If $A = I$, we recover the squared Frobenius norm of E.

Let us now fix t and simplify notation by setting

$$\begin{aligned} H &:= H_{t-1}^{-1}, \\ H' &:= H_t^{-1}, \\ E &:= E_t, \\ \boldsymbol{\sigma} &:= \mathbf{x}_t - \mathbf{x}_{t-1}, \\ \mathbf{y} &= \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}), \\ \mathbf{r} &= \boldsymbol{\sigma} - H\mathbf{y}. \end{aligned}$$

The update formula then is

$$H' = H + E, \tag{9.8}$$

and the secant condition $\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1})$ becomes

$$H'\mathbf{y} = \boldsymbol{\sigma} \quad (\Leftrightarrow E\mathbf{y} = \mathbf{r}). \tag{9.9}$$

Greenstadt's approach can now be distilled into the following convex constrained minimization problem in the d^2 variables E_{ij}:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|AEA^\top\|_F^2 \\ \text{subject to} & E\mathbf{y} = \mathbf{r} \\ & E^\top - E = 0 \end{array} \tag{9.10}$$

9.4.1 The method of Lagrange multipliers

Minimization subject to equality constraints can be done via the method of *Lagrange multipliers*. Here we need it only for the case of *linear* equality constraints in which case the method assumes a very simple form.

Theorem 9.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, $C \in \mathbb{R}^{m\times d}$ for some $m \in \mathbb{N}$, $\mathbf{e} \in \mathbb{R}^m$, $\mathbf{x}^\star \in \mathbb{R}^d$ such that $C\mathbf{x}^\star = \mathbf{e}$. Then the following two statements are equivalent.*

(i) $\mathbf{x}^\star = \operatorname{argmin}\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^d, C\mathbf{x} = \mathbf{e}\}$

(ii) There exists a vector $\boldsymbol{\lambda} \in \mathbb{R}^m$ such that

$$\nabla f(\mathbf{x}^\star)^\top = \boldsymbol{\lambda}^\top C.$$

The entries of $\boldsymbol{\lambda}$ are known as the Lagrange multipliers.

This is a consequence of earlier material. By Theorem 2.48, a Slater point implies strong Lagrange duality. If, as in (i), there are only affine equality constraints, the Slater point condition is void, and we obtain strong Lagrange duality "for free". In this case, the equivalence of (i) and (ii) follows from the Karush-Kuhn-Tucker necessary and sufficient conditions (Theorems 2.52 and 2.53).

For completeness we reprove Theorem 9.1 here, via elementary arguments.

Proof. The easy direction is (ii)⇒(i): if $\boldsymbol{\lambda}$ as specified exists and $\mathbf{x} \in \mathbb{R}^d$ satisfies $C\mathbf{x} = \mathbf{e}$, we get

$$\nabla f(\mathbf{x}^\star)^\top(\mathbf{x} - \mathbf{x}^\star) = \boldsymbol{\lambda}^\top C(\mathbf{x} - \mathbf{x}^\star) = \boldsymbol{\lambda}^\top(\mathbf{e} - \mathbf{e}) = 0.$$

Hence, $\mathbf{x}^\star$ is a minimizer of f over $\{\mathbf{x} \in \mathbb{R}^d : C\mathbf{x} = \mathbf{e}\}$ by the optimality condition of Lemma 2.28.

The other direction is Exercise 61. □

9.4.2 Application to Greenstadt's Update

In order to apply this method to (9.10), we need to compute the gradient of $f(E) = \frac{1}{2}\|AEA^\top\|_F^2$. Formally, this is a d^2-dimensional vector, but it is customary and more practical to write it as a matrix again,

$$\nabla f(E) = \left(\frac{\partial f(E)}{\partial E_{ij}}\right)_{1\le i,j\le d}.$$

Fact 9.2 (Exercise 62)**.** *Let $A, B \in \mathbb{R}^{d\times d}$ two matrices. With $f : \mathbb{R}^{d\times d} \to \mathbb{R}$, $f(E) := \frac{1}{2}\|AEB\|_F^2$, we have*

$$\nabla f(E) = A^\top AEBB^\top.$$

The second step is to write the system of equations $E\mathbf{y} = \mathbf{r}, E^\top - E = 0$ in Greenstadt's convex program (9.10) in matrix form $C\mathbf{x} = \mathbf{e}$ so that we can apply the method of Lagrange multipliers according to Theorem 9.1.

As there are $d + d^2$ equations in d^2 variables, it is best to think of the rows of C as being indexed with elements $i \in [d] := \{1, \dots, d\}$ for the first d equations $E\mathbf{y} = \mathbf{r}$, and pairs $(i, j) \in [d] \times [d]$ for the last d^2 symmetry constraints (more than half of which are redundant but we don't care). Columns of C are indexed with pairs (i, j) as well.

Let us denote by $\boldsymbol{\lambda} \in \mathbb{R}^d$ the Lagrange multipliers for the first d equations and $\Gamma \in \mathbb{R}^{d\times d}$ the ones for the last d^2 ones.

In column (i, j) of C corresponding to variable E_{ij}, we have entry y_j in row i as well as entries 1 (row (j, i)) and -1 (row (i, j)). Taking the inner product with the Lagrange multipliers, this column therefore yields

$$\lambda_i y_j + \Gamma_{ji} - \Gamma_{ij}.$$

After aggregating these entries into a $d \times d$ matrix, Theorem 9.1 tells us that we should aim for equality with $\nabla f(E)$ as derived in Fact 9.2. We have proved the following intermediate result.

Lemma 9.3. *An update matrix $E^\star$ satisfying the constraints $E\mathbf{y} = \mathbf{r}$ (secant condition in the next step) and $E^\top - E = 0$ (symmetry) is a minimizer of the error function $f(E) := \frac{1}{2}\|AEA^\top\|_F^2$ subject to the aforementioned constraints if and only if there exists a vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ and a matrix $\Gamma \in \mathbb{R}^{d\times d}$ such that*

$$WE^\star W = \boldsymbol{\lambda}\mathbf{y}^\top + \Gamma^\top - \Gamma, \tag{9.11}$$

where $W := A^\top A$ (a symmetric and positive definite matrix).

Note that $\boldsymbol{\lambda}\mathbf{y}^\top$ is the *outer product* of a column and a row vector and hence a matrix. As we assume A to be invertible, the quadratic function $f(E)$ is easily seen to be strongly convex and as a consequence has a unique minimizer $E^\star$ subject to the set of linear equations in (9.10) (see Lemma 3.12 which also applies if we minimize over a closed set). Hence, we know that the minimizer $E^\star$ and corresponding Lagrange multipiers $\boldsymbol{\lambda}, \Gamma$ exist.

9.4.3 The Greenstadt family

We need to solve the system of equations

$$\begin{aligned} E\mathbf{y} &= \mathbf{r}, && (9.12)\\ E^\top - E &= 0, && (9.13)\\ WEW &= \boldsymbol{\lambda}\mathbf{y}^\top + \Gamma^\top - \Gamma. && (9.14) \end{aligned}$$

This system is linear in $E, \boldsymbol{\lambda}, \Gamma$, hence easy to solve computationally. However, we want a formula for the unique solution $E^\star$ in terms of the parameters $W, \mathbf{y}, \boldsymbol{\sigma} = \mathbf{r} + H\mathbf{y}$. In the following derivation, we closely follow Greenstadt [Gre70, pages 4–5].

With $M := W^{-1}$ (which exists since $W = A^\top A$ is positive definite), (9.14) can be rewritten as

$$E = M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \Gamma^\top - \Gamma\right)M. \tag{9.15}$$

Transposing this system (using that M is symmetric) yields

$$E^\top = M\left(\mathbf{y}\boldsymbol{\lambda}^\top + \Gamma - \Gamma^\top\right)M.$$

By symmetry (9.13), we can subtract the latter two equations to obtain

$$M\left(\boldsymbol{\lambda}\mathbf{y}^\top - \mathbf{y}\boldsymbol{\lambda}^\top + 2\Gamma^\top - 2\Gamma\right)M = 0.$$

As M is invertible, this is equivalent to

$$\Gamma^\top - \Gamma = \frac{1}{2}\left(\mathbf{y}\boldsymbol{\lambda}^\top - \boldsymbol{\lambda}\mathbf{y}^\top\right),$$

so we can eliminate Γ by substituting back into (9.15):

$$E = M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \frac{1}{2}\left(\mathbf{y}\boldsymbol{\lambda}^\top - \boldsymbol{\lambda}\mathbf{y}^\top\right)\right)M = \frac{1}{2}M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right)M. \tag{9.16}$$

To also eliminate $\boldsymbol{\lambda}$, we now use (9.12)—the secant condition in the next step—to get

$$E\mathbf{y} = \frac{1}{2}M\left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right)M\mathbf{y} = \mathbf{r}.$$

Premultiplying with $2M^{-1}$ gives

$$2M^{-1}\mathbf{r} = \left(\boldsymbol{\lambda}\mathbf{y}^\top + \mathbf{y}\boldsymbol{\lambda}^\top\right)M\mathbf{y} = \boldsymbol{\lambda}\mathbf{y}^\top M\mathbf{y} + \mathbf{y}\boldsymbol{\lambda}^\top M\mathbf{y}.$$

Hence,

$$\boldsymbol{\lambda} = \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2M^{-1}\mathbf{r} - \mathbf{y}\boldsymbol{\lambda}^\top M\mathbf{y}\right). \tag{9.17}$$

To get rid of $\boldsymbol{\lambda}$ on the right hand side, we premultiply this with $\mathbf{y}^\top M$ to obtain

$$\underbrace{\mathbf{y}^\top M\boldsymbol{\lambda}}_{z} = \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2\mathbf{y}^\top\mathbf{r} - (\mathbf{y}^\top M\mathbf{y})(\underbrace{\boldsymbol{\lambda}^\top M\mathbf{y}}_{z})\right) = \frac{2\mathbf{y}^\top\mathbf{r}}{\mathbf{y}^\top M\mathbf{y}} - \underbrace{\boldsymbol{\lambda}^\top M\mathbf{y}}_{z}$$

It follows that

$$z = \boldsymbol{\lambda}^\top M\mathbf{y} = \frac{\mathbf{y}^\top\mathbf{r}}{\mathbf{y}^\top M\mathbf{y}}.$$

This in turn can be substituted into the right-hand side of (9.17) to remove $\boldsymbol{\lambda}$ there, and we get

$$\boldsymbol{\lambda} = \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2M^{-1}\mathbf{r} - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}\mathbf{y}\right).$$

Consequently,

$$\begin{aligned}
\boldsymbol{\lambda}\mathbf{y}^\top &= \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2M^{-1}\mathbf{r}\mathbf{y}^\top - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}\mathbf{y}\mathbf{y}^\top\right),\\
\mathbf{y}\boldsymbol{\lambda}^\top &= \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2\mathbf{y}\mathbf{r}^\top M^{-1} - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}\mathbf{y}\mathbf{y}^\top\right).
\end{aligned}$$

This gives us an explicit formula for E, by substituting the previous expressions back into (9.16). For this, we compute

$$\begin{aligned}
M\boldsymbol{\lambda}\mathbf{y}^\top M &= \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2\mathbf{r}\mathbf{y}^\top M - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}M\mathbf{y}\mathbf{y}^\top M\right),\\
M\mathbf{y}\boldsymbol{\lambda}^\top M &= \frac{1}{\mathbf{y}^\top M\mathbf{y}}\left(2M\mathbf{y}\mathbf{r}^\top - \frac{(\mathbf{y}^\top\mathbf{r})}{\mathbf{y}^\top M\mathbf{y}}M\mathbf{y}\mathbf{y}^\top M\right),
\end{aligned}$$

and consequently,

$$E = \frac{1}{2} M \left(\boldsymbol{\lambda} \mathbf{y}^\top + \mathbf{y} \boldsymbol{\lambda}^\top \right) M = \frac{1}{\mathbf{y}^\top M \mathbf{y}} \left(\mathbf{r}\mathbf{y}^\top M + M \mathbf{y} \mathbf{r}^\top - \frac{(\mathbf{y}^\top \mathbf{r})}{\mathbf{y}^\top M \mathbf{y}} M \mathbf{y}\mathbf{y}^\top M \right). \tag{9.18}$$

Finally, we use $\mathbf{r} = \boldsymbol{\sigma} - H\mathbf{y}$ to obtain the update matrix $E^\star$ in terms of the original parameters $H = H_{t-1}^{-1}$ (previous approximation of the inverse Hessian that we now want to update to $H_t^{-1} = H' = H + E^\star$), $\boldsymbol{\sigma} = \mathbf{x}_t - \mathbf{x}_{t-1}$ (previous Quasi-Newton step) and $\mathbf{y} = \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})$ (previous change in gradients). This gives us the Greenstadt family of Quasi-Newton methods.

Definition 9.4. *Let $M \in \mathbb{R}^{d \times d}$ be a symmetric and invertible matrix. Consider the Quasi-Newton method*

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H_t^{-1} \nabla f(\mathbf{x}_t), \quad t \geq 1,$$

where $H_0 = I$ (or some other positive definite matrix), and $H_t^{-1} = H_{t-1}^{-1} + E_t$ is chosen for all $t \geq 1$ in such a way that H_t^{-1} is symmetric and satisfies the secant condition

$$\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}) = H_t(\mathbf{x}_t - \mathbf{x}_{t-1}).$$

For any fixed t, set

$$\begin{aligned} H &:= H_{t-1}^{-1}, \\ H' &:= H_t^{-1}, \\ \boldsymbol{\sigma} &:= \mathbf{x}_t - \mathbf{x}_{t-1}, \\ \mathbf{y} &:= \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1}), \end{aligned}$$

and define

$$\begin{aligned} E^\star = \frac{1}{\mathbf{y}^\top M \mathbf{y}} \Big(\boldsymbol{\sigma}\mathbf{y}^\top M &+ M\mathbf{y}\boldsymbol{\sigma}^\top - H\mathbf{y}\mathbf{y}^\top M - M\mathbf{y}\mathbf{y}^\top H \\ &- \frac{1}{\mathbf{y}^\top M \mathbf{y}} (\mathbf{y}^\top \boldsymbol{\sigma} - \mathbf{y}^\top H \mathbf{y}) M \mathbf{y}\mathbf{y}^\top M \Big). \end{aligned} \tag{9.19}$$

If the update matrix $E_t = E^\star$ is used, the method is called the Greenstadt method *with parameter M.*

9.4.4 The BFGS method

In his paper, Greenstadt suggested two obvious choices for the matrix M In Definition 9.4, namely $M = H$ (the previous approximation of the inverse Hessian) and $M = I$. In the next paper of the same issue of the same journal, Goldfarb suggested to use the matrix $M = H'$, the *next* approximation of the inverse Hessian. Even though we don't yet have it, we can use it in the formula (9.19) since we know that H' will by design satisfy the secant condition $H'\mathbf{y} = \boldsymbol{\sigma}$. And as M always appears next to $\mathbf{y}$ in (9.19), $M\mathbf{y} = H'\mathbf{y} = \boldsymbol{\sigma}$, so H' disappears from the formula!

Definition 9.5. *The* BFGS method *is the Greenstadt method with parameter $M = H' = H_t^{-1}$ in step t, in which case the update matrix $E^\star$ assumes the form*

$$\begin{aligned} E^\star &= \frac{1}{\mathbf{y}^\top\boldsymbol{\sigma}}\Big(2\boldsymbol{\sigma}\boldsymbol{\sigma}^\top - H\mathbf{y}\boldsymbol{\sigma}^\top - \boldsymbol{\sigma}\mathbf{y}^\top H - \frac{1}{\boldsymbol{\sigma}^\top\mathbf{y}}(\mathbf{y}^\top\boldsymbol{\sigma} - \mathbf{y}^\top H\mathbf{y})\boldsymbol{\sigma}\boldsymbol{\sigma}^\top\Big) \\ &= \frac{1}{\mathbf{y}^\top\boldsymbol{\sigma}}\Big(-H\mathbf{y}\boldsymbol{\sigma}^\top - \boldsymbol{\sigma}\mathbf{y}^\top H + \Big(1 + \frac{\mathbf{y}^\top H\mathbf{y}}{\mathbf{y}^\top\boldsymbol{\sigma}}\Big)\boldsymbol{\sigma}\boldsymbol{\sigma}^\top\Big)., \end{aligned} \tag{9.20}$$

where $H = H_{t-1}^{-1}, \boldsymbol{\sigma} = \mathbf{x}_t - \mathbf{x}_{t-1}, \mathbf{y} = \nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})$.

We leave it as Exercise 63 (i) to prove that the denominator $\mathbf{y}^\top\boldsymbol{\sigma}$ appearing twice in the formula is positive, unless the function f is flat between the iterates $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$. And under $\mathbf{y}^\top\boldsymbol{\sigma} > 0$, the BFGS method has another nice property: if the previous matrix H is positive definite, then also the next matrix H' is positive definite; see Exercise 63 (ii). In this sense, the matrices H_t^{-1} behave like proper inverse Hessians.

The method is named after Broyden, Fletcher, Goldfarb and Shanno who all came up with it independently around 1970. Greenstadt's name is mostly forgotten.

Let's take a step back and see what we have achieved. Recall that our starting point was that Newton's method needs to compute and invert Hessian matrices in each iteration and therefore has in practice a cost of $\mathcal{O}(d^3)$ per iteration. Did we improve over this?

First of all, any method in Greenstadt's family avoids the computation of Hessian matrices altogether. Only gradients are needed. In the BFGS method in particular, the cost per iteration drops to $\mathcal{O}(d^2)$. Indeed, the computation of the update matrix $E^\star$ in Definition 9.5 reduces to matrix-vector multiplications and outer-product computations, all of which can be done in $\mathcal{O}(d^2)$ time.

Newton and Quasi-Newton methods are often performed with *scaled steps*. This means that the iteration becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t H_t^{-1} \nabla f(\mathbf{x}_t), \quad t \geq 1, \tag{9.21}$$

for some $\alpha_t \in \mathbb{R}_+$. This parameter can for example be chosen such that $f(\mathbf{x}_{t+1})$ is minimized (line search). Another approach is *backtracking line search* where we start with $\alpha_t = 1$, and as long as this does not lead to sufficient progress, we halve α_t. Line search ensures that the matrices H_t^{-1} in the BFGS method remain positive definite [Gol70].

As the Greenstadt update method just depends on the step $\boldsymbol{\sigma} = \mathbf{x}_t - \mathbf{x}_{t-1}$ but not on how it was obtained, the update works in exactly the same way as before even if scaled steps are being used.

9.4.5 The L-BFGS method

In high dimensions d, even an iteration cost of $O(d^2)$ as in the BFGS method may be prohibitive. In fact, already at the end of the 1970s, the first *limited memory* (and limited time) variants of the method have been proposed. Here we essentially follow Nocedal [Noc80]. The idea is to use only information from the previous m iterations, for some small value of m, and "forget" anything older. In order to describe the resulting L-BFGS method, we first rewrite the BFGS update formula in product form.

Observation 9.6. *With $E^\star$ as in Definition 9.5 and $H' = H + E^\star$, we have*

$$H' = \left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) H \left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) + \frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}. \tag{9.22}$$

To verify this, simply expand the product in the right-hand side and compare with (9.20).

We further observe that we do not need the actual matrix $H' = H_t^{-1}$ to perform the next Quasi-Newton step (9.6), but only the vector $H'\nabla f(\mathbf{x}_t)$. Here is the crucial insight.

Lemma 9.7. *Let H, H' as in Observation 9.6, i.e.*

$$H' = \left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) H \left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) + \frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}.$$

Let $\mathbf{g}' \in \mathbb{R}^d$. Suppose that we have an oracle to compute $\mathbf{s} = H\mathbf{g}$ for any vector $\mathbf{g}$. Then $\mathbf{s}' = H'\mathbf{g}'$ can be computed with one oracle call and $O(d)$ additional arithmetic operations, assuming that $\boldsymbol{\sigma}$ and $\mathbf{y}$ are known.

Proof. From (9.22), we conclude that

$$H'\mathbf{g}' = \underbrace{\underbrace{\left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right) \underbrace{H \underbrace{\left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right)\mathbf{g}'}_{\mathbf{g}}}_{\mathbf{s}}}_{\mathbf{w}} + \underbrace{\frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\mathbf{g}'}_{\mathbf{h}}}_{\mathbf{z}}.$$

We compute the vectors $\mathbf{h}, \mathbf{g}, \mathbf{s}, \mathbf{w}, \mathbf{z}$ in turn. We have

$$\mathbf{h} = \frac{\boldsymbol{\sigma}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\mathbf{g}' = \boldsymbol{\sigma}\frac{\boldsymbol{\sigma}^\top\mathbf{g}'}{\mathbf{y}^\top\boldsymbol{\sigma}},$$

so $\mathbf{h}$ can be computed with two inner products, a real division, and a multiplication of $\boldsymbol{\sigma}$ with a scalar. For $\mathbf{g}$, we obtain

$$\mathbf{g} = \left(I - \frac{\mathbf{y}\boldsymbol{\sigma}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right)\mathbf{g}' = \mathbf{g}' - \mathbf{y}\frac{\boldsymbol{\sigma}^\top\mathbf{g}'}{\mathbf{y}^\top\boldsymbol{\sigma}}.$$

which is a multiplication of $\mathbf{y}$ with a scalar that we already know, followed by a vector addition. To get $\mathbf{s} = H\mathbf{g}$, we call the oracle. For $\mathbf{w}$, we similarly have

$$\mathbf{w} = \left(I - \frac{\boldsymbol{\sigma}\mathbf{y}^\top}{\mathbf{y}^\top\boldsymbol{\sigma}}\right)\mathbf{s} = \mathbf{s} - \boldsymbol{\sigma}\frac{\mathbf{y}^\top\mathbf{s}}{\mathbf{y}^\top\boldsymbol{\sigma}},$$

which is one inner product (the other one we already know), a real divison, a multiplication of $\boldsymbol{\sigma}$ with a scalar, and a vector addition. Finally,

$$H'\mathbf{g}' = \mathbf{z} = \mathbf{w} + \mathbf{h}$$

is a vector addition. In total, we needed three inner product computations, three scalar multiplications, three vector additions, two real divisions, and one oracle call. □

How do we implement the oracle? We simply apply the previous Lemma recursively. Let

$$\begin{aligned} \boldsymbol{\sigma}_k &= \mathbf{x}_k - \mathbf{x}_{k-1}, \\ \mathbf{y}_k &= \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1}) \end{aligned}$$

be the values of $\boldsymbol{\sigma}$ and $\mathbf{y}$ in iteration $k \leq t$. When we perform the Quasi-Newton step $\mathbf{x}_{t+1} = \mathbf{x}_t - H_t^{-1}\nabla f(\mathbf{x}_t)$ in iteration $t \geq 1$, we have already computed these vectors for $k = 1, \ldots, t$. Using Lemma 9.7, we could therefore call the recursive procedure in Algorithm 1 with $k = t, \mathbf{g}' = \nabla f(\mathbf{x}_t)$ to compute the required vector $H_t^{-1}\nabla f(\mathbf{x}_t)$ in iteration t. To maintain the immediate connection to Lemma 9.7, we refrain from introducing extra variables for values that occur several times; but in an actual implementation, this would be done, of course.

Algorithm 1 Recursive view of the BFGS method. To compute $H_t^{-1}\nabla f(\mathbf{x}_t)$, call the function with arguments $(t, \nabla f(\mathbf{x}_t))$; values $\boldsymbol{\sigma}_k, \mathbf{y}_k$ from iterations $1, \ldots, t$ are assumed to be available.

function BFGS-STEP($k, \mathbf{g}'$) ▷ returns $H_k^{-1}\mathbf{g}'$
  if $k = 0$ **then**
    return $H_0^{-1}\mathbf{g}'$
  else ▷ apply Lemma 9.7
    $\mathbf{h} = \boldsymbol{\sigma} \dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{g} = \mathbf{g}' - \mathbf{y} \dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{s} =$ BFGS-STEP($k-1, \mathbf{g}$)
    $\mathbf{w} = \mathbf{s} - \boldsymbol{\sigma}_k \dfrac{\mathbf{y}_k^\top \mathbf{s}}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{z} = \mathbf{w} + \mathbf{h}$
    return $\mathbf{z}$
  end if
end function

By Lemma 9.7, the runtime of BFGS-STEP$(t, \nabla f(\mathbf{x}_t))$ is $O(td)$. For $t > d$, this is slower (and needs more memory) than the standard BFGS step according to Definition 9.5 which always takes $O(d^2)$ time.

The benefit of the recursive variant is that it can easily be adapted to a step that is *faster* (and needs *less* memory) than the standard BFGS step. The idea is to let the recursion bottom out after a fixed number m of recursive calls (in practice, values of $m \leq 10$ are not uncommon). The step then has runtime $O(md)$ which is a substantial saving over the standard step if m is much smaller than d.

The only remaining question is what we return when the recursion now bottoms out prematurely at $k = t - m$. As we don't know the matrix H_{t-m}^{-1}, we cannot return $H_{t-m}^{-1}\mathbf{g}'$ (which would be the correct output in this case). Instead, we pretend that we have started the whole method just now and use our initial matrix H_0 instead of H_{t-m}.[1] The resulting algorithm is depicted in Algorithm 2.

Algorithm 2 The L-BFGS method. To compute $H_t^{-1}\nabla f(\mathbf{x}_t)$ based on the previous m iterations, call the function with arguments $(t, m, \nabla f(\mathbf{x}_t))$; values $\boldsymbol{\sigma}_k, \mathbf{y}_k$ from iterations $t - m + 1, \ldots, t$ are assumed to be available.

---

function L-BFGS-STEP($k, \ell, \mathbf{g}'$) ▷ $\ell \leq k$; returns $\mathbf{s}' \approx H_k^{-1}\mathbf{g}'$
  if $\ell = 0$ **then**
    return $H_0^{-1}\mathbf{g}'$
  else ▷ apply Lemma 9.7
    $\mathbf{h} = \boldsymbol{\sigma}\dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{g} = \mathbf{g}' - \mathbf{y}\dfrac{\boldsymbol{\sigma}_k^\top \mathbf{g}'}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{s} =$ L-BFGS-STEP$(k-1, \ell-1, \mathbf{g})$
    $\mathbf{w} = \mathbf{s} - \boldsymbol{\sigma}_k\dfrac{\mathbf{y}_k^\top \mathbf{s}}{\mathbf{y}_k^\top \boldsymbol{\sigma}_k}$
    $\mathbf{z} = \mathbf{w} + \mathbf{h}$
    return $\mathbf{z}$
  end if
end function

Note that the L-BFGS method is still a Quasi-Newton method as long as $m \geq 1$: if we go through at least one update step of the form $H' = H + E$,

[1] In practice, we can do better: as we already have some information from previous steps, we can use this information to construct a more tuned H_0. We don't go into this here.

the matrix H' will satisfy the secant condition by design, irrespective of H.

9.5 Exercises

Exercise 59. *Consider a step of the secant method:*

$$x_{t+1} = x_t - f(x_t)\frac{x_t - x_{t-1}}{f(x_t) - f(x_{t-1})}, \quad t \geq 1.$$

Assuming that $x_t \neq x_{t-1}$ and $f(x_t) \neq f(x_{t-1})$, prove that the line through the two points $(x_{t-1}, f(x_{t-1}))$ and $(x_t, f(x_t))$ intersects the x-axis at the point $x = x_{t+1}$.

Exercise 60. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a twice differentiable function with nonzero Hessians everywhere. Prove that the following two statements are equivalent.*

(i) f is a nondegenerate quadratic function, meaning that

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top M\mathbf{x} - \mathbf{q}^\top\mathbf{x} + c,$$

where $M \in \mathbb{R}^{d\times d}$ is an invertible symmetric matrix, $\mathbf{q} \in \mathbb{R}^d, c \in R$ (see also Lemma 8.1).

(ii) Applied to f, Newton's update step

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla^2 f(\mathbf{x}_t)^{-1}\nabla f(\mathbf{x}_t), \quad t \geq 1$$

defines a Quasi-Newton method for all $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^d$.

Exercise 61. *Prove the direction (i)⇒(ii) of Theorem 9.1! You may want to do proceed in the following steps.*

1. *Prove the* Poor Man's Farkas Lemma*: a system of n linear equations $A\mathbf{x} = \mathbf{b}$ in d variables has a solution if and only if for all $\boldsymbol{\lambda} \in \mathbb{R}^n$, $\boldsymbol{\lambda}^\top A = \mathbf{0}^\top$ implies $\boldsymbol{\lambda}^\top\mathbf{b} = 0$. (You may use the fact that the row rank of a matrix equals its column rank.)*
2. *Argue that $\mathbf{x}^\star = \operatorname{argmin}\{\nabla f(\mathbf{x}^\star)^\top\mathbf{x} : \mathbf{x} \in \mathbb{R}^d, C\mathbf{x} = \mathbf{e}\}$.*
3. *Apply the Poor Man's Farkas Lemma.*

Exercise 62. *Prove Fact 9.2!*

Exercise 63. *Consider the BFGS method (Definition 9.5).*

(i) Prove that $\mathbf{y}^\top \sigma > 0$*, unless* $\mathbf{x}_t = \mathbf{x}_{t-1}$*, or* $f(\lambda \mathbf{x}_t + (1-\lambda)\mathbf{x}_{t-1}) = \lambda f(\mathbf{x}_t) + (1-\lambda) f(\mathbf{x}_{t-1})$ *for all* $\lambda \in (0,1)$*.*

(ii) Prove that if H *is positive definite and* $\mathbf{y}^\top \sigma > 0$*, then also* H' *is positive definite. You may want to use the product form of the BFGS update as developed in Observation 9.6.*

Bibliography

[ACD+22] Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, pages 1–50, 2022.

[ACGH18] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *CoRR*, abs/1810.02281, 2018.

[AE08] Herbert Amann and Joachim Escher. *Analysis II*. Birkhäuser, 2008.

[AWBR09] Alekh Agarwal, Martin J Wainwright, Peter Bartlett, and Pradeep Ravikumar. Information-theoretic lower bounds on the oracle complexity of convex optimization. *Advances in Neural Information Processing Systems*, 22, 2009.

[AZ99] Nina Amenta and Günter M. Ziegler. Deformed products and maximal shadows of polytopes. In B. Chazelle, J.E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 57–90. American Mathematical Society, 1999.

[BB07] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20, 2007.

[BCN18] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[BEHW89] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[Ber05] Dimitri P. Bertsekas. Lecture slides on convex analysis and optimization, 2005. `http://athenasc.com/Convex_Slides.pdf`.

[BG17] Nikhil Bansal and Anupam Gupta. Potential-function proofs for first-order methods. *CoRR*, abs/1712.04581, 2017.

[Bor87] Karl Heinz Borgwardt. *The Simplex Method*. Algorithms and Combinatorics. Springer, 1987.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. `https://web.stanford.edu/~boyd/cvxbook/`.

[Cla10] Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms*, 6(4), sep 2010.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ron L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, Mass, 3rd ed. edition, 2009.

[CLSH19] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *ICLR*, 2019.

[CO19] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in Neural Information Processing Systems*, 32:15236–15245, 2019.

[Dan16] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 2016.

[Dav59] William C. Davidon. Variable metric method for minimization. Technical Report ANL-5990, AEC Research and Development, 1959.

[Dav91] William C. Davidon. Variable metric method for minimization. *SIAM J. Optimization*, 1(1):1–17, 1991.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[DHS11a] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[DHS11b] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[Die69] J. Dieudonneé. *Foundations of Modern Analysis*. Academic Press, 1969.

[Don17] David Donoho. Fifty years of data science. *Journal of Computational and Graphical Statistics*, 24(6):745–766, 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 *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279, 2008.

[FLLZ18] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.

[FM91] M. Furi and M. Martelli. On the mean value theorem, inequality, and inclusion. *The American Mathematical Monthly*, 98(9):840–846, 1991.

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

[Gal69] David Gale. How to solve linear inequalities. *The American Mathematical Monthly*, 76(6):589–599, 1969.

[GM12] Bernd Gärtner and Jiří Matoušek. *Approximation Algorithms and Semidefinite Programming*. Springer, 2012.

[Gol70] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

[Gre70] J. Greenstadt. Variations on variable-metric methods. *Mathematics of Computation*, 24(109):1–22, 1970.

[Gro18] Alexey Gronskiy. *Statistical Mechanics and Information Theory in Approximate Robust Inference*. PhD thesis, ETH Zurich, Zurich, 2018.

[GSBR20] Robert M Gower, Mark Schmidt, Francis Bach, and Peter Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

[Haz08] Elad Hazan. Sparse approximate solutions to semidefinite programs. In Eduardo Sany Laber, Claudson Bornstein, Loana Tito Nogueira, and Luerbio Faria, editors, *LATIN 2008: Theoretical Informatics*, pages 306–316, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[HL20] Robert M. Freund Haihao Lu. Generalized stochastic Frank-Wolfe algorithm with stochastic substitute gradient for structured convex optimization. *Mathematical Programming*, pages 317–349, 2020.

[HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[Jag13] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML - International Conference on Machine Learning*, pages 427–435, 2013.

[JZ13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kha80] Leonid G. Khachiyan. Polynomial algorithms in linear programming. *U.S.S.R. Comput. Math. and Math. Phys.*, 20:53–72, 1980.

[KM72] Victor Klee and George J. Minty. How good is the simplex algorithm? In Oliver Shisha, editor, *Inequalities, III*, pages 159–175, New York, 1972. Academic Press.

[KNS16] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Łojasiewicz Condition. In *ECML PKDD 2016: Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[KPd18] Thomas Kerdreux, Fabian Pedregosa, and Alexandre d'Aspremont. Frank-wolfe with subsampling oracle, 2018.

[KSJ18] Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Global linear convergence of Newton's method without strong-convexity or Lipschitz gradients. *arXiv*, 2018.

[LBZR21] Zhize Li, Hongyan Bao, Xiangliang Zhang, and Peter Richtárik. Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *International Conference on Machine Learning*, pages 6286–6295. PMLR, 2021.

[Lev17] Kfir Levy. Online to offline conversions, universality and adaptive minibatch sizes. *NeurIPS*, 30, 2017.

[LJH+20] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020.

[LKTJ17] Francesco Locatello, Rajiv Khanna, Michael Tschannen, and Martin Jaggi. A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe. In *AISTATS - Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *PMLR*, pages 860–868, 2017.

[LO19] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *AISTATS*, pages 983–992. PMLR, 2019.

[LW19] Ching-Pei Lee and Stephen Wright. First-order algorithms converge faster than $o(1/k)$ on convex problems. In *ICML - Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, pages 3754–3762, Long Beach, California, USA, 2019.

[MG07] Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Universitext. Springer-Verlag, 2007.

[MH17] Mahesh Chandra Mukkamala and Matthias Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *ICML*, pages 2545–2553. PMLR, 2017.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Math. Dokl.*, 27(2), 1983.

[Nes12] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[Nes18] Yurii Nesterov. *Lectures on Convex Optimization*, volume 137 of *Springer Optimization and Its Applications*. Springer, second edition, 2018.

[NJLS09] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[NLST17] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2613–2621. JMLR. org, 2017.

[NN94] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Methods in Con- vex Programming*. Society for Industrial and Applied Mathematics, 1994.

[Noc80] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[NP06] Yurii Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[NSL+15] Julie Nutini, Mark W Schmidt, Issam H Laradji, Michael P Friedlander, and Hoyt A Koepke. Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection. In *ICML - Proceedings of the 32nd International Conference on Machine Learning*, pages 1632–1641, 2015.

[NY83] Arkady. S. Nemirovsky and D. B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley, 1983.

[Pad95] Manfred Padberg. *Linear Optimization and Extensions*, volume 12 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg, 1995.

[RKK18] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

[RKK19] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[Roc97] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics. Princeton University Press, 1997.

[San12] Francisco Santos. A counterexample to the hirsch conjecture. *Annals of Mathematics*, 176:383–412, 2012.

[SLRB17] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, 2017.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[ST09] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 52(10):76–84, 2009.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *J. R. Statist. Soc. B*, 58(1):267–288, 1996.

[Tse01] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[VBS19] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, pages 1195–1204. PMLR, 2019.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[Vis15] Nisheeth Vishnoi. A mini-course on convex optimization (with a view toward designing fast algorithms), 2015. `https://theory.epfl.ch/vishnoi/Nisheeth-VishnoiFall2014-ConvexOptimization.pdf`.

[WJZ+19] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. In *Advances in Neural Information Processing Systems*, pages 2406–2416, 2019.

[WLC+20] Guanghui Wang, Shiyin Lu, Quan Cheng, Wei-wei Tu, and Lijun Zhang. Sadam: A variant of adam for strongly convex functions. In *ICLR*, 2020.

[WRS+17] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[WWB19] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *ICML*, pages 6677–6686. PMLR, 2019.

[XZ14] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[Zei12] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[Zha21] Liang Zhang. Variance reduction for non-convex stochastic optimization: General analysis and new applications. Master's thesis, ETH Zurich, 2021.

[Zim16] Judith Zimmermann. *Information Processing for Effective and Stable Admission*. PhD thesis, ETH Zurich, 2016.

[ZRS+18] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *NeurIPS*, 31, 2018.