EECS 127/227AT

Optimization Models in Engineering

Course Reader

Spring 2023

Acknowledgements

This reader is based on lectures from Spring 2021, Fall 2022 and Spring 2023 iterations of EECS 127/227A by Prof. Gireeja Ranade. The reader was mostly written by Spring 2023 GSIs Druv Pai, Arwa Alanqary, and Aditya Ramabadran, and reviewed by Prof. Ranade. Fall 2022 tutor Jeffrey Wu collaborated with Druv on a writeup about the Eckart-Young theorem which was folded into the reader.

Contents

Chapter 1

Introduction

Relevant sections of the textbooks:

- [1] Chapter 1.
- [2] Chapter 1.

1.1 What is Optimization?

Try to see what the following "problems" have in common.

- A statistical model, such as a neural network, trains using finite data samples.
- A robot learns a strategy using the environment, so that it does what you want.
- A major gas company decides what mixture of different fuels to process in order to get maximum profit.
- The EECS department decides how to set class sizes in order to maximize the number of credits offered subject to budget constraints.

While it might seem that these four examples are very distinct, they can all be formulated as minimizing an *objective function* over a *feasible set*. Thus, they can all be put into the framework of optimization.

To develop the basics of optimization, including precisely defining an objective function and a feasible set, we use some motivating examples from the third and fourth "problems". (The first and second "problems" will be discussed at the very end of the course.)

Example 1 (Oil and Gas)**.** Say that we are a gas company with 10^5 barrels of crude oil that we *must* refine by an expiration date. There are two refineries: one which processes crude oil into jet fuel, and one which processes crude oil into gasoline. We can sell a barrel of jet fuel to consumers for $\$0.10$, while we can sell a barrel of gasoline fuel for $\$0.20$. So, letting x_1 be a variable denoting the number of barrels of jet fuel produced, and x_2 be a variable denoting the number of barrels of gasoline produced, we aim to solve the problem:

$$\begin{aligned} \max_{x_1,x_2} \quad & \frac{1}{10}x_1 + \frac{1}{5}x_2 \\ \text{s.t.} \quad & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned} \tag{1.1}$$

$$x_1 + x_2 = 10^5.$$

That is, we aim to choose x_1 and x_2 which maximize the *objective function* $\frac{1}{10}x_1 + \frac{1}{5}x_2$, but with the caveat that they must obey the *constraints* $x_1 \geq 0$, $x_2 \geq 0$, and $x_1 + x_2 = 10^5$. The *feasible set* is the set of all (x_1, x_2) pairs which obey the constraints. As you may have noticed, constraints can be equalities or inequalities in the x_i, which we formalize shortly.

The solution to this problem can be seen to be $(x_1^\star, x_2^\star) = (0, 10^5)$, which corresponds to refining all the crude oil into gasoline. This makes sense – after all, gasoline sells for more! And with all else equal between gasoline and jet fuel, to maximize our profit, we just need to produce gasoline.

To model another constraint, say that we need at least 10^3 gallons of jet fuel and $5 \cdot 10^2$ gallons of gasoline, we can directly incorporate them into the constraint set:

$$\begin{aligned} \max_{x_1, x_2} \quad & \frac{1}{10}x_1 + \frac{1}{5}x_2 \\ \text{s.t.} \quad & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_1 \geq 10^3 \\ & x_2 \geq 5 \cdot 10^2 \\ & x_1 + x_2 = 10^5. \end{aligned} \tag{1.2}$$

We then notice that $x_1 \geq 0$ is made redundant by the constraint $x_1 \geq 10^3$. That is, no pair (x_1, x_2) which satisfies $x_1 \geq 10^3$ is not going to satisfy $x_1 \geq 0$. Thus, we can eliminate the latter constraint, since it defines the same feasible set. We can do the same thing for the constraints $x_2 \geq 0$ and $x_2 \geq 5 \cdot 10^2$, the latter making the former redundant. Thus, we can simplify the above problem to only include the redundant constraints:

$$\begin{aligned} \max_{x_1, x_2} \quad & \frac{1}{10}x_1 + \frac{1}{5}x_2 \\ \text{s.t.} \quad & x_1 \geq 10^3 \\ & x_2 \geq 5 \cdot 10^2 \\ & x_1 + x_2 = 10^5. \end{aligned} \tag{1.3}$$

Let's say that we want to incorporate one final business need. Before, we were modeling that the oil refinement is free, since we don't have an objective or constraint term which involves this cost. Now, let us say that we can transport a total of $2 \cdot 10^6$ "barrel-miles" – that is, the number of barrels times the number of miles we can transport is no greater than $2 \cdot 10^6$. Let us further say that the jet fuel refinery is 10 miles away from the crude oil storage, and the gasoline refinery is 30 miles away from the crude oil storage. We can incorporate this further constraint into the constraint set directly:

$$\begin{aligned} \max_{x_1, x_2} \quad & \frac{1}{10}x_1 + \frac{1}{5}x_2 \\ \text{s.t.} \quad & x_1 \geq 10^3 \\ & x_2 \geq 5 \cdot 10^2 \\ & 10x_1 + 30x_2 \leq 2 \cdot 10^6 \\ & x_1 + x_2 = 10^5. \end{aligned} \tag{1.4}$$

This is a good first problem; we have a non-trivial objective function, non-trivial inequality and equality constraints, and even got to work with manipulating constraints (so as to remove redundant ones)!

This type of optimization problem is called a *linear program*. We will learn more about how to formulate and solve linear programs later in the course.

A more generic reformulation of the above optimization problem is the following "standard form".

Definition 2 (Standard Form of Optimization Problem)
We say that an optimization problem is written in *standard form* if it is of the form

$$\begin{aligned} \min_{\vec{x} \in \mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \le 0, \qquad \forall i \in \{1, \ldots, m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1, \ldots, p\}. \end{aligned} \tag{1.5}$$

Here:

- $\vec{x} \in \mathbb{R}^n$ is the optimization variable.
- $f_1, \ldots, f_m$ and $h_1, \ldots, h_p$ are functions $\mathbb{R}^n \to \mathbb{R}$.
- f_0 is the objective function.
- f_i are inequality constraint functions; the expression "$f_i(\vec{x}) \le 0$" is an inequality constraint.
- Similarly, h_j are equality constraint functions, and the expression "$h_j(\vec{x}) = 0$" is an equality constraint.
- The feasible set, i.e., the set of all $\vec{x}$ that satisfy all constraints, is

$$\Omega \doteq \left\{ \vec{x} \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} f_i(\vec{x}) \le 0, & \forall i \in \{1, \ldots, m\} \\ h_j(\vec{x}) = 0, & \forall j \in \{1, \ldots, p\} \end{array} \right\}. \tag{1.6}$$

  We can thus also write the problem (1.5) as

$$\min_{\vec{x} \in \Omega} f_0(\vec{x}). \tag{1.7}$$

- A *solution* to this optimization problem is any $\vec{x}^\star \in \Omega$ which attains the minimum value of $f(\vec{x})$ across all $\vec{x} \in \Omega$. Correspondingly, $\vec{x}^\star$ is also called a minimizer of f_0 over Ω.

It's perfectly fine if $m = 0$ (in which case there are no inequality constraints) and/or $p = 0$ (in which case there are no equality constraints). If there are no constraints, then $\Omega = \mathbb{R}^n$ and the problem is called *unconstrained*; otherwise it is called *constrained*.

Let us try another example now, which has vector-valued quantities.

Example 3. Consider the following table of EECS courses:

Class	Size	Credits	Resources per Student
127	x_1	c_1	r_1
126	x_2	c_2	r_2
182	x_3	c_3	r_3
189	x_4	c_4	r_4
162	x_5	c_5	r_5
188	x_6	c_6	r_6
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Suppose there are n classes in total. Let $\vec{x} \doteq \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$ be the decision variable, and let $\vec{c} \doteq \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}^\top \in \mathbb{R}^n$ and $\vec{r} \doteq \begin{bmatrix} r_1 & r_2 & \cdots & r_n \end{bmatrix}^\top \in \mathbb{R}^n$ be constants. Then, in order to maximize the total number of credit hours subject to a total resource budget b, we set up the linear program

$$\begin{aligned} \max_{\vec{x} \in \mathbb{R}^n} \quad & \vec{c}^\top \vec{x} \\ \text{s.t.} \quad & \vec{r}^\top \vec{x} \leq b \\ & x_i \geq 0, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{1.8}$$

As notation, instead of the last set of constraints $x_i \geq 0$, we can write the vector constraint $\vec{x} \geq \vec{0}$.

More generally, recall that if we have a vector equality constraint $\vec{h}(\vec{x}) = \vec{0}$, it can be viewed as short-hand for the several scalar equality constraints $h_1(\vec{x}) = 0, \ldots, h_p(\vec{x}) = 0$. Correspondingly, we define the vector inequality constraint $\vec{f}(\vec{x}) \leq \vec{0}$ to be short-hand for the several scalar inequality constraints $f_1(\vec{x}) \leq 0, \ldots, f_m(\vec{x}) \leq 0$.

1.2 Least Squares

We begin with one of the simplest optimization problems, that of least squares. We've probably seen this formulation before. Mathematically, we are given a data matrix $A \in \mathbb{R}^{m \times n}$ and a vector of outcomes $\vec{y} \in \mathbb{R}^m$, and attempt to find a parameter vector $\vec{x} \in \mathbb{R}^n$ which minimizes the residual $\|A\vec{x} - \vec{y}\|_2^2$. Here $\|\cdot\|_2$ is the standard Euclidean norm $\|\vec{z}\|_2 \doteq \sqrt{\vec{z}^\top \vec{z}} = \sqrt{\sum_{i=1}^n z_i^2}$; it is labeled with the 2 for a reason we will see later in the course.

More precisely, we attempt to solve the following optimization problem:

$$\min_{\vec{x} \in \mathbb{R}^n} \|A\vec{x} - \vec{y}\|_2^2 . \tag{1.9}$$

Theorem 4 (Least Squares Solution)
Let $A \in \mathbb{R}^{m \times n}$ have full column rank, and let $\vec{y} \in \mathbb{R}^m$. Then the solution to (1.9), i.e., the solution to

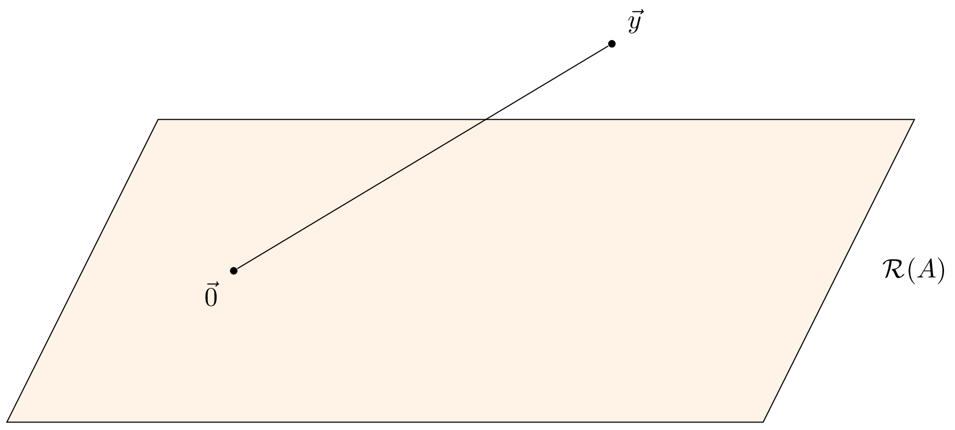
$$\min_{\vec{x} \in \mathbb{R}^n} \|A\vec{x} - \vec{y}\|_2^2 , \tag{1.9}$$

is given by

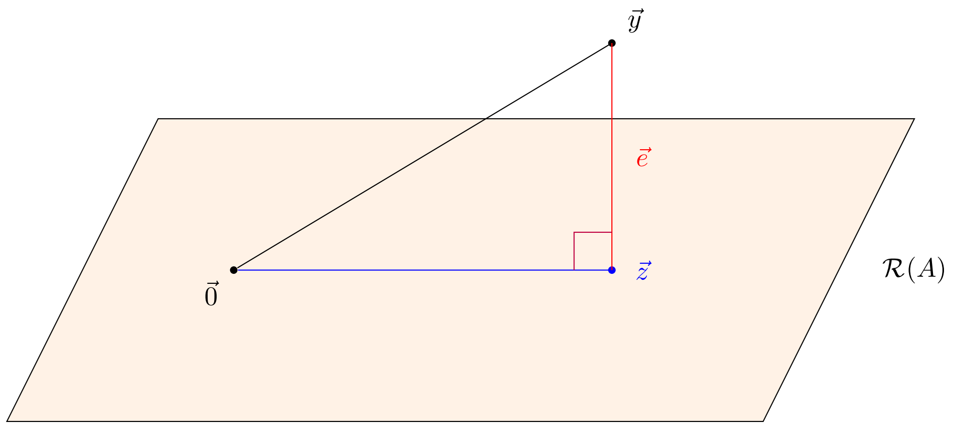
$$\vec{x}^\star = (A^\top A)^{-1} A^\top \vec{y}. \tag{1.10}$$

Proof. The idea is to find $A\vec{x} \in \mathcal{R}(A)$ which is closest to $\vec{y}$. Here $\mathcal{R}(A)$ is the range, or column space, or column span, of A. In general, We have no guarantee that $\vec{y} \in \mathcal{R}(A)$, so there is not necessarily an $\vec{x}$ such that $A\vec{x} = \vec{y}$. Instead, we are finding an approximate solution to the equation $A\vec{x} = \vec{y}$.

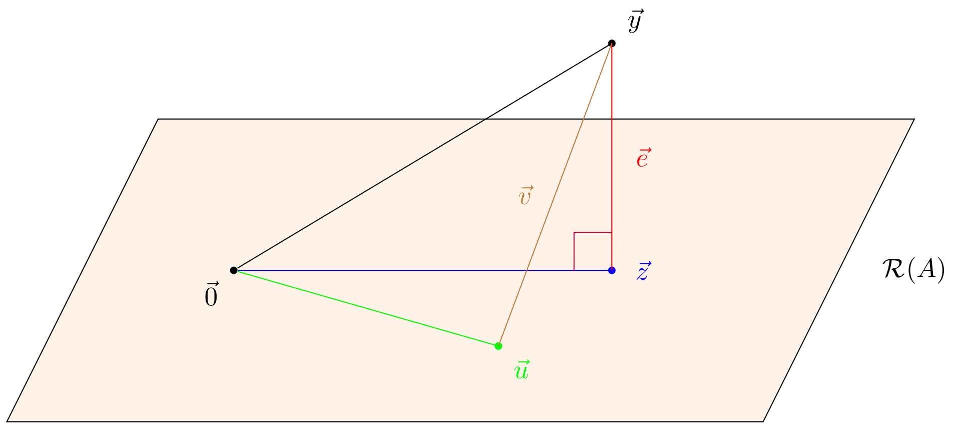
Recall that $\mathcal{R}(A)$ is a subspace, and that $\vec{y}$ itself may not belong to $\mathcal{R}(A)$. Thus we can visualize the geometry of the problem as the following picture:



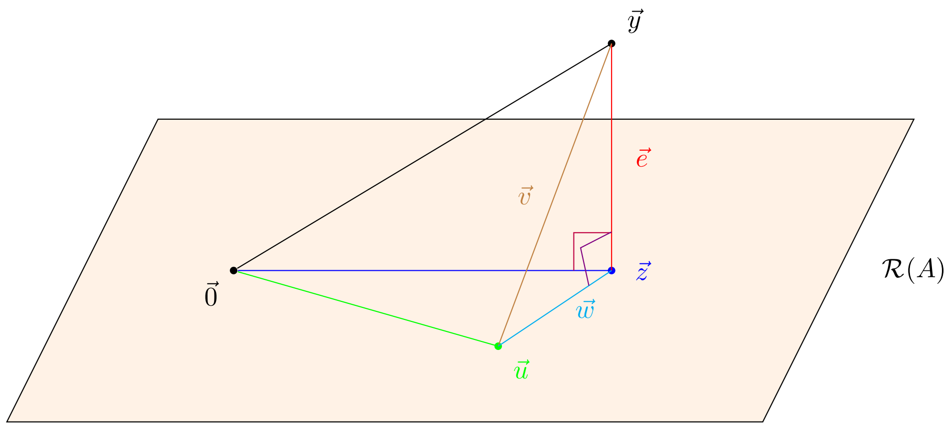
We can now solve this problem using ideas from geometry. We claim that the closest point to $\vec{y}$ contained in $\mathcal{R}(A)$ is the orthogonal projection of $\vec{y}$ onto $\mathcal{R}(A)$; call this point $\vec{z}$. Also, define $\vec{e} \doteq \vec{y} - \vec{z}$. This gives the following diagram.



From this diagram, we see that $\vec{e}$ is orthogonal to any vector in $\mathcal{R}(A)$. But remember that we still have to prove that $\vec{z}$ is the closest point to $\vec{y}$ within $\mathcal{R}(A)$. To see this, consider any another point $\vec{u} \in \mathcal{R}(A)$ and define $\vec{v} \doteq \vec{y} - \vec{u}$. This gives the following diagram:



To complete our proof, we define $\vec{w} \doteq \vec{z} - \vec{u}$, noting that the angle $\vec{u} \to \vec{z} \to \vec{y}$ is a right angle; in other words, $\vec{w}$ and $\vec{e}$ are orthogonal. This gives the following picture.



By the Pythagorean theorem, we see that

$$\begin{aligned}
\|\vec{y} - \vec{u}\|_2^2 &= \|\vec{v}\|_2^2 && (1.11)\\
&= \|\vec{w}\|_2^2 + \|\vec{e}\|_2^2 && (1.12)\\
&= \underbrace{\|\vec{z} - \vec{u}\|_2^2}_{>0} + \|\vec{e}\|_2^2 && (1.13)\\
&> \|\vec{e}\|_2^2 && (1.14)\\
&= \|\vec{y} - \vec{z}\|_2^2 . && (1.15)
\end{aligned}$$

Therefore, $\vec{z}$ is the closest point to $\vec{y}$ within $\mathcal{R}(A)$.

Now, we want to find $\vec{z} \in \mathcal{R}(A)$, i.e., the orthogonal projection of $\vec{y}$ onto $\mathcal{R}(A)$, such that $\vec{e} = \vec{y} - \vec{z}$ is orthogonal to all vectors in $\mathcal{R}(A)$. By the definition of $\mathcal{R}(A)$, it's equivalent to find $\vec{x}^\star \in \mathbb{R}^n$ such that $\vec{y} - A\vec{x}^\star$ is orthogonal to all vectors in $\mathcal{R}(A)$. Since the columns of A form a spanning set for $\mathcal{R}(A)$, it's equivalent to find $\vec{x}^\star \in \mathbb{R}^n$ such that $\vec{y} - A\vec{x}^\star$ is orthogonal to all columns of A. This implies

$$\begin{aligned}
\vec{0} &= A^\top(\vec{y} - A\vec{x}^\star) && (1.16)\\
&= A^\top\vec{y} - A^\top A\vec{x}^\star && (1.17)\\
\implies A^\top A\vec{x}^\star &= A^\top\vec{y} && (1.18)\\
\implies \vec{x}^\star &= (A^\top A)^{-1}A^\top\vec{y}. && (1.19)
\end{aligned}$$

Here $A^\top A$ is invertible because A has full column rank. □

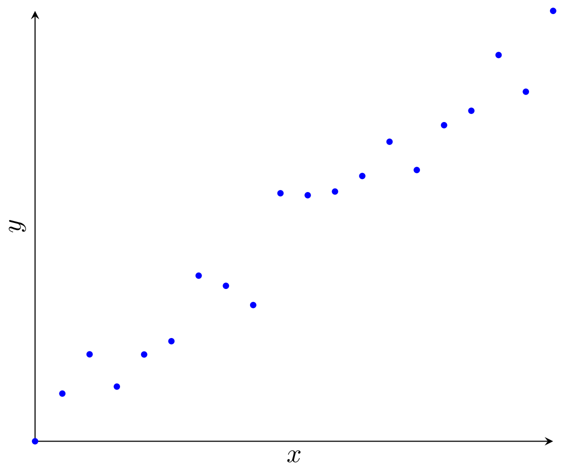
We'll conclude with a statistical application of least squares to linear regression. Suppose we are given data $(x_1, y_1), \ldots, (x_n, y_n)$, and want to fit an *affine* model $y = mx + b$ through these data points. This corresponds to approximately solving the system

$$\begin{aligned}
mx_1 + b &= y_1\\
mx_2 + b &= y_2\\
&\vdots\\
mx_n + b &= y_n.
\end{aligned} \tag{1.20}$$

Formulating it in terms of vectors and matrices, we have

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{1.21}$$

In the case where the data is noisy or inconsistent with the model, as in the below figure, the linear system will be overdetermined and have no solutions. Then, we find an approximate solution – a line of best fit – via least squares on the above system.



As a last note, solving least squares (and similar problems) is easy because it is a so-called *convex* problem. Convex problems are easy to solve because any local optimum is a global optimum, which allows us to use a variety of simple techniques to find global optima. It is generally much more difficult to solve non-convex problems, though we solve a few during this course.

We discuss much more about convexity and convex problems later in the course.

1.3 Solution Concepts and Notation

Sometimes we assign values to our optimization problems. For example in the framework of (1.5) we may write

$$\begin{aligned} p^\star = \min_{\vec{x}\in\mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \leq 0 \quad \forall i \in \{1,\dots,m\} \\ & h_j(\vec{x}) = 0 \quad \forall j \in \{1,\dots,p\}. \end{aligned} \tag{1.22}$$

On the other hand, in the framework of (1.7) and using the definition of Ω in (1.6), we may write

$$p^\star = \min_{\vec{x}\in\Omega} f_0(\vec{x}). \tag{1.23}$$

This means that $p^\star \in \mathbb{R}$ is the minimum value of f_0 over all $\vec{x} \in \Omega$; formally,

$$p^\star = \min_{\vec{x}\in\Omega} f_0(\vec{x}) \doteq \min\{f_0(\vec{x}) \mid \vec{x} \in \Omega\}. \tag{1.24}$$

As an example, consider the two-element set $\Omega = \{0, 1\}$ and $f_0(x) = 3x^2 + 2$. Then $p^\star = \min\{f(0), f(1)\} = \min\{2, 5\} = 2$. We emphasize that $p^\star$ is a *real number*, not a vector.

To extract the minimizers, i.e., the points $\vec{x} \in \Omega$ which minimize $f_0(\vec{x})$, we use the argmin notation, which gives us the set of arguments which minimize our objective function. Formally, we define:

$$\operatorname*{argmin}_{\vec{x} \in \Omega} f_0(\vec{x}) \doteq \left\{ \vec{x} \in \mathbb{R}^n \,\middle|\, f_0(\vec{x}) = \min_{\vec{u} \in \Omega} f_0(\vec{u}) \right\} \tag{1.25}$$

We can thus write the set of solutions to (1.5) as

$$\begin{aligned} \operatorname*{argmin}_{\vec{x} \in \mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \leq 0 \quad \forall i \in \{1, \ldots, m\} \\ & h_j(\vec{x}) = 0 \quad \forall j \in \{1, \ldots, p\}. \end{aligned} \tag{1.26}$$

And, as just discussed, we can write the set of solutions to (1.7) as

$$\operatorname*{argmin}_{\vec{x} \in \Omega} f_0(\vec{x}). \tag{1.27}$$

We emphasize that the argmin is a *set of vectors*, any of which are an optimal solution, i.e., a minimizer, of the optimization problem at hand. It is possible for the argmin to contain 0 vectors (in which case the minimum value is not realized and the problem has no global optima), any positive number of vectors, or an infinite number of vectors.

Let us consider the same example as before. In particular, consider the two-element set $\Omega = \{0, 1\}$ and $f_0(x) = 3x^2 + 2$. Then $\operatorname{argmin}_{x \in \Omega} f_0(x) = \{0\}$. But, in different scenarios, the argmin can have zero elements; for example, if $f_0(x) = 3x$, then $\operatorname{argmin}_{x \in \mathbb{R}} f_0(x) = \emptyset$. And it can have multiple elements; for example, if $f_0(x) = 3x^2(x-1)^2$, then $\operatorname{argmin}_{x \in \mathbb{R}} f_0(x) = \{0, 1\}$. It can even have infinitely many elements; for example, if $f_0(x) = 0$, then $\operatorname{argmin}_{x \in \mathbb{R}} f_0(x) = \mathbb{R}$.

Though we must remember to keep in mind that technically argmin is a set, in the problems we study, it usually contains exactly one element. Thus, instead of writing, for example, $\vec{x}^\star \in \operatorname{argmin}_{\vec{x} \in \Omega} f_0(\vec{x})$, we may also write $\vec{x}^\star = \operatorname{argmin}_{\vec{x} \in \Omega} f_0(\vec{x})$. The former expression is technically more correct, but both usages are fine, if — and only if — the argmin in question contains exactly one element.

1.4 (OPTIONAL) Infimum Versus Minimum

There is one remaining issue with our formulation, which we can conceptually consider as a "corner case". What happens if the minimum does not exist? This may seem like a very esoteric case, yet one can construct a straightforward example, such as the following. We know that the minimum of any set of numbers must be contained in the set. But what happens if we try to find the minimum of the open interval $(0, 1)$? For any $x \in (0, 1)$ which we claim to be our minimum, we see that $\frac{x}{2}$ is also contained in $(0, 1)$ and is smaller than x, which is a contradiction to our claim. Thus the set $(0, 1)$ has no minimum.

It seems like 0 is a useful notion of "minimum" for this set — that is, it's the largest number which is $\leq$ all numbers in the set, i.e., its "greatest lower bound" — but it isn't contained in the set and thus cannot be the minimum. Fortunately, this notion of greatest lower bound of a set is formalized in real analysis as the concept of an "infimum", denoted inf. For our purposes, we can think of the infimum as a generalization of the minimum which takes care of these corner cases and always exists. When the minimum exists, it is always equal to the infimum.

Based on this discussion, we can write our optimization problems as

$$p^\star = \inf_{\vec{x} \in \mathbb{R}^n} \quad f_0(\vec{x}) \tag{1.28}$$

$$\begin{aligned} \text{s.t.} \quad & f_i(\vec{x}) \leq 0 \quad \forall i \in \{1, \dots, m\} \\ & h_j(\vec{x}) = 0 \quad \forall j \in \{1, \dots, p\}. \end{aligned}$$

and

$$p^\star = \inf_{\vec{x} \in \Omega} f_0(\vec{x}). \tag{1.29}$$

However, the argmin retains the same definition. In fact, one can prove that if we replaced the $\min$ in the argmin definition (1.25) with $\inf$, that this "new" argmin would be exactly equivalent in every case to the "old" argmin, which we use henceforth. The analogous quantity to infimum for maximization — that is, the appropriate generalization of $\max$ — is the supremum, denoted $\sup$.

Interested readers are encouraged to consult a real analysis textbook such as [3] for a more comprehensive coverage.

Though we have gone over the technical detail here, for the rest of the course we will omit it for simplicity, and stick to using $\min$ and $\max$.

Chapter 2

Linear Algebra Review

Relevant sections of the textbooks:

- [1] Appendix A.
- [2] Chapters 2, 3, 4, 5.

2.1 Norms

2.1.1 Definitions

Definition 5 (Norm)
Let X be a vector space over $\mathbb{R}$. A function $f\colon X \to \mathbb{R}$ is a norm if:

- Positive definiteness: $f(\vec{x}) \geq 0$ for all $\vec{x} \in X$, and $f(\vec{x}) = 0$ if and only if $\vec{x} = \vec{0}$.
- Positive homogeneity: $f(\alpha\vec{x}) = |\alpha|\, f(\vec{x})$ for all $\alpha \in \mathbb{R}$ and $\vec{x} \in X$.
- Triangle inequality: $f(\vec{x} + \vec{y}) \leq f(\vec{x}) + f(\vec{y})$ for all $\vec{x}, \vec{y} \in X$.

We can check that the familiar Euclidean norm $\|\cdot\|_2 : \vec{x} \mapsto \sqrt{\sum_{i=1}^{n} x_i^2}$ satisfies these properties. A generalization of the Euclidean norm is the following very useful class of norms.

Definition 6 (ℓ^p Norms)
Let $1 \leq p < \infty$. The ℓ^p-norm on $\mathbb{R}^n$ is given by

$$\|\vec{x}\|_p \doteq \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}. \tag{2.1}$$

The ℓ^∞-norm on $\mathbb{R}^n$ is given by

$$\|\vec{x}\|_\infty \doteq \max_{i \in \{1,\ldots,n\}} |x_i|. \tag{2.2}$$

Example 7 (Examples of ℓ^p Norms)**.**

(a) The Euclidean norm, given by $\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$, is an ℓ^p-norm for $p = 2$. (This is why we gave the subscript 2 to the Euclidean norm previously).

(b) The ℓ^1-norm is given by $\|\vec{x}\|_1 = \sum_{i=1}^n |x_i|$.

(c) The ℓ^∞-norm, given by $\|\vec{x}\|_\infty = \max_{i\in\{1,\ldots,n\}} |x_i|$, is the limit of the ℓ^p norms as $p \to \infty$:

$$\|\vec{x}\|_\infty = \lim_{p\to\infty} \|\vec{x}\|_p . \tag{2.3}$$

We do not prove this here; it is left as an exercise.

2.1.2 Inequalities

There are a variety of useful *inequalities* which are associated with the ℓ^p norms. Before we provide them, we will take a second to discuss the importance of inequalities for optimization.

A priori, it may not be clear why we need to care about inequalities; why does it matter whether one arrangement of variables is always greater or less than another arrangement? It turns out that such inequalities are very helpful for characterizing the minimum and maximum of a given set of things; we can obtain upper bounds and lower bounds for things using these inequalities. This is definitely very helpful for optimization.

With that out of the way, let us get to the first major inequality.

Theorem 8 (Cauchy-Schwarz Inequality)
For any $\vec{x}, \vec{y} \in \mathbb{R}^n$, we have

$$\left|\vec{x}^\top \vec{y}\right| \leq \|\vec{x}\|_2 \|\vec{y}\|_2 . \tag{2.4}$$

Proof. Let θ be the angle between $\vec{x}$ and $\vec{y}$. We write

$$\begin{aligned} \left|\vec{x}^\top \vec{y}\right| &= \left|\|\vec{x}\|_2 \|\vec{y}\|_2 \cos\theta\right| && (2.5)\\ &= \|\vec{x}\|_2 \|\vec{y}\|_2 \left|\cos\theta\right| && (2.6)\\ &\leq \|\vec{x}\|_2 \|\vec{y}\|_2 . && (2.7) \end{aligned}$$

□

We can get this result for ℓ^2 norms. A natural next question is whether we can generalize it to ℓ^p norms for $p \neq 2$. It turns out that we can, as we demonstrate shortly.

Theorem 9 (Hölder's Inequality)
Let $1 \leq p, q \leq \infty$ such that $\frac{1}{p} + \frac{1}{q} = 1$.[a] Then for any $\vec{x}, \vec{y} \in \mathbb{R}^n$, we have

$$\left|\vec{x}^\top \vec{y}\right| \leq \sum_{i=1}^n |x_i y_i| \leq \|\vec{x}\|_p \|\vec{y}\|_q . \tag{2.8}$$

[a]Such pairs (p, q) are called *Hölder conjugates*.

This inequality collapses to Cauchy-Schwarz Inequality when $p = q = 2$. The proof is out of scope for now since it uses convexity.

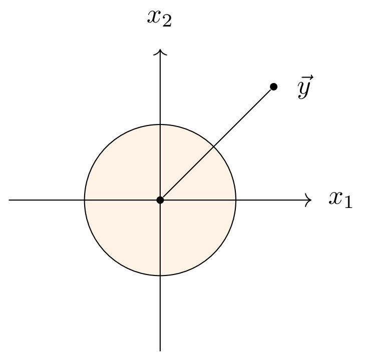
Example 10 (Dual Norms)**.** Fix $\vec{y} \in \mathbb{R}^n$. Let us solve the problem:

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \vec{x}^\top \vec{y}. \tag{2.9}$$

It is initially difficult to see how to proceed, so let us simplify the problem to get back onto familiar territory. We start with $p = 2$, so that the problem becomes:

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 \leq 1}} \vec{x}^\top \vec{y}. \tag{2.10}$$

For $n = 2$, the feasible set and $\vec{y}$ together look like the following:



For any $\vec{x} \in \mathbb{R}^n$, and θ the angle between $\vec{x}$ and $\vec{y}$, we have

$$\vec{x}^\top \vec{y} = \|\vec{x}\|_2 \|\vec{y}\|_2 \cos\theta. \tag{2.11}$$

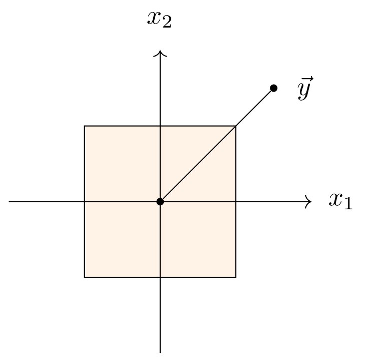
This term is maximized when $\cos\theta = 1$, or equivalently $\theta = 0$. Thus $\vec{x}$ and $\vec{y}$ must point in the same direction, i.e., $\vec{x}$ is a scalar multiple of $\vec{y}$. And since we want to maximize this dot product, we must choose $\vec{x}$ to maximize $\|\vec{x}\|_2$ subject to the constraint $\|\vec{x}\|_2 \leq 1$. Thus, we choose an $\vec{x}$ which has $\|\vec{x}\|_2 = 1$ and points in the same direction as $\vec{y}$. This gives $\vec{x}^\star = \vec{y}/\|\vec{y}\|_2$. Thus,

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 \leq 1}} \vec{x}^\top \vec{y} = (\vec{x}^\star)^\top \vec{y} = \left(\frac{\vec{y}}{\|\vec{y}\|_2}\right)^\top \vec{y} = \frac{\vec{y}^\top \vec{y}}{\|\vec{y}\|_2} = \frac{\|\vec{y}\|_2^2}{\|\vec{y}\|_2} = \|\vec{y}\|_2\,. \tag{2.12}$$

Now let us try $p = \infty$. The problem becomes

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_\infty \leq 1}} \vec{x}^\top \vec{y}. \tag{2.13}$$

The feasible set and $\vec{y}$ are given by the following diagram.



Motivated by this diagram, we see that the constraint $\|\vec{x}\|_\infty \leq 1$ is equivalent to the $2n$ constraints $-1 \leq x_i$ and $x_i \leq 1$. Also, writing out the objective function

$$\vec{x}^\top \vec{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n, \tag{2.14}$$

we see that the problem is

$$\begin{aligned} \max_{\vec{x} \in \mathbb{R}^n} \quad & (x_1 y_1 + x_2 y_2 + \cdots + x_n y_n) \\ \text{s.t.} \quad & -1 \leq x_i \leq 1, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{2.15}$$

This problem has an interesting structure that will be repeated several times in the problems we discuss in this class. Namely, the objective function is the sum of several terms, each of which involves only one x_i. And the constraints are able to be partitioned into some groups, where the constraints in each group constrain only one x_i. Thus, this problem is *separable* into n different scalar problems, such that the optimal solutions for each scalar problem form an optimal solution for the vector problem. Namely, the problems are

$$\max_{\substack{x_i \in \mathbb{R} \\ -1 \leq x_i \leq 1}} x_i y_i \tag{2.16}$$

We solve this much simpler problem by hand. If $y_i > 0$ then $x_i^\star = 1$; on the other hand, if $y_i \leq 0$ then $x_i^\star = -1$. To summarize, $x_i^\star = \operatorname{sgn}(y_i)$, so that $x_i^\star y_i = |y_i|$.

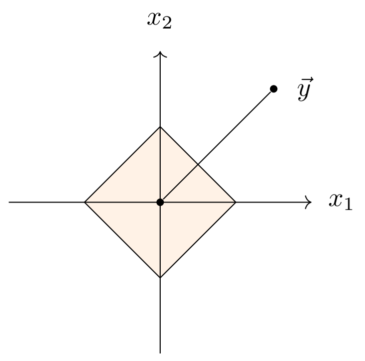
Putting all the scalar problems together, we see that $\vec{x}^\star = \operatorname{sgn}(\vec{y})$, and the vector problem's optimal value is given by

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_\infty \leq 1}} \vec{x}^\top \vec{y} = (\vec{x}^\star)^\top \vec{y} = \sum_{i=1}^{n} x_i^\star y_i = \sum_{i=1}^{n} \operatorname{sgn}(y_i) y_i = \sum_{i=1}^{n} |y_i| = \|\vec{y}\|_1 . \tag{2.17}$$

As a final exercise, we consider $p = 1$, so that the problem becomes

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_1 \leq 1}} \vec{x}^\top \vec{y}. \tag{2.18}$$

For $n = 2$, the feasible set and $\vec{y}$ together look like the following:



We now bound the objective as

$$\begin{aligned} \vec{x}^\top \vec{y} &\leq \left|\vec{x}^\top \vec{y}\right| && (2.19) \\ &= \left|\sum_{i=1}^{n} x_i y_i\right| && (2.20) \end{aligned}$$

$$\leq \sum_{i=1}^{n} |x_i y_i| \qquad \text{by triangle inequality} \tag{2.21}$$

$$= \sum_{i=1}^{n} |x_i| \, |y_i| \tag{2.22}$$

$$\leq \sum_{i=1}^{n} |x_i| \left(\max_{i \in \{1,\dots,n\}} |y_i| \right) \tag{2.23}$$

$$= \left(\max_{i \in \{1,\dots,n\}} |y_i| \right) \sum_{i=1}^{n} |x_i| \tag{2.24}$$

$$= \|\vec{y}\|_\infty \, \|\vec{x}\|_1 \tag{2.25}$$

$$\leq \|\vec{y}\|_\infty \, . \tag{2.26}$$

Thus we have

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_1 \leq 1}} \vec{x}^\top \vec{y} \leq \|\vec{y}\|_\infty \, . \tag{2.27}$$

This inequality is actually an equality. To show this, we need to show the reverse inequality

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_1 \leq 1}} \vec{x}^\top \vec{y} \underbrace{\geq}_{\text{need to show}} \|\vec{y}\|_\infty \, . \tag{2.28}$$

And showing *this* inequality amounts to choosing, for our fixed $\vec{y}$, a $\vec{x}$ such that $\|\vec{x}\|_1 \leq 1$ and $\vec{x}^\top \vec{y} \geq \|\vec{y}\|_\infty$. This is also called "showing the maximum is attained". To do this, we can find an $\vec{x}$ such that $\|\vec{x}\|_p \leq 1$ and all the inequalities in the chain are met with equality.

- First, the inequality in (2.21) is a triangle inequality with the absolute value, i.e., $|\sum_{i=1}^n x_i y_i| \leq \sum_{i=1}^n |x_i y_i|$. To make sure this is an equality, it's enough to make sure that all terms $x_i y_i$ are the same sign or 0.
- Next, the inequality in (2.23) says that $\sum_{i=1}^n |x_i| \, |y_i| \leq \sum_{i=1}^n |x_i| \left(\max_{i \in \{1,\dots,n\}} |y_i| \right)$. The most obvious instance in which this inequality is met with equality is when $|y_i| = \max_{j \in \{1,\dots,n\}} |y_j|$ for all i. But we can't choose $\vec{y}$, as it's fixed, so we can't be assured that this holds. An alternate way in which this holds is that $|x_i| = 0$ for all i for which $|y_i| \neq \max_{j \in \{1,\dots,n\}} |y_j|$, i.e., $i \notin \operatorname{argmax}_{j \in \{1,\dots,n\}} |y_j|$.
- Finally, the inequality in (2.26) says that $\|\vec{x}\|_1 \|\vec{y}\|_\infty \leq \|\vec{y}\|_\infty$; to meet this inequality with equality, it is sufficient to have $\|\vec{x}\|_1 = 1$.

To meet all three of these constraints, we can construct $\vec{x}^\star$ via the following process:

- For each $i \notin \operatorname{argmax}_{j \in \{1,\dots,n\}} |y_j|$, set $\widetilde{x}_i = 0$, as per the second bullet point above.
- For each $i \in \operatorname{argmax}_{j \in \{1,\dots,n\}} |y_j|$, set $\widetilde{x}_i = \operatorname{sgn}(y_i)$, as per the first bullet point above.
- To get the true solution vector $\vec{x}^\star$, divide $\vec{\widetilde{x}}$ by $\left\|\vec{\widetilde{x}}\right\|_1$; that is, $\vec{x}^\star = \vec{\widetilde{x}} / \left\|\vec{\widetilde{x}}\right\|_1$. This ensures that $\|\vec{x}^\star\|_1 = 1$, as per the third bullet point above.

This $\vec{x}^\star$ "achieves the maximum", showing that

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_1 \leq 1}} \vec{x}^\top \vec{y} = \|\vec{y}\|_\infty \, . \tag{2.29}$$

This notion where the ℓ^2-norm constraint leads to the ℓ^2-norm objective, the ℓ^∞-norm constraint leads to the ℓ^1-norm objective, and the ℓ^1-norm constraint leads to the ℓ^∞-norm objective, hints at a greater pattern. Indeed, one can show that for $1 \leq p, q \leq \infty$ such that $\frac{1}{p} + \frac{1}{q} = 1$, an ℓ^p-norm constraint leads to an ℓ^q-norm objective:

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \vec{x}^\top \vec{y} = \|\vec{y}\|_q . \tag{2.30}$$

As before, we can prove this equality by proving the two constituent inequalities:

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \vec{x}^\top \vec{y} \leq \|\vec{y}\|_q \qquad \text{and} \qquad \max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \vec{x}^\top \vec{y} \geq \|\vec{y}\|_q . \tag{2.31}$$

The proof of the first inequality ($\leq$) follows from applying Hölder's inequality to the objective function:

$$\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \vec{x}^\top \vec{y} \leq \max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \|\vec{x}\|_p \|\vec{y}\|_q = \|\vec{y}\|_q \cdot \max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_p \leq 1}} \|\vec{x}\|_p = \|\vec{y}\|_q . \tag{2.32}$$

The second inequality ($\geq$) can follow if, for our fixed choice of $\vec{y}$, we produce some $\vec{x}$ such that $\|\vec{x}\|_p \leq 1$ and $\vec{x}^\top \vec{y} \geq \|\vec{y}\|_q$, i.e., "the maximum is attained". This is more complicated to do, and we won't do it here.

The above equality (2.30) means that the norms $\|\cdot\|_p$ and $\|\cdot\|_q$ are so-called *dual* norms. We will explore aspects of duality later in the course, though frankly we are just scratching the surface.

These problems, which are short and easy to state, contain a couple of core ideas within their solutions, which are broadly generalizable to a lot of optimization problems. For your convenience, we discuss these explicitly below.

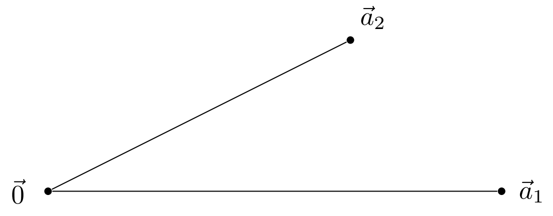
Problem Solving Strategy 11 (Separating Vector Problems into Scalar Problems)**.** *When trying to simplify an optimization problem, try to see if you can simplify it into several independent scalar problems. Then solve each scalar problem — this is usually much easier than solving the whole vector problem at once. The optimal solutions to each scalar problem will then form the optimal solution to the whole vector problem.*

Problem Solving Strategy 12 (Proving Optimality in an Optimization Problem)**.** *To solve an optimization problem, you can use inequalities to bound the objective function, and then try to show that this bound is tight by finding a feasible choice of optimization variable which makes all the inequalities into equalities.*

2.2 Gram-Schmidt and QR Decomposition

The Gram-Schmidt algorithm is a way to turn a linearly independent set $\{\vec{a}_1, \ldots, \vec{a}_k\}$ of vectors into an *orthonormal* set $\{\vec{q}_1, \ldots, \vec{q}_k\}$ which spans the same space. To reiterate, an orthonormal set is a set of vectors in which each vector has norm 1 and is orthogonal to all others in the basis.

Suppose for simplicity that $n = k = 2$, and that we have the following vectors.

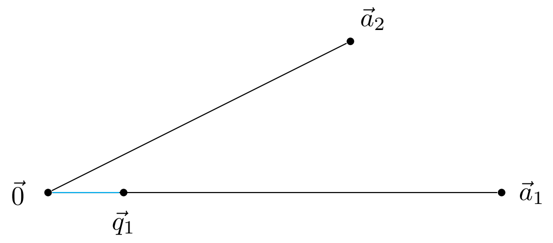


We begin with $\vec{a}_1$. We want to construct a vector $\vec{q}_1$ such that

- it's orthogonal to all the $\vec{q}_i$ which came before it — which is none of them, so we don't have to worry; and
- it has unit norm, so $\|\vec{q}_1\|_2 = 1$.

To achieve this, the simplest choice is

$$\vec{q}_1 \doteq \frac{\vec{a}_1}{\|\vec{a}_1\|_2}. \tag{2.33}$$



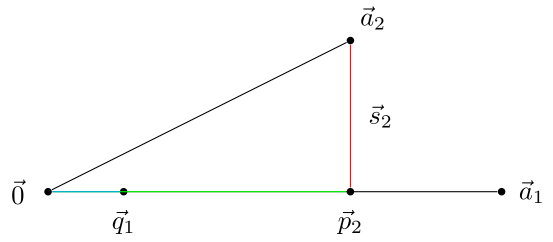
Then we go to $\vec{a}_2$. To find $\vec{q}_2$ which is orthogonal to all the $\vec{q}_i$ before it — that is, $\vec{q}_1$ — we subtract off the orthogonal projection of $\vec{a}_2$ onto $\vec{q}_1$ from $\vec{a}_2$. The orthogonal projection of $\vec{a}_2$ onto $\vec{q}_1$ is given by

$$\vec{p}_2 \doteq \vec{q}_1(\vec{q}_1^\top \vec{a}_2) \tag{2.34}$$

and so the projection residual is given by

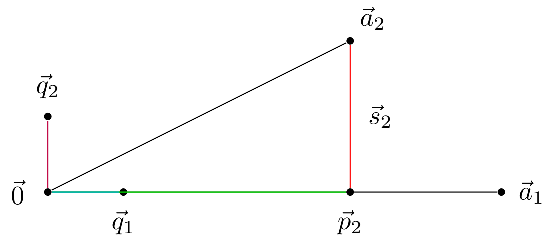
$$\vec{s}_2 \doteq \vec{a}_2 - \vec{p}_2 = \vec{a}_2 - \vec{q}_1(\vec{q}_1^\top \vec{a}_2). \tag{2.35}$$

Note that these formulas *only* hold because $\vec{q}_1$ is normalized, i.e., has norm 1.



While $\vec{s}_2$ is orthogonal to $\vec{q}_1$, because we want a $\vec{q}_2$ that is normalized, we normalize $\vec{s}_2$ to get $\vec{q}_2$:

$$\vec{q}_2 \doteq \frac{\vec{s}_2}{\|\vec{s}_2\|_2}. \tag{2.36}$$



If we had a vector $\vec{q}_3$ (and weren't limited by drawing in 2D space), we would ensure that $\vec{q}_3$ were orthogonal to $\vec{q}_1$ and $\vec{q}_2$, as well as normalized, in a similar way as before. First we would compute the projection

$$\vec{p}_3 \doteq \vec{q}_1(\vec{q}_1^\top \vec{a}_3) + \vec{q}_2(\vec{q}_2^\top \vec{a}_3). \tag{2.37}$$

and the residual

$$\vec{s}_3 \doteq \vec{a}_3 - \vec{p}_3 = \vec{a}_3 - \vec{q}_1(\vec{q}_1^\top \vec{a}_3) - \vec{q}_2(\vec{q}_2^\top \vec{a}_3). \tag{2.38}$$

These projection formulas only hold because $\{\vec{q}_1, \vec{q}_2\}$ is an orthonormal set. And then we could compute

$$\vec{q}_3 \doteq \frac{\vec{s}_3}{\|\vec{s}_3\|_2}. \tag{2.39}$$

And so on. The general algorithm goes similar.

Algorithm 1 Gram-Schmidt algorithm.

```
1: function GRAMSCHMIDTALGORITHM(linearly independent set $\{\vec{a}_1, \dots, \vec{a}_k\}$)
2:     $\vec{q}_1 \doteq \vec{a}_1 / \|\vec{a}_1\|_2$.
3:     for $i \in \{2, 3, \dots, k\}$ do
4:         $\vec{p}_i \doteq \sum_{j=1}^{i-1} \vec{q}_j(\vec{q}_j^\top \vec{a}_i)$
5:         $\vec{s}_i \doteq \vec{a}_i - \vec{p}_i$
6:         $\vec{q}_i \doteq \vec{s}_i / \|\vec{s}_i\|_2$
7:     end for
8:     return orthonormal set $\{\vec{q}_1, \dots, \vec{q}_k\}$
9: end function
```

This algorithm has the following two properties, which you can formally prove as an exercise.

Proposition 13 (Gram-Schmidt Algorithm)
Algorithm 1 has the following properties:

1. For each $i \in \{1, \dots, k\}$, we have

$$\operatorname{span}(\vec{a}_1, \dots, \vec{a}_i) = \operatorname{span}(\vec{q}_1, \dots, \vec{q}_i). \tag{2.40}$$

   In particular, $\{\vec{a}_1, \dots, \vec{a}_k\}$ spans the same subspace as $\{\vec{q}_1, \dots, \vec{q}_k\}$, as was stated in our original goal.

2. $\{\vec{q}_1, \dots, \vec{q}_k\}$ is an orthonormal set.

The Gram-Schmidt algorithm leads to something called the QR decomposition. Because, for each i, we have $\operatorname{span}(\vec{a}_1, \dots, \vec{a}_i) = \operatorname{span}(\vec{q}_1, \dots, \vec{q}_i)$, it means that we can write $\vec{a}_i$ as a linear combination of the $\vec{q}_j$:

$$\vec{a}_i = r_{1i}\vec{q}_1 + r_{2i}\vec{q}_2 + \cdots + r_{ii}\vec{q}_i = \sum_{j=1}^{i} r_{ji}\vec{q}_j \tag{2.41}$$

Putting all the k equations in a matrix form, we can write

$$\begin{bmatrix} \vec{a}_1 & \cdots & \vec{a}_k \end{bmatrix} = \begin{bmatrix} \vec{q}_1 & \cdots & \vec{q}_k \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ 0 & r_{22} & \cdots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{kk} \end{bmatrix}. \tag{2.42}$$

More generally, we can decompose every tall matrix with full column rank into a product of a tall matrix with orthonormal columns Q and an upper-triangular matrix R.

Theorem 14 (QR Decomposition)
Let $A \in \mathbb{R}^{n \times k}$ where $k \leq n$ (so A is tall). Suppose A has full column rank. Then there is a matrix $Q \in \mathbb{R}^{n \times k}$ with orthonormal columns, and a matrix $R \in \mathbb{R}^{k \times k}$ which is upper triangular, such that $A = QR$.

As a final note, there are various alterations to the QR decomposition that work for matrices which are wide and/or do not have full column rank. Those are out of scope, but the idea is the same.

The QR decomposition is also relevant in numerical linear algebra, where it can be used to solve tall linear systems $A\vec{x} = \vec{y}$ efficiently, especially if the underlying matrix A has special structure. All such connections are out of scope.

2.3 Fundamental Theorem of Linear Algebra

The fundamental theorem of linear algebra is a tool for understanding what happens to vectors and vector spaces under a linear transformation. Matrix multiplication transforms one vector space into another. This is helpful for allowing us to change our coordinate system, which tells us more about the problem.

Definition 15 (Direct Sum)
Let $U, V \subseteq \mathbb{R}^n$ be subspaces. We say that U and V *direct sum* to $\mathbb{R}^n$, denoted $U \oplus V = \mathbb{R}^n$, if and only if:

- Every vector $\vec{x} \in \mathbb{R}^n$ can be written as $\vec{x} = \vec{x}_1 + \vec{x}_2$, where $\vec{x}_1 \in U$ and $\vec{x}_2 \in V$.
- Furthermore, this decomposition is unique, in the sense that if $\vec{x} = \vec{x}_1 + \vec{x}_2 = \vec{y}_1 + \vec{y}_2$ are two instances of the above decomposition, then $\vec{x}_1 = \vec{y}_1$ and $\vec{x}_2 = \vec{y}_2$.

Theorem 16 (Fundamental Theorem of Linear Algebra)
Let $A \in \mathbb{R}^{m \times n}$. Then

$$\mathcal{N}(A) \oplus \mathcal{R}\left(A^\top\right) = \mathbb{R}^n. \tag{2.43}$$

Note that we *cannot* replace $\mathcal{R}\left(A^\top\right)$ by $\mathcal{R}(A)$, since vectors in $\mathcal{R}(A)$ and $\mathcal{N}(A)$ do not even have the same number of entries or lie in the same Euclidean space. If we want to make a statement about $\mathcal{R}(A)$, we can replace A by $A^\top$ in the above theorem to get the following corollary.

Corollary 17. *Let $A \in \mathbb{R}^{m \times n}$. Then*

$$\mathcal{N}\left(A^\top\right) \oplus \mathcal{R}(A) = \mathbb{R}^m. \tag{2.44}$$

To prove the fundamental theorem of linear algebra, we use a tool called the *orthogonal decomposition theorem*.

Definition 18 (Orthogonal Complement)
Let $S \subseteq \mathbb{R}^n$ be a subspace. The *orthogonal complement* of S, denoted $S^\perp$, is

$$S^\perp \doteq \{\vec{x} \in \mathbb{R}^n \mid \vec{s}^\top \vec{x} = 0 \text{ for all } \vec{s} \in S\} \tag{2.45}$$

Theorem 19 (Orthogonal Decomposition Theorem, Theorem 2.1 of [2])
Let $S \subseteq \mathbb{R}^n$ be a subspace. Then

$$S \oplus S^\perp = \mathbb{R}^n. \tag{2.46}$$

Proof. Deferred to the textbook. □

Using this theorem, the only thing we need to show to prove the fundamental theorem of linear algebra is that $\mathcal{N}(A)$ and $\mathcal{R}(A^\top)$ are orthogonal complements. We do this below.

Proof of Theorem 16. By Theorem 19, the only thing we need to show is that $\mathcal{N}(A) = \mathcal{R}(A^\top)^\perp$. This is a set equality; we show it by showing that $\mathcal{N}(A) \subseteq \mathcal{R}(A^\top)^\perp$ and that $\mathcal{N}(A) \supseteq \mathcal{R}(A^\top)^\perp$.

We first want to show that $\mathcal{N}(A) \subseteq \mathcal{R}(A^\top)^\perp$. That is, we want to show that for any $\vec{x} \in \mathcal{N}(A)$ we have $\vec{x} \in \mathcal{R}(A^\top)^\perp$. That is, for any $\vec{y} \in \mathcal{R}(A^\top)$, we want to show that $\vec{y}^\top \vec{x} = 0$.

Since $\vec{y} \in \mathcal{R}(A^\top)$ we can write $\vec{y} = A^\top \vec{w}$ for some $\vec{w} \in \mathbb{R}^m$. Then, since $\vec{x} \in \mathcal{N}(A)$ we have $A\vec{x} = \vec{0}$, so

$$\begin{aligned} \vec{y}^\top \vec{x} &= (A^\top \vec{w})^\top \vec{x} && (2.47)\\ &= \vec{w}^\top A\vec{x} && (2.48)\\ &= \vec{w}^\top \vec{0} && (2.49)\\ &= 0. && (2.50) \end{aligned}$$

Thus $\vec{x}$ and $\vec{y}$ are orthogonal, so $\vec{x} \in \mathcal{R}(A^\top)^\perp$, which shows that $\mathcal{N}(A) \subseteq \mathcal{R}(A^\top)^\perp$.

We now want to show that $\mathcal{R}(A^\top)^\perp \subseteq \mathcal{N}(A)$. That is, we want to show that for any $\vec{x} \in \mathcal{R}(A^\top)^\perp$, we want to show that $\vec{x} \in \mathcal{N}(A)$. That is, we want to show that $A\vec{x} = \vec{0}$.

By definition, for every $\vec{y} \in \mathcal{R}(A^\top)$, we have $\vec{y}^\top \vec{x} = 0$. By writing $\vec{y} = A^\top \vec{w}$ for arbitrary $\vec{w} \in \mathbb{R}^m$, we get that for every $\vec{w} \in \mathbb{R}^m$ we have $(A^\top \vec{w})^\top \vec{x} = 0$. But the left-hand side is $\vec{w}^\top A\vec{x}$, so we have that $\vec{w}^\top A\vec{x} = 0$ for every $\vec{w} \in \mathbb{R}^m$. This is true for all $\vec{w} \in \mathbb{R}^m$, so it is true for the specific choice of $\vec{w} = A\vec{x}$, which yields

$$\begin{aligned} 0 &= \vec{w}^\top A\vec{x} && (2.51)\\ &= (A\vec{x})^\top A\vec{x} && (2.52)\\ &= \|A\vec{x}\|_2^2 && (2.53)\\ \implies A\vec{x} &= \vec{0}. && (2.54) \end{aligned}$$

This implies that $\vec{x} \in \mathcal{N}(A)$ as desired, so $\mathcal{R}(A^\top)^\perp \subseteq \mathcal{N}(A)$.

Thus, we have shown that $\mathcal{N}(A) = \mathcal{R}(A^\top)^\perp$, and so by Theorem 19 we have $\mathcal{N}(A) \oplus \mathcal{R}(A^\top) = \mathbb{R}^n$. □

This will help us solve a very important optimization problem, which is considered "dual" to least squares in some sense. Recall that least squares helps us find an approximate solution to the linear system $A\vec{x} = \vec{y}$, when A is a *tall* matrix with full column rank. In other words, the linear system is over-determined, there are many more equations than unknowns, and there are generally no exact solutions, so we pick the solution with minimum squared error.

What about when A is a *wide* matrix with full *row* rank? There are now more unknowns than equations, and infinitely many exact solutions. So how do we pick one solution in particular? It really depends on which engineering problem we are solving. One *common* solution is to pick the minimum-energy or minimum-norm problem, which is the solution to the optimization problem:

$$\min_{\vec{x} \in \mathbb{R}^n} \quad \|\vec{x}\|_2^2 \tag{2.55}$$

$$\text{s.t.} \quad A\vec{x} = \vec{y}.$$

Note that this principle of choosing the smallest or simplest solution — the "Occam's Razor" principle — is much more broadly generalized beyond the case of finding solutions to linear systems, and is used within control theory and machine learning. But we deal with just this linear system case for now.

Theorem 20 (Minimum-Norm Solution)

Let $A \in \mathbb{R}^{m\times n}$ have full row rank, and let $\vec{y} \in \mathbb{R}^m$. Then the solution to Equation (2.55), i.e., the solution to

$$\begin{aligned} \min_{\vec{x}\in\mathbb{R}^n} \quad & \|\vec{x}\|_2^2 \\ \text{s.t.} \quad & A\vec{x} = \vec{y}, \end{aligned} \tag{2.55}$$

is given by

$$\vec{x}^\star = A^\top (AA^\top)^{-1}\vec{y}. \tag{2.56}$$

Proof. Observe that the constraint $A\vec{x} = \vec{y}$ under-specifies the $\vec{x}$ — in particular, any component of $\vec{x}$ in $\mathcal{N}(A)$ will not affect the constraint and only the objective. In this sense, it is "wasteful", and we should intuitively remove it. This motivates using Theorem 16 to decompose $\vec{x}$ into a component inside $\mathcal{N}(A)$ — which we want to remove — and a component inside $\mathcal{R}(A^\top)$ — which we will optimize over.

Indeed, write $\vec{x} = \vec{u} + \vec{v}$, where $\vec{u} \in \mathcal{N}(A)$ and $\vec{v} \in \mathcal{R}(A^\top)$. Thus, there exists $\vec{w} \in \mathbb{R}^m$ such that $\vec{v} = A^\top\vec{w}$. The constraint becomes

$$\begin{aligned} \vec{y} &= A\vec{x} && (2.57)\\ &= A(\vec{u} + \vec{v}) && (2.58)\\ &= A\vec{u} + A\vec{v} && (2.59)\\ &= \vec{0} + AA^\top\vec{w} && (2.60)\\ &= AA^\top\vec{w}. && (2.61) \end{aligned}$$

And the objective function becomes

$$\begin{aligned} \|\vec{x}\|_2^2 &= \|\vec{u} + \vec{v}\|_2^2 && (2.62)\\ &= \vec{u}^\top\vec{u} + 2\vec{u}^\top\vec{v} + \vec{v}^\top\vec{v} && (2.63)\\ &= \|\vec{u}\|_2^2 + 2\vec{v}^\top\vec{u} + \|\vec{v}\|_2^2 && (2.64)\\ &= \|\vec{u}\|_2^2 + 2(A^\top\vec{w})^\top\vec{u} + \|\vec{v}\|_2^2 && (2.65)\\ &= \|\vec{u}\|_2^2 + 2\vec{w}^\top A\vec{u} + \|\vec{v}\|_2^2 && (2.66)\\ &= \|\vec{u}\|_2^2 + 2\vec{w}^\top\vec{0} + \|\vec{v}\|_2^2 && (2.67)\\ &= \|\vec{u}\|_2^2 + 2\cdot 0 + \|\vec{v}\|_2^2 && (2.68)\\ &= \|\vec{u}\|_2^2 + \|\vec{v}\|_2^2 && (2.69)\\ &= \|\vec{u}\|_2^2 + \left\|A^\top\vec{w}\right\|_2^2 && (2.70) \end{aligned}$$

Thus, the minimum-norm problem can be reformulated in terms of $\vec{u}$ and $\vec{w}$:

$$\min_{\substack{\vec{u}\in\mathbb{R}^n\\ \vec{w}\in\mathbb{R}^m}} \quad \|\vec{u}\|_2^2 + \left\|A^\top\vec{w}\right\|_2^2 \tag{2.71}$$

$$\begin{aligned} \text{s.t.} \quad & \vec{y} = AA^\top \vec{w} \\ & A\vec{u} = \vec{0}. \end{aligned}$$

Now, because A has full row rank, $AA^\top$ is invertible, so the first constraint implies that $\vec{w}^\star = (AA^\top)^{-1}\vec{y}$, so $\vec{v}^\star = A^\top \vec{w}^\star = A^\top(AA^\top)^{-1}\vec{y}$. And because we are trying to minimize the objective, which only involves $\vec{u}$ through $\|\vec{u}\|_2^2$, the ideal solution is to set $\vec{u}^\star = \vec{0}$, which also satisfies the second constraint and so is feasible. Thus $\vec{x}^\star = \vec{v}^\star = A^\top(AA^\top)^{-1}\vec{y}$ as desired. □

2.4 Symmetric Matrices

Symmetric matrices are a sub-class of matrices which have many special properties, and in engineering applications one usually tries to work with symmetric matrices as much as possible.

Definition 21 (Symmetric Matrix)
Let $A \in \mathbb{R}^{n\times n}$ be a square matrix. We say that A is *symmetric* if $A = A^\top$. The set of all symmetric matrices is denoted $\mathbb{S}^n$.

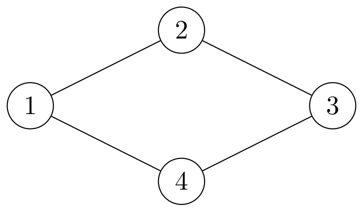
Equivalently, $A_{ij} = A_{ji}$ for all i and j.

Example 22. The 2×2 matrix $\begin{bmatrix} a & b \\ b & c \end{bmatrix}$ is symmetric.

Example 23 (Covariance Matrices)**.** Any matrix of the form $A = BB^\top$, such as the covariance matrices we will discuss in the next section, is a symmetric matrix, since

$$A^\top = (BB^\top)^\top = (B^\top)^\top(B)^\top = BB^\top = A. \tag{2.72}$$

Example 24 (Adjacency Matrix)**.** Consider an *undirected* connected graph $G = (V, E)$, for example the following:



Its adjacency matrix A has coordinate $A_{ij} = 1$ if $(i, j) \in E$, and $A_{ij} = 0$ otherwise; in the above example, we have

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}. \tag{2.73}$$

Since the graph is undirected, $(i, j) \in E$ if and only if $(j, i) \in E$, so $A_{ij} = A_{ji}$, and so A is a symmetric matrix.

Why do we care about symmetric matrices? Symmetric matrices have two nice properties: real eigenvalues, and guaranteed diagonalizability.

In general, a (non-symmetric) matrix need not be diagonalizable. For example, the matrix $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ is not diagonalizable. How can we characterize the diagonalizability of a matrix, then?

First, we will need the following definitions.

Definition 25 (Multiplicities)
Let $A \in \mathbb{R}^{n\times n}$, and let λ be an eigenvalue of A.

(a) The *algebraic multiplicity* μ of eigenvalue λ in A is the number of times λ is a root of the characteristic polynomial $p_A(x) \doteq \det(xI - A)$ of A, i.e., it is the power of $(x-\lambda)$ in the factorization of $p_A(x)$.

(b) The *geometric multiplicity* ϕ of eigenvalue λ in A is the dimension of the null space $\Phi \doteq \mathcal{N}(\lambda I - A)$.

Theorem 26 (Diagonalizability)
A square matrix $A \in \mathbb{R}^{n\times n}$ is diagonalizable if and only if every eigenvalue of A has equal algebraic and geometric multiplicities.

Example 27 (Multiplicities of Degenerate Matrix)**.** We were earlier told that the matrix $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ is not diagonalizable. To check this, let us compute its eigenvalues, algebraic multiplicities, and geometric multiplicities.

First, its characteristic polynomial is

$$\begin{aligned} p_A(x) &= \det(xI - A) && (2.74)\\ &= \det\left(\begin{bmatrix} x-1 & 1 \\ 0 & x-1 \end{bmatrix}\right) && (2.75)\\ &= (x-1)^2. && (2.76)\end{aligned}$$

Thus, A has only one eigenvalue $\lambda = 1$. Since $(x-1)$ has power 2 in the factorization of p_A, the eigenvalue $\lambda = 1$ has algebraic multiplicity $\mu = 2$.

The corresponding null space is

$$\begin{aligned} \Phi &= \mathcal{N}(\lambda I - A) && (2.77)\\ &= \mathcal{N}\left(\begin{bmatrix} 1-1 & 1 \\ 0 & 1-1 \end{bmatrix}\right) && (2.78)\\ &= \mathcal{N}\left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\right) && (2.79)\\ &= \operatorname{span}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) && (2.80)\end{aligned}$$

which has dimension $\phi = 1$. Thus, for $\lambda = 1$, we have $\mu \neq \phi$ and the matrix is indeed not diagonalizable.

This allows us to formally state the spectral theorem.

Theorem 28 (Spectral Theorem)
Let $A \in \mathbb{S}^n$ have eigenvalues λ_i with algebraic multiplicities μ_i, eigenspaces $\Phi_i \doteq \mathcal{N}(\lambda_i I - A)$, and geometric multiplicities $\phi_i \doteq \dim(\Phi_i)$.

(a) All eigenvalues are real: $\lambda_i \in \mathbb{R}$ for each i.

(b) Eigenspaces corresponding to different eigenvalues are orthogonal: Φ_i and Φ_j are orthogonal subspaces, i.e., for every $\vec{p}_i \in \Phi_i$ and $\vec{p}_j \in \Phi_j$ we have $\vec{p}_i^\top \vec{p}_j = 0$.

(c) A is diagonalizable: $\mu_i = \phi_i$ for each i.

(d) A is *orthonormally diagonalizable*; there exists an *orthonormal* matrix $U \in \mathbb{R}^{n \times n}$ and *diagonal* matrix $\Lambda \in \mathbb{R}^{n \times n}$ such that $A = U\Lambda U^\top$.

Recall that orthonormal matrices are matrices whose columns are orthonormal, i.e., are pairwise orthogonal and unit-norm. Orthonormal matrices U have the nice property that $U^\top U = I$, and if U is square, then $U^\top = U^{-1}$.

Proof of Theorem 28. Part (a) might be left to homework; part (b) will definitely be left to homework; we prove parts (c) and (d) here. In particular, we assume that parts (a) and (b) are true, and attempt to prove (d). Note that (d) implies (c), as an orthonormal diagonalization is a type of diagonalization, and so the existence of an orthonormal diagonalization must require the algebraic and geometric multiplicities to be equal.

Our proof strategy is to use induction on n, the size of the matrix. The base case of our induction is 1×1 matrices, for which the diagonalization is trivial. Now consider the inductive step. Our hope is, given $A \in \mathbb{S}^n$ which has eigenvalue λ, to get a decomposition of the form

$$A = V \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & B \end{bmatrix} V^\top \qquad \text{or equivalently} \qquad V^\top A V = \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & B \end{bmatrix} \tag{2.81}$$

where $V \in \mathbb{R}^{n \times n}$ is orthonormal and $B \in \mathbb{S}^{n-1}$ is symmetric. If we can do that, then we can inductively diagonalize $B = W\Gamma W^\top$, and finally use that to construct a diagonalization for $A = U\Lambda U^\top$, where $U \in \mathbb{R}^{n \times n}$ is orthonormal and $\Lambda \doteq \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & \Gamma \end{bmatrix}$.

Let $\vec{u}$ be a unit-norm eigenvector of A corresponding to eigenvalue λ. Remember that we want an orthonormal matrix $V \in \mathbb{R}^{n \times n}$ which "isolates" λ. This motivates using $\vec{u}$ and a basis of the orthogonal complement of $\operatorname{span}(\vec{u})$ to form V. To construct this matrix V, we run Gram-Schmidt on the columns of the matrix $\begin{bmatrix} \vec{u} & I \end{bmatrix} \in \mathbb{R}^{n \times (n+1)}$, throwing out the single vector which will have 0 projection residual (there must be exactly one such vector by a counting argument; to get n linearly independent vectors from a spanning set of $n+1$ vectors, we need to remove exactly one vector), and obtaining the orthonormal matrix $V = \begin{bmatrix} \vec{u} & V_1 \end{bmatrix} \in \mathbb{R}^{n \times n}$ where $V_1 \in \mathbb{R}^{n \times (n-1)}$ is itself orthonormal. By construction, we have $V_1^\top \vec{u} = \vec{0}$ and $\vec{u}^\top V_1 = \vec{0}^\top$. Thus,

$$V^\top A V = \begin{bmatrix} \vec{u} & V_1 \end{bmatrix}^\top A \begin{bmatrix} \vec{u} & V_1 \end{bmatrix} \tag{2.82}$$

$$= \begin{bmatrix} \vec{u}^\top \\ V_1^\top \end{bmatrix} A \begin{bmatrix} \vec{u} & V_1 \end{bmatrix} \tag{2.83}$$

$$= \begin{bmatrix} \vec{u}^\top \\ V_1^\top \end{bmatrix} \begin{bmatrix} A\vec{u} & AV_1 \end{bmatrix} \tag{2.84}$$

$$= \begin{bmatrix} \vec{u}^\top \\ V_1^\top \end{bmatrix} \begin{bmatrix} \lambda\vec{u} & AV_1 \end{bmatrix} \tag{2.85}$$

$$= \begin{bmatrix} \lambda\vec{u}^\top\vec{u} & \vec{u}^\top A V_1 \\ \lambda V_1^\top \vec{u} & V_1^\top A V_1 \end{bmatrix} \tag{2.86}$$

$$= \begin{bmatrix} \lambda \|\vec{u}\|_2^2 & (A^\top \vec{u})^\top V_1 \\ \vec{0} & V_1^\top A V_1 \end{bmatrix} \tag{2.87}$$

$$= \begin{bmatrix} \lambda & (A\vec{u})^\top V_1 \\ \vec{0} & V_1^\top A V_1 \end{bmatrix} \tag{2.88}$$

$$= \begin{bmatrix} \lambda & \lambda \vec{u}^\top V_1 \\ \vec{0} & V_1^\top A V_1 \end{bmatrix} \tag{2.89}$$

$$= \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & V_1^\top A V_1 \end{bmatrix} \tag{2.90}$$

$$= \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & B \end{bmatrix}, \tag{2.91}$$

where $B \doteq V_1^\top A V_1$ in accordance with our proof outline. Now we need to check that B is symmetric; indeed, we have

$$B^\top = (V_1^\top A V_1)^\top \tag{2.92}$$

$$= (V_1)^\top A^\top (V_1^\top)^\top \tag{2.93}$$

$$= V_1^\top A V_1 \tag{2.94}$$

$$= B. \tag{2.95}$$

By induction, we can orthonormally diagonalize this matrix as $B = W\Gamma W^\top \in \mathbb{R}^{(n-1)\times(n-1)}$, where $W \in \mathbb{R}^{(n-1)\times(n-1)}$ is orthonormal and $\Gamma \in \mathbb{R}^{(n-1)\times(n-1)}$ is diagonal. Thus, by using $W^{-1} = W^\top$, we have

$$\Gamma = W^\top B W \tag{2.96}$$

$$= W^\top V_1^\top A V_1 W \tag{2.97}$$

$$= (V_1 W)^\top A (V_1 W). \tag{2.98}$$

We want an orthonormal matrix $U \in \mathbb{R}^{n\times n}$ such that $U^\top A U = \Lambda = \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & \Gamma \end{bmatrix}$. Thus, the above calculation motivates the choice $U = \begin{bmatrix} \vec{u} & V_1 W \end{bmatrix} \in \mathbb{R}^{n\times n}$. Thus

$$U^\top A U = \begin{bmatrix} \vec{u} & V_1 W \end{bmatrix}^\top A \begin{bmatrix} \vec{u} & V_1 W \end{bmatrix} \tag{2.99}$$

$$= \begin{bmatrix} \vec{u}^\top \\ W^\top V_1^\top \end{bmatrix} A \begin{bmatrix} \vec{u} & V_1 W \end{bmatrix} \tag{2.100}$$

$$= \begin{bmatrix} \vec{u}^\top \\ W^\top V_1^\top \end{bmatrix} \begin{bmatrix} A\vec{u} & A V_1 W \end{bmatrix} \tag{2.101}$$

$$= \begin{bmatrix} \vec{u}^\top \\ W^\top V_1^\top \end{bmatrix} \begin{bmatrix} \lambda\vec{u} & A V_1 W \end{bmatrix} \tag{2.102}$$

$$= \begin{bmatrix} \lambda \vec{u}^\top \vec{u} & \vec{u}^\top A V_1 W \\ \lambda W^\top V_1^\top \vec{u} & W^\top V_1^\top A V_1 W \end{bmatrix} \tag{2.103}$$

$$= \begin{bmatrix} \lambda \|\vec{u}\|_2^2 & (A^\top \vec{u})^\top V_1 W \\ \lambda W^\top \vec{0} & W^\top B W \end{bmatrix} \tag{2.104}$$

$$= \begin{bmatrix} \lambda & (A\vec{u})^\top V_1 W \\ \vec{0} & \Gamma \end{bmatrix} \tag{2.105}$$

$$= \begin{bmatrix} \lambda & \lambda \vec{u}^\top V_1 W \\ \vec{0} & \Gamma \end{bmatrix} \tag{2.106}$$

$$= \begin{bmatrix} \lambda & \lambda \vec{0}^\top W \\ \vec{0} & \Gamma \end{bmatrix} \tag{2.107}$$

$$= \begin{bmatrix} \lambda & \vec{0}^\top \\ \vec{0} & \Gamma \end{bmatrix} \tag{2.108}$$

$$= \Lambda, \tag{2.109}$$

as desired. Thus $A = U\Lambda U^\top$ is an orthonormal diagonalization of A. This proves (d), and hence (c). □

One nice thing about diagonalization is that we can read off the eigenvalues and eigenvectors from the components of the diagonalization.

Proposition 29
Let $A \in \mathbb{S}^n$ have orthonormal diagonalization $A = U\Lambda U^\top$, where $U = \begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_n \end{bmatrix} \in \mathbb{R}^{n\times n}$ is square orthonormal, and $\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n\times n}$ is diagonal. Then for each i, the pair $(\lambda_i, \vec{u}_i)$ is an eigenvalue-eigenvector pair for A.

Proof. By using $U^\top = U^{-1}$, we have

$$A = U\Lambda U^\top \tag{2.110}$$

$$AU = U\Lambda \tag{2.111}$$

$$A\begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_n \end{bmatrix} = \begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_n \end{bmatrix}\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \tag{2.112}$$

$$\begin{bmatrix} A\vec{u}_1 & \cdots & A\vec{u}_n \end{bmatrix} = \begin{bmatrix} \lambda_1\vec{u}_1 & \cdots & \lambda_n\vec{u}_n \end{bmatrix}. \tag{2.113}$$

Therefore, each $(\lambda_i, \vec{u}_i)$ is an eigenvalue-eigenvector pair of A. □

Using this, we can work with another nice property of the orthonormal diagonalization. Namely, we can read off bases for $\mathcal{N}(A)$ and $\mathcal{R}(A)$. That is, a basis for $\mathcal{N}(A)$ is the set of eigenvectors $\vec{u}_i$ corresponding to the eigenvalues λ_i of A which are equal to 0. Since U is orthonormal, the remaining eigenvectors $\vec{u}_i$ span the orthogonal complement to $\mathcal{N}(A)$. But by the fundamental theorem of linear algebra (Theorem 16), we have $\mathcal{N}(A)^\perp = \mathcal{R}(A^\top) = \mathcal{R}(A)$, so these eigenvectors form a basis for $\mathcal{R}(A)$. Soon, we'll discover the singular value decomposition, which allows for this kind of decomposition of a matrix into its range and null spaces, except for arbitrary matrices.

Before we get into those, we will first state and solve a quick optimization problem which yields the eigenvalues of a symmetric matrix. This optimization problem turns out to be quite useful for further study of optimization.

Theorem 30 (Variational Characterization of Eigenvalues)
Let $A \in \mathbb{S}^n$. Let $\lambda_{\min}\{A\}$ and $\lambda_{\max}\{A\}$ be the maximum and minimum eigenvalues of A (which is well-defined since by the spectral theorem, all eigenvalues of A are real). Then

$$\lambda_{\max}\{A\} = \max_{\substack{\vec{x}\in\mathbb{R}^n \\ \vec{x}\neq\vec{0}}} \frac{\vec{x}^\top A\vec{x}}{\vec{x}^\top\vec{x}} = \max_{\substack{\vec{x}\in\mathbb{R}^n \\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x} \tag{2.114}$$

$$\lambda_{\min}\{A\} = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \vec{x}\neq\vec{0}}} \frac{\vec{x}^\top A\vec{x}}{\vec{x}^\top\vec{x}} = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x}. \tag{2.115}$$

The term $\frac{\vec{x}^\top A\vec{x}}{\vec{x}^\top \vec{x}}$ is called the *Rayleigh quotient* of A; it is a function of $\vec{x} \in \mathbb{R}^n$.

Proof. Before we start trying to prove any equalities, let us try to simplify the crucial term $\vec{x}^\top A\vec{x}$; the intuition behind this is that it looks like the easiest term to use the orthonormal diagonalization on and achieve results.

Let $A = U\Lambda U^\top$ be an orthonormal diagonalization of A. We have

$$\begin{aligned}
\vec{x}^\top A\vec{x} &= \vec{x}^\top U\Lambda U^\top \vec{x} && (2.116)\\
&= (U^\top \vec{x})^\top \Lambda (U^\top \vec{x}) && (2.117)\\
&= \vec{y}^\top \Lambda \vec{y} && (2.118)\\
&= \sum_{i=1}^{n} \lambda_i\{A\} y_i^2 && (2.119)
\end{aligned}$$

with the invertible change of variables $\vec{y} \doteq U^\top \vec{x} \iff \vec{x} = U\vec{y}$. Also we note that this change of variables preserves the norm, i.e.,

$$\|\vec{y}\|_2^2 = \left\|U^\top \vec{x}\right\|_2^2 = \vec{x}^\top U U^\top \vec{x} = \vec{x}^\top \vec{x} = \|\vec{x}\|_2^2 . \quad (2.120)$$

We now turn to the first equality chain (with $\max$). Immediately, we have

$$\begin{aligned}
\max_{\substack{\vec{x}\in\mathbb{R}^n\\ \vec{x}\neq\vec{0}}} \frac{\vec{x}^\top A\vec{x}}{\vec{x}^\top \vec{x}} &= \max_{\substack{\vec{x}\in\mathbb{R}^n\\ \vec{x}\neq\vec{0}}} \frac{\vec{x}^\top A\vec{x}}{\|\vec{x}\|_2^2} && (2.121)\\
&= \max_{\substack{\vec{x}\in\mathbb{R}^n\\ \vec{x}\neq\vec{0}}} \left(\frac{\vec{x}}{\|\vec{x}\|_2}\right)^\top A \left(\frac{\vec{x}}{\|\vec{x}\|_2}\right). && (2.122)
\end{aligned}$$

Now, because the norm of our optimization variable $\vec{x}$ does not matter, in that it only affects the objective through its normalization $\vec{x}/\|\vec{x}\|_2$, it is equivalent to optimize over only unit-norm $\vec{x}$, so

$$\max_{\substack{\vec{x}\in\mathbb{R}^n\\ \vec{x}\neq\vec{0}}} \frac{\vec{x}^\top A\vec{x}}{\vec{x}^\top \vec{x}} = \max_{\substack{\vec{x}\in\mathbb{R}^n\\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x}. \quad (2.123)$$

With the invertible change of variables $\vec{y} = U^\top \vec{x}$ already discussed, we can write

$$\begin{aligned}
\max_{\substack{\vec{x}\in\mathbb{R}^n\\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x} &= \max_{\substack{\vec{y}\in\mathbb{R}^n\\ \|\vec{y}\|_2=1}} \sum_{i=1}^{n} \lambda_i y_i^2 && (2.124)\\
&\leq \lambda_{\max}\{A\} \cdot \max_{\substack{\vec{y}\in\mathbb{R}^n\\ \|\vec{y}\|_2=1}} \sum_{i=1}^{n} y_i^2 && (2.125)\\
&= \lambda_{\max}\{A\} \cdot \max_{\substack{\vec{y}\in\mathbb{R}^n\\ \|\vec{y}\|_2=1}} \|\vec{y}\|_2^2 && (2.126)\\
&= \lambda_{\max}\{A\} \cdot \max_{\substack{\vec{y}\in\mathbb{R}^n\\ \|\vec{y}\|_2=1}} 1 && (2.127)\\
&= \lambda_{\max}\{A\}. && (2.128)
\end{aligned}$$

It is left to exhibit a $\vec{y}$ which makes this inequality an equality; indeed, it is achieved when $y_i = 1$ for one i such that $\lambda_i\{A\} = \lambda_{\max}\{A\}$ and $y_i = 0$ otherwise. The achieving $\vec{x}$ can be recovered by $\vec{x} = U\vec{y}$. Note that since this $\vec{y}$ is a standard basis vector $\vec{y} = \vec{e}_i$ for some i such that $\lambda_i\{A\} = \lambda_{\max}\{A\}$, then $\vec{x} = U\vec{y} = U\vec{e}_i = \vec{u}_i$, i.e., the i^{th} column of U, is an eigenvector of A corresponding to the maximum eigenvalue of A.

The analysis for $\lambda_{\min}\{A\}$ goes exactly analogously. □

This characterization motivates defining a new sub-class (or really several new sub-classes) of matrices.

Definition 31 (Positive Semidefinite and Positive Definite Matrices)
Let $A \in \mathbb{S}^n$. We say that A is *positive semidefinite* (PSD), denoted $A \in \mathbb{S}^n_+$, if $\vec{x}^\top A\vec{x} \geq 0$ for all $\vec{x}$. We say that A is *positive definite* (PD), denoted $A \in \mathbb{S}^n_{++}$, if $\vec{x}^\top A\vec{x} > 0$ for all nonzero $\vec{x}$.

There are also negative semidefinite (NSD) and negative definite (ND) symmetric matrices, defined analogously. There are also indefinite symmetric matrices, which are none of the above. It is clear to see that PD matrices are themselves PSD.

Proposition 32
We have $A \in \mathbb{S}^n_+$ if and only if each eigenvalue of A is non-negative. Also, $A \in \mathbb{S}^n_{++}$ if and only if each eigenvalue of A is positive.

Proof. If A is PSD, we have

$$\lambda_{\min}\{A\} = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x} \geq 0. \tag{2.129}$$

Now suppose that each eigenvalue of A is non-negative. Then

$$0 \leq \lambda_{\min}\{A\} = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x} \tag{2.130}$$

which implies that $\vec{x}^\top A\vec{x} \geq 0$ for all $\vec{x}$ with unit norm, and by scaling we see that $\vec{x}^\top A\vec{x} \geq 0$ for all $\vec{x} \neq \vec{0}$, while the inequality certainly holds for $\vec{x} = \vec{0}$. Thus $\vec{x}^\top A\vec{x} \geq 0$ for all $\vec{x}$ so $A \in \mathbb{S}^n_+$.

If A is PD, we have

$$\lambda_{\min}\{A\} = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x} > 0. \tag{2.131}$$

Now suppose that each eigenvalue of A is positive. Then

$$0 < \lambda_{\min}\{A\} = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \|\vec{x}\|_2=1}} \vec{x}^\top A\vec{x} \tag{2.132}$$

which implies that $\vec{x}^\top A\vec{x} > 0$ for all $\vec{x}$ with unit norm, and by scaling we see that $\vec{x}^\top A\vec{x} > 0$ for all $\vec{x} \neq \vec{0}$, so $A \in \mathbb{S}^n_{++}$. □

The final construction we discuss is that of the *positive semidefinite square root*.

Proposition 33
Let $A \in \mathbb{S}^n_+$. Then there exists a unique symmetric PSD matrix $B \in \mathbb{S}^n_+$, usually denoted $B = A^{1/2}$, such that $A = B^2$.

Proof. Discussion or homework. Note that there are non-symmetric matrices B such that $A = B^2$, but there is a unique PSD B. □

2.5 Principal Component Analysis

Principal components analysis is a way to recover the eponymous principal components of the data. These principal components are those that are most representative of the data structure. Formally, if we have data in $\mathbb{R}^d$, we want to find an underlying p-dimensional linear structure, where $p \ll d$.

This idea has many, many use cases. For example, in modern machine learning, most data has thousands or millions of dimensions. In order to visualize it properly, we need to reduce its dimension to a reasonable number, in order to get an idea about the underlying structure of the data.

Let us first lay out some notation and definitions. Suppose we have the data points $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$. We organize these into a *data matrix* X where data points form the *rows*:[1]

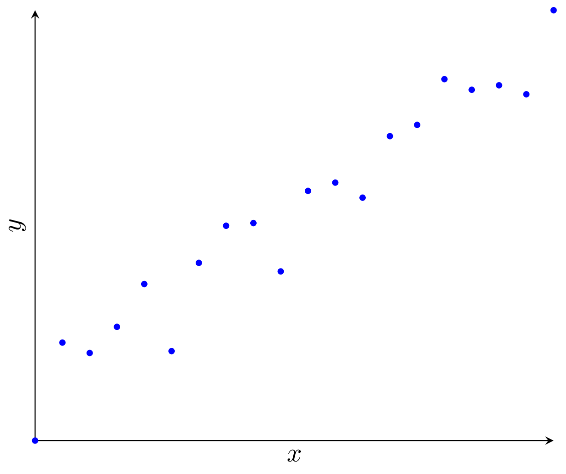
$$X \doteq \begin{bmatrix} \vec{x}_1^\top \\ \vdots \\ \vec{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d} \qquad \text{so that} \qquad X^\top = \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix} \in \mathbb{R}^{d \times n}. \tag{2.133}$$

We define the *covariance* matrix $C \in \mathbb{R}^{d \times d}$ by

$$C \doteq \frac{1}{n} X^\top X = \frac{1}{n} \begin{bmatrix} \vec{x}_1 & \cdots & \vec{x}_n \end{bmatrix} \begin{bmatrix} \vec{x}_1^\top \\ \vdots \\ \vec{x}_n^\top \end{bmatrix} = \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i \vec{x}_i^\top. \tag{2.134}$$

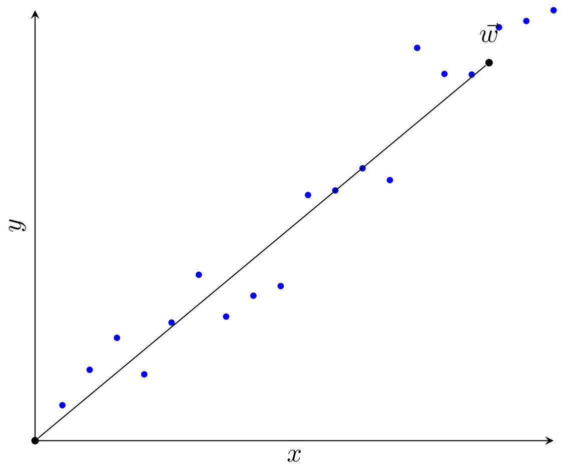
We see that C is symmetric since $X^\top X$ is symmetric, so really $C \in \mathbb{S}^d$.

Before we progress further, let us try to get some intuition for what we might want to be doing. Consider the case $d = 2$ and $p = 1$. Suppose we have the dataset given as below.



While this dataset is clearly fully two-dimensional, there is equally clearly some inherent 1-dimensional linear structure to the data. So when we want to look for an underlying low-dimensional structure, we're looking for something like this. Here, if we could find the direction $\vec{w}$ as in below:

[1]Different textbooks handle this differently. For instance, some textbooks define a data matrix as one where the data points form the columns. If you're unsure, work it out from first principles.



And, if we are in generic $\mathbb{R}^d$ space, we want to find orthonormal vectors $\vec{w}_1, \ldots, \vec{w}_p$ such that projection onto them uncovers the underlying data structure. The process of accurately characterizing these $\vec{w}_i$ is what we will discuss in what follows.

We begin with a motivating example. Consider the MNIST dataset of handwritten digits. Each image is a 28-pixel by 28-pixel grid with each numerical entry in the grid denoting the greyscale value at that grid point. This can be represented by a 28×28 size matrix, or alternatively unrolled into a $28^2 = 784$-dimensional vector. It is impossible to directly visualize 784-dimensional space, so we seek to find $\vec{w}_1, \ldots, \vec{w}_8 \in \mathbb{R}^{784}$ such that the projection onto the $\vec{w}_i$ preserve a lot of structure. Say that we take $\vec{w}_i = \vec{e}_i$, where $\vec{e}_i$ is the i^{th} standard basis vector in $\mathbb{R}^{784}$. Then for most images, the projection onto the $\vec{w}_i$'s will be 0 or near-0. Thus, the projection of all of the data onto the $\vec{w}_i$ preserves almost none of the structure and collapses all points in the dataset to just a few points in $\mathbb{R}^8$. There is instead a much more principled way to choose the $\vec{w}_i$ that will preserve most of the structure.

We now discuss how to choose the first principal component $\vec{w}_1 \in \mathbb{R}^d$. To preserve the structure of the underlying data as much as possible, we want the vectors $\vec{x}_i$ projected onto the span of $\vec{w}_1$ to be as close as possible to the original vectors $\vec{x}_i$. We also want $\|\vec{w}_1\|_2 = 1$. Thus, the error of the projection across all data points is

$$\operatorname{err}(\vec{w}_1) = \frac{1}{n}\sum_{i=1}^{n} \left\|\vec{x}_i - \vec{w}_1(\vec{w}_1^\top \vec{x}_i)\right\|_2^2 .$$

Expanding, we have

$$\begin{aligned}
\operatorname{err}(\vec{w}_1) &= \frac{1}{n}\sum_{i=1}^{n} \left\|\vec{x}_i - \vec{w}_1(\vec{w}_1^\top \vec{x}_i)\right\|_2^2 && (2.135)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\vec{x}_i - \vec{w}_1(\vec{w}_1^\top \vec{x}_i))^\top(\vec{x}_i - \vec{w}_1(\vec{w}_1^\top \vec{x}_i)) && (2.136)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\vec{x}_i^\top \vec{x}_i - \vec{x}_i^\top \vec{w}_1(\vec{w}_1^\top \vec{x}_i) - (\vec{w}_1(\vec{w}_1^\top \vec{x}_i))^\top \vec{x}_i + (\vec{w}_1(\vec{w}_1^\top \vec{x}_i))^\top(\vec{w}_1(\vec{w}_1^\top \vec{x}_i))) && (2.137)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\vec{x}_i^\top \vec{x}_i - 2(\vec{x}_i^\top \vec{w}_1)^2 + (\vec{w}_1^\top \vec{w}_1)(\vec{w}_1^\top \vec{x}_i)^2) && (2.138)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\|\vec{x}_i\|_2^2 - 2(\vec{x}_i^\top \vec{w}_1)^2 + (\vec{w}_1^\top \vec{x}_i)^2) && (2.139)\\
&= \frac{1}{n}\sum_{i=1}^{n} (\|\vec{x}_i\|_2^2 - (\vec{x}_i^\top \vec{w}_1)^2). && (2.140)
\end{aligned}$$

Now solving the principal components optimization problem gives

$$\min_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \operatorname{err}(\vec{w}_1) = \min_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \frac{1}{n}\sum_{i=1}^{n}(\|\vec{x}_i\|_2^2 - (\vec{x}_i^\top \vec{w}_1)^2) \tag{2.141}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \min_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \frac{1}{n}\sum_{i=1}^{n}(-(\vec{x}_i^\top \vec{w}_1)^2) \tag{2.142}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \max_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \frac{1}{n}\sum_{i=1}^{n}(\vec{x}_i^\top \vec{w}_1)^2 \tag{2.143}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \max_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \frac{1}{n}\sum_{i=1}^{n}\vec{w}_1^\top \vec{x}_i \vec{x}_i^\top \vec{w}_1 \tag{2.144}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \max_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \vec{w}_1^\top \left(\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i \vec{x}_i^\top\right)\vec{w}_1 \tag{2.145}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \max_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \vec{w}_1^\top \left(\frac{1}{n}X^\top X\right)\vec{w}_1 \tag{2.146}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \max_{\substack{\vec{w}_1 \in \mathbb{R}^d \\ \|\vec{w}_1\|_2 = 1}} \vec{w}_1^\top C \vec{w}_1 \tag{2.147}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x}_i\|_2^2 + \lambda_{\max}\{C\} \tag{2.148}$$

with the $\vec{w}_1$ achieving this upper bound being the eigenvector $\vec{u}_{\max}$ corresponding to the eigenvalue $\lambda_{\max}\{C\}$. Thus, the first principal component is exactly an eigenvector corresponding to the largest eigenvalue of the dot product matrix $C = X^\top X/n$.

This computation is a special case of the singular value decomposition, which is used in practice to compute the PCA of a dataset; understanding this decomposition will allow us to neatly compute the other principal components (i.e., second, third, fourth,...), as well.

2.6 Singular Value Decomposition

Definition 34 (SVD)

Let $A \in \mathbb{R}^{m \times n}$ have rank r. A *singular value decomposition (SVD)* of A is a decomposition of the form

$$A = U\Sigma V^\top \tag{2.149}$$

$$= \begin{bmatrix} U_r & U_{m-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} \begin{bmatrix} V_r^\top \\ V_{n-r}^\top \end{bmatrix} \tag{2.150}$$

$$= U_r \Sigma_r V_r^\top \tag{2.151}$$

$$= \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^\top, \tag{2.152}$$

where:

- $U \in \mathbb{R}^{m\times m}$, $U_r \in \mathbb{R}^{m\times r}$, $U_{m-r} \in \mathbb{R}^{m\times(m-r)}$, $V \in \mathbb{R}^{n\times n}$, $V_r \in \mathbb{R}^{n\times r}$, and $V_{n-r} \in \mathbb{R}^{n\times(n-r)}$ are orthonormal matrices, where $U = \begin{bmatrix} U_r & U_{m-r}\end{bmatrix}$ has columns $\vec{u}_1, \dots, \vec{u}_m$ (*left singular vectors*) and $V = \begin{bmatrix} V_r & V_{n-r}\end{bmatrix}$ has columns $\vec{v}_1, \dots, \vec{v}_n$ (*right singular vectors*).
- $\Sigma_r = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \in \mathbb{R}^{r\times r}$ is a diagonal matrix with ordered positive entries $\sigma_1 \geq \cdots \geq \sigma_r > 0$ (*singular values*), and the zero matrices in the $\Sigma = \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix}$ matrix are shaped to ensure that $\Sigma \in \mathbb{R}^{m\times n}$.

Suppose that A is tall (so $m > n$) with full column rank n. Then the SVD looks like the following:

$$A = U \begin{bmatrix} \Sigma_n \\ 0_{(m-n)\times n} \end{bmatrix} V^\top. \tag{2.153}$$

On the other hand, if A is wide (so $m < n$) with full row rank m, then the SVD looks like the following:

$$A = U \begin{bmatrix} \Sigma_m & 0_{m\times(n-m)} \end{bmatrix} V^\top. \tag{2.154}$$

The last (summation) form of the SVD is called the *dyadic* SVD; this is because terms of the form $\vec{p}\vec{q}^\top$ are called *dyads*, and the dyadic SVD expresses the matrix A as the sum of dyads.

All forms of the SVD are useful conceptually and computationally, depending on the problem we are working on.

We now discuss a method to construct the SVD. Suppose $A \in \mathbb{R}^{m\times n}$ has rank r. We consider the symmetric matrix $A^\top A$ which has rank r and thus r nonzero eigenvalues, which are positive. We can order its eigenvalues as $\lambda_1 \geq \cdots \geq \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = 0$, say with corresponding orthonormal eigenvectors $\vec{v}_1, \dots, \vec{v}_n$.

Then, for $i \in \{1, \dots, r\}$, we define $\sigma_i \doteq \sqrt{\lambda_i} > 0$, and $\vec{u}_i \doteq A\vec{v}_i/\sigma_i$. This only gives us r vectors $\vec{u}_i$, but we need m of them to construct $U \in \mathbb{R}^{m\times m}$. To find the remaining $\vec{u}_i$ we use Gram-Schmidt on the matrix $\begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_r & I \end{bmatrix} \in \mathbb{R}^{m\times(r+m)}$, throwing out the r vectors whose projection residual onto previously processed vectors is 0.

More formally, we can write an algorithm:

Algorithm 2 Construction of the SVD.

```
function SVD($A \in \mathbb{R}^{m\times n}$)
    $r \doteq \mathrm{rank}(A)$
    $(\lambda_1, \vec{v}_1), \dots, (\lambda_n, \vec{v}_n) \doteq \textsc{Eigenpairs}(A^\top A)$    ▷ $\lambda_1 \geq \cdots \geq \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = 0$, and $\vec{v}_i$ orthonormal.
    for $i \in \{1, \dots, r\}$ do
        $\sigma_i \doteq \sqrt{\lambda_i}$    ▷ $\sigma_i > 0$
        $\vec{u}_i \doteq A\vec{v}_i/\sigma_i$
    end for
    $\vec{u}_1, \dots, \vec{u}_r, \vec{u}_{r+1}, \dots, \vec{u}_m \doteq \textsc{GramSchmidt}(\begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_r & I \end{bmatrix})$
    return $\{\vec{u}_1, \dots, \vec{u}_m\}, \{\sigma_1, \dots, \sigma_r\}, \{\vec{v}_1, \dots, \vec{v}_n\}$
end function
```

It's clear that Algorithm 2 gives an orthonormal basis $\{\vec{v}_1, \dots, \vec{v}_n\}$ for $\mathbb{R}^n$ that can be constructed into the orthonormal V matrix, and that it gives singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$. We aim to show two things: the

$\{\vec{u}_1, \ldots, \vec{u}_m\}$ are orthonormal, and that $A = U\Sigma V^\top$ where U, Σ, and V are constructed using the returned vectors and scalars.

Proposition 35
In the context of Algorithm 2, $\{\vec{u}_1, \ldots, \vec{u}_m\}$ is an orthonormal set.

Proof. From our invocation of Gram-Schmidt, $\{\vec{u}_{r+1}, \ldots, \vec{u}_m\}$ is an orthonormal set which spans an orthogonal subspace to the span of $\{\vec{u}_1, \ldots, \vec{u}_r\}$. Thus, we need to show that $\{\vec{u}_1, \ldots, \vec{u}_r\}$ are orthonormal.

Indeed, take $1 \leq i < j \leq r$. Then since the $\vec{v}_j$ are orthonormal eigenvectors of $A^\top A$, we have

$$\vec{u}_i^\top \vec{u}_j = \left(\frac{A\vec{v}_i}{\sigma_i}\right)^\top \left(\frac{A\vec{v}_j}{\sigma_j}\right) \tag{2.155}$$
$$= \frac{\vec{v}_i^\top A^\top A\vec{v}_j}{\sigma_i\sigma_j} \tag{2.156}$$
$$= \frac{\lambda_j \vec{v}_i^\top \vec{v}_j}{\sigma_i\sigma_j} \tag{2.157}$$
$$= \frac{\lambda_j}{\sigma_i\sigma_j} \underbrace{\vec{v}_i^\top \vec{v}_j}_{=0} \tag{2.158}$$
$$= 0. \tag{2.159}$$

On the other hand, for a specific $i \in \{1, \ldots, r\}$, using that $\sigma_i^2 = \lambda_i$, we have

$$\|\vec{u}_i\|_2^2 = \left\|\frac{A\vec{v}_i}{\sigma_i}\right\|_2^2 \tag{2.160}$$
$$= \left(\frac{A\vec{v}_i}{\sigma_i}\right)^\top \left(\frac{A\vec{v}_i}{\sigma_i}\right) \tag{2.161}$$
$$= \frac{\vec{v}_i^\top A^\top A\vec{v}_i}{\sigma_i^2} \tag{2.162}$$
$$= \frac{\lambda_i \vec{v}_i^\top \vec{v}_i}{\sigma_i^2} \tag{2.163}$$
$$= \underbrace{\frac{\lambda_i}{\sigma_i^2}}_{=1} \underbrace{\vec{v}_i^\top \vec{v}_i}_{=1} \tag{2.164}$$
$$= 1. \tag{2.165}$$

Thus the set $\{\vec{u}_1, \ldots, \vec{u}_r\}$ is orthonormal, so the whole set $\{\vec{u}_1, \ldots, \vec{u}_m\}$ is orthonormal. □

Proposition 36
In the context of Algorithm 2, we have $A = U\Sigma V^\top$.

Proof. By construction, we have

$$A\vec{v}_i = \sigma_i \vec{u}_i \qquad \text{for all } i \in \{1, \ldots, r\}, \tag{2.166}$$
$$A\vec{v}_i = \vec{0} \qquad \text{for all } i \in \{r+1, \ldots, m\}, \tag{2.167}$$

This gives us

$$\begin{aligned}
AV &= A\begin{bmatrix}\vec{v}_1 & \cdots & \vec{v}_r & \vec{v}_{r+1} & \cdots & \vec{v}_n\end{bmatrix} && (2.168)\\
&= \begin{bmatrix}A\vec{v}_1 & \cdots & A\vec{v}_r & A\vec{v}_{r+1} & \cdots & A\vec{v}_n\end{bmatrix} && (2.169)\\
&= \begin{bmatrix}\sigma_1\vec{u}_1 & \cdots & \sigma_r\vec{u}_r & \vec{0} & \cdots & \vec{0}\end{bmatrix} && (2.170)\\
&= \begin{bmatrix}U_r\Sigma_r & 0\end{bmatrix} && (2.171)
\end{aligned}$$

On the other hand, we have

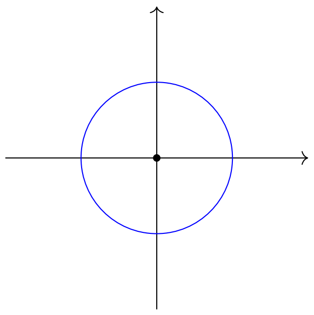
$$\begin{aligned}
U\Sigma &= \begin{bmatrix}U_r & U_{m-r}\end{bmatrix}\begin{bmatrix}\Sigma_r & 0\\ 0 & 0\end{bmatrix} && (2.172)\\
&= \begin{bmatrix}U_r\Sigma_r + U_{m-r}\cdot 0 & U_r\cdot 0 + U_{m-r}\cdot 0\end{bmatrix} && (2.173)\\
&= \begin{bmatrix}U_r\Sigma_r & 0\end{bmatrix}. && (2.174)
\end{aligned}$$

Thus $AV = U\Sigma$. Since V is orthonormal, $V^\top = V^{-1}$, so $A = U\Sigma V^\top$. □

The SVD is not unique: the Gram-Schmidt process could have used any basis for $\mathbb{R}^m$ that wasn't the columns of I and still have been valid; if you had multiple eigenvectors of $A^\top A$ with the same eigenvalue then the choice of eigenvectors in the diagonalization would not be unique; and even if you didn't have multiple eigenvectors with the same eigenvalue, the eigenvectors would only be determined up to a sign change $\vec{v} \mapsto -\vec{v}$ anyways. So there is a lot of ambiguity in the construction, which reflects the non-uniqueness.

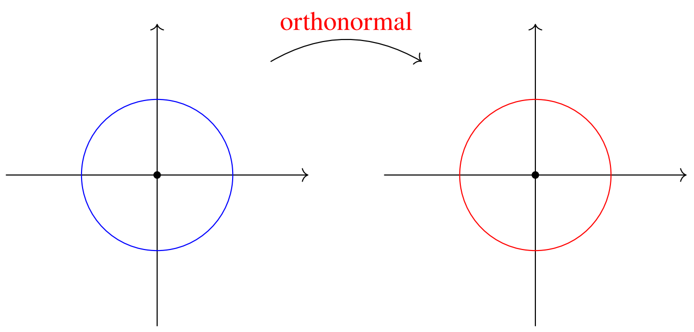
We now discuss the geometry of the SVD, especially how each component of the SVD acts on vectors. To do this we will fix $A \in \mathbb{R}^{2\times 2}$ with SVD $A = U\Sigma V^\top$, and find the behavior of $A\vec{x}$ for all $\vec{x}$ on the unit circle (i.e., with norm 1). We will analyze the behavior of $A\vec{x}$ by using the behavior of $V^\top\vec{x}$, $\Sigma V^\top\vec{x}$ and finally $U\Sigma V^\top\vec{x}$. In the end, we will interpret U as a *rotation or reflection*, Σ as a *scaling*, and $V^\top$ as another *rotation or reflection*.

Before we start, let us discuss what different types of matrices look like as linear transformations. Consider our friendly unit circle:

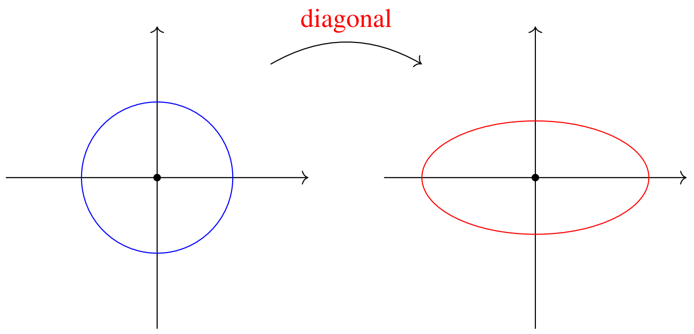


This unit circle consists of all vectors in $\mathbb{R}^2$ with norm 1.

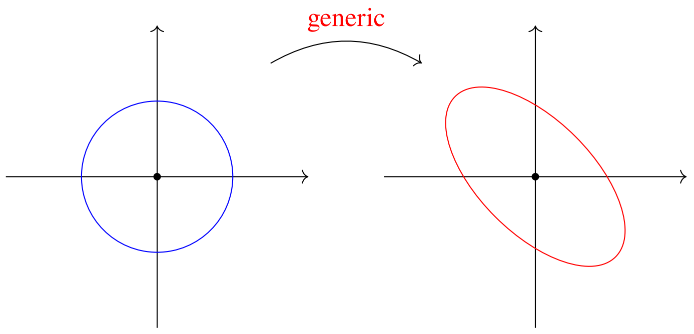
Multiplying each vector in that circle by the same orthonormal matrix will not change the norm of any vector in that circle, and thus will amount to nothing more than a rotation of the circle, thus not changing its shape.



On the other hand, multiplying each vector in the unit circle by the same diagonal matrix will scale the vectors in the coordinate directions. This means that the unit circle will be mapped, in general, to an axis-aligned ellipse.

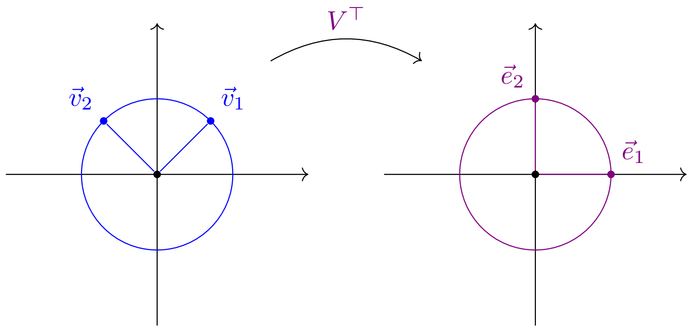


In general, matrices aren't orthonormal or diagonal, and so they will both rotate and scale in various ways. This means that the unit circle will be mapped to an ellipse which isn't necessarily axis-aligned.

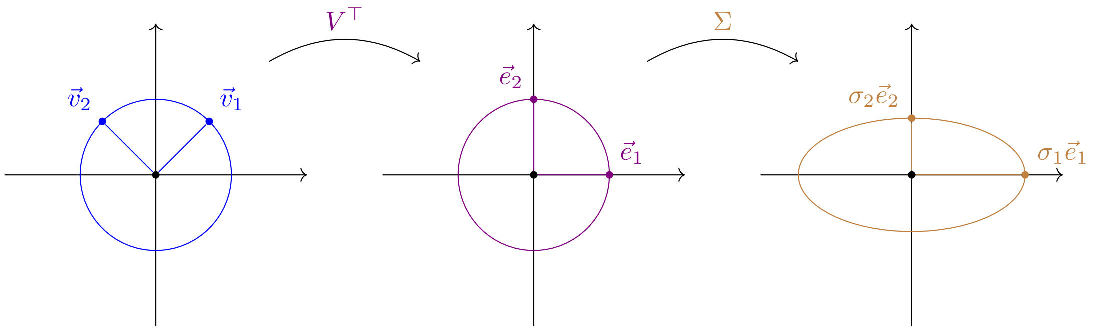


Let us now systematically study such $A = U\Sigma V^{\top}$ through the lens of the unit circle, as well as where the right singular vectors $\vec{v}_1$ and $\vec{v}_2$ are mapped by A.

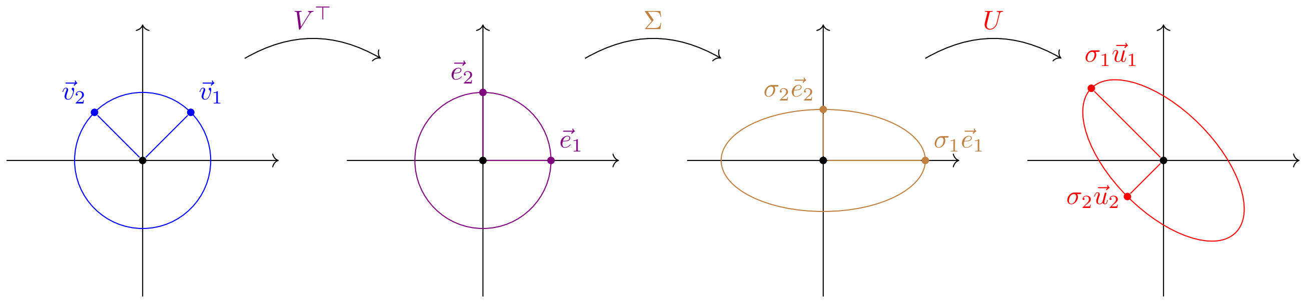
Since $V^{\top}$ is an orthonormal matrix, it represents a rotation and/or reflection, and so it maps the unit circle to the unit circle, much like our observed figure. Specifically, the right singular vectors $\vec{v}_i$ have $V^{\top}\vec{v}_i = \vec{e}_i$, so they get mapped onto the standard basis by $V^{\top}$. This gives the following picture.



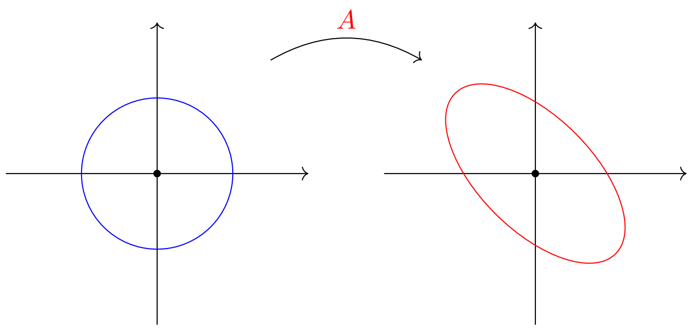
Now, the diagonal matrix Σ will scale each $\vec{e}_i$ by σ_i, obtaining an ellipse.



Finally, the orthonormal matrix U will map this axis-aligned ellipse to an ellipse which isn't necessarily axis-aligned. Specifically, the vectors that we're looking at, i.e., $\sigma_i\vec{e}_i$, have $U(\sigma_i\vec{e}_i) = \sigma_i U\vec{e}_i = \sigma_i\vec{u}_i$. These $\vec{u}_i$ will be the axes of the resulting ellipse in the same sense as $\sigma_i\vec{e}_i$ were the axes of the axis-aligned ellipse.



Recall that we originally started with a depiction that didn't have any fine-grained description of any vectors, yet obtained the same result:



To understand the impact of A on any general vector $\vec{x}$, we write it in the V basis: $\vec{x} = \alpha_1\vec{v}_1 + \alpha_2\vec{v}_2$, and use linearity to obtain $A\vec{x} = \alpha_1\sigma_1\vec{u}_1 + \alpha_2\sigma_2\vec{u}_2$. One can draw this graphically using scaled versions of the above ellipses.

This perspective also says that σ_1 is the maximum scaling of any vector obtained by multiplication by A, and σ_r is the minimum nonzero scaling. (If $r < n$, i.e., A is not full column rank, then there are some nonzero vectors in $\mathbb{R}^n$ which are sent to $\vec{0}$ by A, so the minimum scaling is 0.) You will formally prove this in homework.

2.7 Low-Rank Approximation

Sometimes, in real datasets, matrices have billions or trillions of entries. Storing all of them would be prohibitively expensive, and we would need a way to compress them down into their most important parts. This turns out to be doable via the SVD, as we will see.

To formally talk about a compression algorithm that stores a compressed version of the data with minimal error, we need to talk about what kind of errors are appropriate to discuss in the context of matrices. In the case of vectors, we can use the ℓ^2-norm, or more generally the ℓ^p norm, to define a distance function; then, the error would just be the distance between the true and the perturbed vectors. This motivates thinking about matrix norms, which allow us to quantify the distance between matrices, and thus create error functions.

There are two ways to think about a matrix. The first way is as a *block of numbers*. Similarly to how we thought of a vector as a block of numbers and found the norm based on this, we can think of the matrix as a big list of vectors and take the norm. This norm is called the *Frobenius norm*, and it corresponds to unrolling an $m \times n$ matrix into a length $m \cdot n$ vector and taking its ℓ^2-norm.

Definition 37 (Frobenius Norm)
For a matrix $A \in \mathbb{R}^{m\times n}$, its Frobenius norm is defined as

$$\|A\|_F \doteq \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}. \tag{2.175}$$

The following property will help us when we study vector calculus.

Proposition 38
For a matrix $A \in \mathbb{R}^{m\times n}$, we have $\|A\|_F^2 = \operatorname{tr}\left(A^\top A\right)$.

The following pair of results will help us now.

Proposition 39
For a matrix $A \in \mathbb{R}^{m\times n}$ and orthonormal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, we have

$$\|UAV\|_F = \|UA\|_F = \|AV\|_F = \|A\|_F \,. \tag{2.176}$$

We do not prove this property now, though it might be on homework.

Proposition 40

For a matrix $A \in \mathbb{R}^{m \times n}$ with rank r and singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$, we have

$$\|A\|_F^2 = \sum_{i=1}^{r} \sigma_i^2. \tag{2.177}$$

Proof. Let $A = U\Sigma V^\top$ be the SVD of A; then, we use the previous proposition to get

$$\begin{aligned} \|A\|_F^2 &= \left\|U\Sigma V^\top\right\|_F^2 && (2.178)\\ &= \|\Sigma\|_F^2 && (2.179)\\ &= \sum_{i=1}^{r} \sigma_i^2. && (2.180) \end{aligned}$$

$\square$

Under this perspective, $\|A - B\|_F$ is small if each component of A and B is close; that is, they are very similar in terms of the block-of-numbers interpretation.

The second way to think about a matrix is as a *linear transformation*. In this case, the matrix is defined by how it acts on vectors via multiplication. A suitable notion of size in this case is the largest scaling factor of the matrix on any unit vector; this is called the *spectral norm* or the *matrix ℓ^2-norm*.

Definition 41 (Spectral Norm)

For a matrix $A \in \mathbb{R}^{m \times n}$, its spectral norm is defined by

$$\|A\|_2 \doteq \max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 = 1}} \|A\vec{x}\|_2 . \tag{2.181}$$

Fortunately, this optimization problem has a solution — what's more, we've actually seen this solution before.

Proposition 42

For a matrix $A \in \mathbb{R}^{m \times n}$ with rank r and singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$, we have

$$\|A\|_2 = \sigma_1. \tag{2.182}$$

Proof. We use the Rayleigh quotient to say that

$$\begin{aligned} \|A\|_2 &\doteq \max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 = 1}} \|A\vec{x}\|_2 && (2.183)\\ &= \max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 = 1}} \sqrt{\|A\vec{x}\|_2^2} && (2.184)\\ &= \sqrt{\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 = 1}} \|A\vec{x}\|_2^2} && (2.185)\\ &= \sqrt{\max_{\substack{\vec{x} \in \mathbb{R}^n \\ \|\vec{x}\|_2 = 1}} \vec{x}^\top A^\top A\vec{x}} && (2.186)\\ &= \sqrt{\lambda_{\max}\{A^\top A\}} && (2.187)\\ &= \sigma_1. && (2.188) \end{aligned}$$

□

Connecting back to the ellipse transformations, the spectral norm captures how much the ellipse is stretched in its most stretched direction. Thus, the matrix is viewed as as a linear map.

To present our main theorems about how to approximate the matrix well under these norms, we need to define notation. Fix a matrix $A \in \mathbb{R}^{m \times n}$. For convenience, let $p \doteq \min\{m, n\}$. Suppose that A has rank $r \leq p$, and that A has SVD

$$A = \sum_{i=1}^{p} \sigma_i \vec{u}_i \vec{v}_i^\top \tag{2.189}$$

where we note that $\sigma_1 \geq \cdots \geq \sigma_r$ and define $\sigma_{r+1} = \sigma_{r+2} = \cdots = 0$. Then, for $k \leq p$, we can define

$$A_k \doteq \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top. \tag{2.190}$$

Note that if $k \ll p$, then A_k can be stored much more efficiently than A. For instance, A_k needs to store k scalars (σs), k vectors in $\mathbb{R}^m$ ($\vec{u}$s), and k vectors in $\mathbb{R}^n$ ($\vec{v}$s), for a total storage of $k(m + n + 1)$ floats. On the other hand, storing A requires mn floats naively, and even storing the (full) SVD of A is not much better. So the former is much more efficient to store. The only thing left to do is to show that it actually is a good approximation in A (otherwise what would be the point to storing it instead of A?)

It turns out that A_k indeed well-approximates A in the sense of the two norms — the Frobenius norm and the spectral norm — that we discussed previously. The two results are collectively known as the *Eckart-Young* (sometimes *Eckart-Young-Mirsky*) theorem(s). We state and prove these now.

We begin with the Eckart-Young theorem for the spectral norm, since it will help us prove the analogous result for Frobenius norms.

Theorem 43 (Eckart-Young Theorem for Spectral Norm)
We have

$$A_k \in \operatorname*{argmin}_{\substack{B \in \mathbb{R}^{m \times n} \\ \operatorname{rank}(B) \leq k}} \|A - B\|_2, \tag{2.191}$$

or, equivalently,

$$\|A - A_k\|_2 \leq \|A - B\|_2, \qquad \forall B \in \mathbb{R}^{m \times n} \colon \operatorname{rank}(B) \leq k. \tag{2.192}$$

Proof of Theorem 43. The proof of Eckart-Young Theorem for Spectral Norm is partitioned into two parts which together jointly show the conclusion. First, we explicitly calculate $\|A - A_k\|_2$; then for an arbitrary B of rank $\leq k$ we show that $\|A - B\|_2$ is lower-bounded by this quantity.

Part 1. First, we calculate $\|A - A_k\|_2$. Indeed, we have

$$A - A_k = \left(\sum_{i=1}^{p} \sigma_i \vec{u}_i \vec{v}_i^\top\right) - \left(\sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top\right) \tag{2.193}$$

$$= \sum_{i=k+1}^{p} \sigma_i \vec{u}_i \vec{v}_i^\top. \tag{2.194}$$

Since $\sigma_{k+1} \geq \sigma_{k+2} \geq \cdots$, the $\vec{u}_i$ are orthonormal, and the $\vec{v}_i$ are orthonormal, this is a valid singular value decomposition of the rank $r - k$ matrix $A - A_k$. Thus $A - A_k$ has largest singular value equal to σ_{k+1}, so

$$\|A - A_k\|_2 = \sigma_{k+1}. \tag{2.195}$$

Part 2. We now show that for any matrix $B \in \mathbb{R}^{m \times n}$ of rank $\leq k$, we have $\|A - B\|_2 \geq \sigma_{k+1}$. Before we commence with the proof, let us first discuss the overall argument structure.

We have two characterizations of the spectral norm: the maximum singular value, and the maximal Rayleigh coefficient. Up until now, we don't have any advanced machinery such as singular value inequalities to directly show that the maximum singular value of $A - B$ is lower-bounded by some constant. However, we do understand matrix-vector products, and how to characterize their optima, pretty well, so this motivates the use of the maximal Rayleigh coefficient. In particular, suppose $f(\vec{x}) = \|(A - B)\vec{x}\|_2 / \|\vec{x}\|_2$. Then we know that for any specific value of $\vec{x}_0$, we have $\|A - B\|_2 = \max_{\vec{x}} f(\vec{x}) \geq f(\vec{x}_0)$. Thus, one way to get a lower bound on $\|A - B\|_2$ is to plug in any $\vec{x}_0$ into f. The trick is, given B, to find the right value of $\vec{x}_0$ such that $f(\vec{x}_0) = \sigma_{k+1}$. The first part of the argument will be finding an appropriate $\vec{x}_0$; the next will show that it works.

Okay, let's start with the proof.

Step 1. Here we show the existence of an appropriate choice for $\vec{x}_0$.

Because B has rank $\leq k$, the rank-nullity theorem gives

$$\dim(\mathcal{N}(B)) = n - \operatorname{rank}(B) \geq n - k > 0. \tag{2.196}$$

Now, define $V_{k+1} \doteq \begin{bmatrix} \vec{v}_1, \ldots, \vec{v}_{k+1} \end{bmatrix}$. We know by the SVD that the $\vec{v}_i$ are orthonormal, and so are linearly independent. Thus

$$\operatorname{rank}(V_{k+1}) = \dim(\mathcal{R}(V_{k+1})) = k + 1. \tag{2.197}$$

Thus

$$\dim(\mathcal{N}(B)) + \dim(\mathcal{R}(V_{k+1})) \geq (n - k) + (k + 1) = n + 1 > n. \tag{2.198}$$

Claim. There exists a unit vector in $\mathcal{N}(B) \cap \mathcal{R}(V_{k+1})$.

Proof. First, we note that since $\mathcal{N}(B)$ and $\mathcal{R}(V_{k+1})$ are subspaces of $\mathbb{R}^n$, their intersection is a subspace of $\mathbb{R}^n$. Thus, it is closed under scalar multiplication. Thus the existence of a unit vector is equivalent to the existence of a nonzero vector, since one can scale this nonzero vector by its (inverse) norm to get the unit vector.

Suppose for the sake of contradiction that $\mathcal{N}(B)$ and $\mathcal{R}(V_{k+1})$ have no nonzero vectors in common. Fix a basis S_1 for $\mathcal{N}(B)$ and a basis S_2 for $\mathcal{R}(V_{k+1})$. By assumption, we have that $S_1 \cap S_2 = \emptyset$, and $S_1 \cup S_2$ is a linearly independent set. Here, there are two cases.

A. $S_1 \cup S_2$ contains $< n + 1$ vectors. This means that S_1 and S_2 have a vector in common, so $S_1 \cap S_2 \neq \emptyset$ and we have reached a contradiction.

B. $S_1 \cup S_2$ has exactly $n + 1$ vectors. As a set of $n + 1$ vectors in $\mathbb{R}^n$, $S_1 \cup S_2$ is linearly dependent and we have reached a contradiction.

In both cases we have reached a contradiction, so there is a nonzero vector in $\mathcal{N}(B) \cap \mathcal{R}(V_{k+1})$. We may scale this vector by its inverse norm to get a unit vector in $\mathcal{N}(B) \cap \mathcal{R}(V_{k+1})$. □

Let $\vec{x}_0 \in \mathcal{N}(B) \cap \mathcal{R}(V_{k+1})$ be any such unit vector. We use this vector for the Rayleigh coefficient.

Step 2. Now we show that our choice of $\vec{x}_0$ plugged into the Rayleigh coefficient gives σ_{k+1} as a lower bound.

Indeed, we have

$$\|A - B\|_2 = \max_{\vec{x} \neq \vec{0}} \frac{\|(A - B)\vec{x}\|_2}{\|\vec{x}\|_2} \tag{2.199}$$

$$\geq \frac{\|(A-B)\vec{x}_0\|_2}{\|\vec{x}_0\|_2} \tag{2.200}$$

$$= \|(A-B)\vec{x}_0\|_2 \tag{2.201}$$

$$= \|A\vec{x}_0 - B\vec{x}_0\|_2 . \tag{2.202}$$

Since $\vec{x}_0 \in \mathcal{N}(B)$, we have $B\vec{x}_0 = \vec{0}$, so

$$\|A-B\|_2 \geq \|A\vec{x}_0 - B\vec{x}_0\|_2 = \|A\vec{x}_0\|_2 . \tag{2.203}$$

Since $\vec{x}_0 \in \mathcal{R}(V_{k+1})$, there are constants $\alpha_1, \dots, \alpha_{k+1}$, not all zero, such that $\vec{x}_0 = \sum_{i=1}^{k+1} \alpha_i \vec{v}_i$. Thus

$$\|A-B\|_2 \geq \|A\vec{x}_0\|_2 \tag{2.204}$$

$$= \left\| A \sum_{i=1}^{k+1} \alpha_i \vec{v}_i \right\|_2 \tag{2.205}$$

$$= \left\| \left(\sum_{i=1}^{p} \sigma_i \vec{u}_i \vec{v}_i^\top \right) \left(\sum_{i=1}^{k+1} \alpha_i \vec{v}_i \right) \right\|_2 \tag{2.206}$$

$$= \left\| \sum_{i=1}^{p} \sum_{j=1}^{k+1} \sigma_i \alpha_j \vec{u}_i \vec{v}_i^\top \vec{v}_j \right\|_2 . \tag{2.207}$$

By vector algebra, the fact that the $\vec{u}_i$ are orthonormal, and the fact that the $\vec{v}_i$ are orthonormal, one can mechanically show that

$$\|A-B\|_2 \geq \left\| \sum_{i=1}^{p} \sum_{j=1}^{k+1} \sigma_i \alpha_j \vec{u}_i \vec{v}_i^\top \vec{v}_j \right\|_2 \tag{2.208}$$

$$= \left\| \sum_{i=1}^{k+1} \alpha_i \sigma_i \vec{u}_i \right\|_2 \tag{2.209}$$

$$= \sqrt{\left\| \sum_{i=1}^{k+1} \alpha_i \sigma_i \vec{u}_i \right\|_2^2} \tag{2.210}$$

$$= \sqrt{\sum_{i=1}^{k+1} \alpha_i^2 \sigma_i^2} \tag{2.211}$$

$$\geq \sqrt{\sigma_{k+1}^2 \sum_{i=1}^{k+1} \alpha_i^2} \tag{2.212}$$

$$= \sigma_{k+1} \sqrt{\sum_{i=1}^{k+1} \alpha_i^2} \tag{2.213}$$

$$= \sigma_{k+1} \|\vec{x}_0\|_2 \tag{2.214}$$

$$= \sigma_{k+1}. \tag{2.215}$$

This proves the claim.

□

We now state and prove the other result, which is the Eckart-Young theorem for the Frobenius norm.

Theorem 44 (Eckart-Young Theorem for Frobenius Norm)
We have

$$A_k \in \operatorname*{argmin}_{\substack{B \in \mathbb{R}^{m \times n} \\ \operatorname{rank}(B) \leq k}} \|A - B\|_F\,, \tag{2.216}$$

or, equivalently,

$$\|A - A_k\|_F^2 \leq \|A - B\|_F^2\,, \qquad \forall B \in \mathbb{R}^{m \times n} \colon \operatorname{rank}(B) \leq k. \tag{2.217}$$

Proof of Theorem 44. Like the Frobenius norm, the proof of Eckart-Young Theorem for Frobenius Norm is partitioned into two parts which together jointly show the conclusion. First, we explicitly calculate $\|A - A_k\|_F^2$; then for an arbitrary B of rank $\leq k$ we show that $\|A - B\|_F^2$ is lower-bounded by this quantity.

Part 1. First, we calculate $\|A - A_k\|_F^2$. Indeed, we have

$$\begin{aligned} A - A_k &= \left(\sum_{i=1}^{p} \sigma_i \vec{u}_i \vec{v}_i^\top\right) - \left(\sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^\top\right) && (2.218)\\ &= \sum_{i=k+1}^{p} \sigma_i \vec{u}_i \vec{v}_i^\top. && (2.219)\end{aligned}$$

Since $\sigma_{k+1} \geq \sigma_{k+2} \geq \cdots$, the $\vec{u}_i$ are orthonormal, and the $\vec{v}_i$ are orthonormal, this is a valid singular value decomposition of the rank $r - k$ matrix $A - A_k$. Thus $A - A_k$ has largest singular value equal to σ_{k+1}, so

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{p} \sigma_i^2. \tag{2.220}$$

Part 2. We now show that for any matrix $B \in \mathbb{R}^{m \times n}$ of rank $\leq k$, we have $\|A - B\|_F^2 \geq \sum_{i=k+1}^{p} \sigma_i^2$. Again, let us begin by discussing the overall argument structure.

Our goal is to show an inequality between two sums of squares of singular values. To do this, it is simplest to match up terms in the two sums and show the inequality between each pair of terms. Then summing over all terms preserves the inequality.

More concretely, define $C \doteq A - B$. Suppose C has SVD $C = \sum_{i=1}^{p} \gamma_i \vec{y}_i \vec{z}_i^\top$. We want to show

$$\|A - B\|_F^2 = \sum_{i=1}^{p} \gamma_i^2 \underbrace{\geq}_{\text{still need to show}} \sum_{i=k+1}^{p} \sigma_i^2. \tag{2.221}$$

Thus, we claim that $\gamma_i \geq \sigma_{i+k}$ for every $i \in \{1, \ldots, p\}$, with the understanding that $\sigma_{p+1} = \sigma_{p+2} = \cdots = 0$.

To use our earlier knowledge about the spectral norm to our advantage, we write the singular value γ_i as the spectral norm of the approximation error matrix:

$$\gamma_i = \|C - C_{i-1}\|_2\,. \tag{2.222}$$

We can expand to get

$$\gamma_i = \|C - C_{i-1}\|_2 = \|(A - B) - C_{i-1}\|_2 = \|A - (B + C_{i-1})\|_2\,. \tag{2.223}$$

Now this looks somewhat like the low-rank approximation setup, because we are computing the spectral norm of the difference between A and some matrix. But what is the rank of this matrix? We have

$$\operatorname{rank}(B + C_{i-1}) \leq \operatorname{rank}(B) + \operatorname{rank}(C_{i-1}) \leq k + i - 1. \tag{2.224}$$

Thus, by applying Eckart-Young Theorem for Spectral Norm to the rank $i+k-1$ approximation of A, that

$$\gamma_i = \|A - (B + C_{i-1})\|_2 \geq \|A - A_{i+k-1}\|_2 = \sigma_{i+k} \tag{2.225}$$

which is what we wanted to show.

To finish the proof, we use the following inequalities:

$$\begin{aligned} \gamma_i^2 &\geq \sigma_{i+k}^2, \qquad \forall i \in \{1, \ldots, p\} && \quad (2.226)\\ \sum_{i=1}^{p} \gamma_i^2 &\geq \sum_{i=1}^{p} \sigma_{i+k}^2 && \text{summing over all } i, \quad (2.227)\\ \|A - B\|_F^2 &\geq \sum_{i=k+1}^{p} \sigma_i^2 && \quad (2.228) \end{aligned}$$

as desired.

□

Chapter 3

Vector Calculus

Relevant sections of the textbooks:

- [1] Appendix A.4.

3.1 Gradient, Jacobian, and Hessian

To motivative this section, we start with a familiar concept which should have been covered in the prerequisites: the derivatives of a *scalar* function $f\colon \mathbb{R} \to \mathbb{R}$ which takes in scalar input and produces a scalar output. The derivative quantifies the (instantaneous) rate of change of the function due to the change of its input. We recall the limit definition of the derivative.

Definition 45 (Derivative for Scalar Functions)
Let $f\colon \mathbb{R} \to \mathbb{R}$ be differentiable. The *derivative* of f with respect to x is the function $\frac{\mathrm{d}f}{\mathrm{d}x}\colon \mathbb{R} \to \mathbb{R}$ defined by

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}. \tag{3.1}$$

In this section, we aim to generalize the concept of derivatives beyond scalar functions. We will focus on two types of functions:

1. *Multivariate functions* $f\colon \mathbb{R}^n \to \mathbb{R}$ which take a vector $\vec{x} \in \mathbb{R}^n$ as input and produce a scalar $f(\vec{x}) \in \mathbb{R}$ as output. Familiar examples of such functions include $f(\vec{x}) = \|\vec{x}\|_p$ and $f(\vec{x}) = \vec{a}^\top \vec{x}$.
2. *Vector-valued functions* $\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^m$ which take a vector $\vec{x} \in \mathbb{R}^n$ as input and produce another vector $\vec{f}(\vec{x}) \in \mathbb{R}^m$ as output. A familiar example of such functions is $f(\vec{x}) = A\vec{x}$.

One tool that allows us to compute derivatives of scalar functions is the chain rule, which describes the derivative of the composition of two functions.

Theorem 46 (Chain Rule for Scalar Functions)
Let $f\colon \mathbb{R} \to \mathbb{R}$ and $g\colon \mathbb{R} \to \mathbb{R}$ be two differentiable scalar functions, and let $h\colon \mathbb{R} \to \mathbb{R}$ be defined as

$h(x) = f(g(x))$ for all $x \in \mathbb{R}$. Then h is differentiable, and

$$\frac{\mathrm{d}h}{\mathrm{d}x}(x) = \frac{\mathrm{d}f}{\mathrm{d}x}(g(x)) \cdot \frac{\mathrm{d}g}{\mathrm{d}x}(x). \tag{3.2}$$

We will see in the next section that the chain rule can be generalized to settings of multivariate functions.

3.1.1 Partial Derivatives

For multivariate functions $f\colon \mathbb{R}^n \to \mathbb{R}$, when we talk about the rate of change of the function with respect to its input, we need to specify which input we are talking about. Partial derivatives quantify this and give us the rate of change of the function due to the change of one of its inputs, say x_i, while keeping all other inputs fixed. Keeping all but one input fixed renders a scalar function, for which we know how to compute the derivative. To formalize this we introduce the limit definition of the partial derivative.

Definition 47 (Partial Derivative)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be differentiable. The *partial derivative* of f with respect to x_i is the function $\frac{\partial f}{\partial x_i}\colon \mathbb{R}^n \to \mathbb{R}$ defined by

$$\frac{\partial f}{\partial x_i}(\vec{x}) = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(\vec{x})}{h}, \tag{3.3}$$

or equivalently,

$$\frac{\partial f}{\partial x_i}(\vec{x}) = \lim_{h \to 0} \frac{f(\vec{x} + h \cdot \vec{e}_i) - f(\vec{x})}{h} \tag{3.4}$$

where $\vec{e}_i$ is the i^{th} standard basis vector.

This limit definition gives an alternative way of interpreting partial derivatives: $\frac{\partial f}{\partial x_i}$ is the rate of change of the function along the direction of the standard basis vector $\vec{e}_i$.

In practice, we do not use the limit definition to compute regular derivatives. Similarly, we do not use the limit definition to compute partial derivatives. The main way to compute partial derivatives uses the following tip.

Problem Solving Strategy 48. *To compute the partial derivative $\frac{\partial f}{\partial x_i}$, pretend that all x_j for $j \neq i$ are constants, then take the ordinary derivative in x_i.*

Example 49. Consider the function $f(\vec{x}) = \vec{a}^\top \vec{x}$. Then

$$\begin{aligned}
\frac{\partial f}{\partial x_i}(\vec{x}) &= \frac{\partial}{\partial x_i} \vec{a}^\top \vec{x} &(3.5)\\
&= \frac{\partial}{\partial x_i} \sum_{j=1}^{n} a_j x_j &(3.6)\\
&= \frac{\partial}{\partial x_i}(a_1 x_1 + \cdots + a_n x_n) &(3.7)\\
&= a_i. &(3.8)
\end{aligned}$$

Let us consider the case where the input $\vec{x}$ is not an independent vector but rather depends on another variable $t \in \mathbb{R}$, i.e., $\vec{x}\colon \mathbb{R} \to \mathbb{R}^n$ is a function. In such case the function $f(\vec{x}(t))$ has *one independent* input, which is t. If we are interested in finding the derivative of f with respect to t, we can utilize a chain rule to do so.

Theorem 50 (Chain Rule For Multivariate Functions)

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ and $\vec{x}\colon \mathbb{R} \to \mathbb{R}^n$ be differentiable functions. Define the function $g\colon \mathbb{R} \to \mathbb{R}$ by $g(t) = f(\vec{x}(t))$ for all $t \in \mathbb{R}$. Then g is differentiable and has derivative

$$\frac{\mathrm{d}g}{\mathrm{d}t}(t) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\vec{x}(t)) \cdot \frac{\mathrm{d}x_i}{\mathrm{d}t}(t). \tag{3.9}$$

3.1.2 Gradient

We will now use the definition of partial derivatives to introduce the gradient of multivariate functions.

Definition 51 (Gradient)

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. The *gradient* of f is the function $\nabla f\colon \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$\nabla f(\vec{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\vec{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\vec{x}) \end{bmatrix}. \tag{3.10}$$

Note that the gradient is a column vector. The transpose of the gradient is (confusingly!) referred to as the *derivative* of the function. We will now list two important geometric properties of the gradient. The first can be stated straight away:

Proposition 52

Let $\vec{x} \in \mathbb{R}^n$. The gradient $\nabla f(\vec{x})$ points in the direction of *steepest ascent* at $\vec{x}$, i.e., the direction around $\vec{x}$ in which f has the maximum rate of change. Furthermore, this rate of change is quantified by the norm $\|\nabla f(\vec{x})\|_2$.

To list the second property, first we need a quick definition.

Definition 53 (Level Set)

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a function, and $\alpha \in \mathbb{R}$ be a scalar.

- The α-*level set* of f is the set of points $\vec{x}$ such that $f(\vec{x}) = \alpha$:

$$L_\alpha(f) = \{\vec{x} \in \mathbb{R}^n \mid f(\vec{x}) = \alpha\}. \tag{3.11}$$

- The α-*sublevel set* of f is the set of points $\vec{x}$ such that $f(\vec{x}) \leq \alpha$:

$$L_{\leq\alpha}(f) = \{\vec{x} \in \mathbb{R}^n \mid f(\vec{x}) \leq \alpha\}. \tag{3.12}$$

- The α-*superlevel set* of f is the set of points $\vec{x}$ such that $f(\vec{x}) \geq \alpha$:

$$L_{\geq\alpha}(f) = \{\vec{x} \in \mathbb{R}^n \mid f(\vec{x}) \geq \alpha\}. \tag{3.13}$$

Proposition 54
Let $\vec{x} \in \mathbb{R}^n$ and suppose $f(\vec{x}) = \alpha$. Then $\nabla f(\vec{x})$ is orthogonal to the hyperplane which is tangent at $\vec{x}$ to the α-level set of f.

We illustrate the two properties through examples but do not cover their proofs. The proof of the first property will be a homework exercise.

Example 55 (Gradient of the Squared ℓ^2 Norm)**.** In this example we will compute and visualize the gradient of the function $f(\vec{x}) = \|\vec{x}\|_2^2$ where $\vec{x} \in \mathbb{R}^2$.

$$\begin{aligned}\nabla f(\vec{x}) &= \begin{bmatrix} \frac{\partial f}{\partial x_1}(\vec{x}) \\ \frac{\partial f}{\partial x_2}(\vec{x}) \end{bmatrix} && (3.14)\\ &= \begin{bmatrix} \frac{\partial}{\partial x_1}(x_1^2 + x_2^2) \\ \frac{\partial}{\partial x_2}(x_1^2 + x_2^2) \end{bmatrix} && (3.15)\\ &= \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} && (3.16)\\ &= 2\vec{x}. && (3.17)\end{aligned}$$

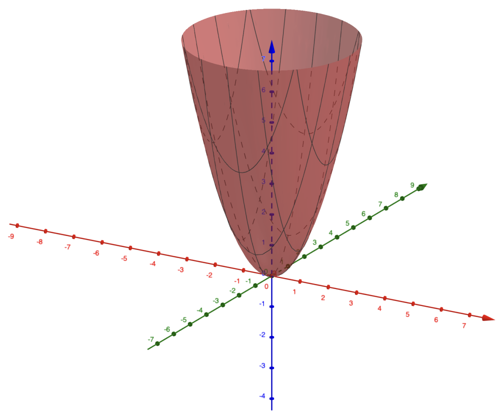
This function has a paraboloid-shaped graph, as shown in Figure 3.1a. Let us now find the α-level set of the function f for some constant α.

$$\begin{aligned}\alpha &= f(\vec{x}) && (3.18)\\ &= x_1^2 + x_2^2 && (3.19)\end{aligned}$$

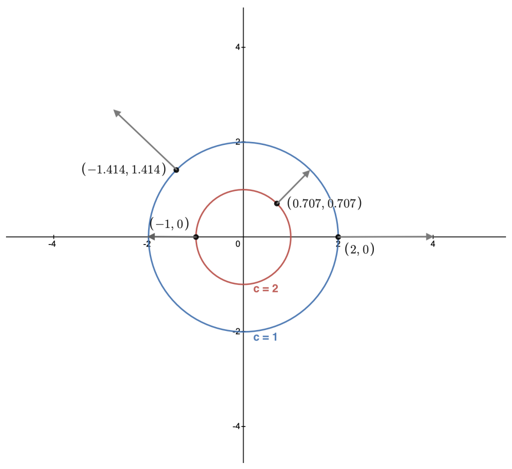
For a given $\alpha \geq 0$, the α-level set is a circle centered at the origin which has radius $\sqrt{\alpha}$. Now we evaluate the gradient at a few points on these level sets:

$$\nabla f(-1, 0) = (-2, 0) \tag{3.20}$$
$$\nabla f(1/\sqrt{2}, 1/\sqrt{2}) = (\sqrt{2}, \sqrt{2}) \tag{3.21}$$
$$\nabla f(2, 0) = (4, 0) \tag{3.22}$$
$$\nabla f(-\sqrt{2}, \sqrt{2}) = (-2\sqrt{2}, 2\sqrt{2}) \tag{3.23}$$

We plot the level sets for $\alpha = 1$ and $\alpha = 4$ and visualize the gradient directions in Figure 3.1b.



(a) Plot of the function $f(\vec{x}) = \|\vec{x}\|_2^2$



(b) The level sets of the function $f(\vec{x}) = \|\vec{x}\|_2^2 = \alpha$ for $\alpha = 1$ and $\alpha = 4$, and the gradients at some points along the level sets.

Figure 3.1: Example 55

We make the following observations:

1. At each point on a given level set, the gradient vector at that point is orthogonal to the line tangent to the level set at that point.
2. The length of the gradient vector increases as we move away from the origin. This means that the function gets steeper in that direction (i.e., it changes more rapidly).

Example 56 (Gradient of Linear Function)**.** In this example we will compute and comment on the gradient of the linear function $f\colon \mathbb{R}^n \to \mathbb{R}$ is defined by $f(\vec{x}) = \vec{a}^\top \vec{x}$ where $\vec{a} \in \mathbb{R}^n$ is fixed. We have

$$f(\vec{x}) = \vec{a}^\top \vec{x}, \tag{3.24}$$

$$\nabla f(\vec{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\vec{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\vec{x}) \end{bmatrix} \tag{3.25}$$

$$= \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \tag{3.26}$$

$$= \vec{a}. \tag{3.27}$$

We make the following observations:

1. The α-level sets of this function are the hyperplanes given by all $\vec{x}$ such that $\vec{a}^\top \vec{x} = \alpha$. These hyperplanes have normal vector $\vec{a}$. Notice that the normal vector (which is orthogonal to all vectors in the hyperplane) is exactly the gradient vector.
2. The gradient is constant, meaning that the function has a constant rate of change everywhere.

Example 57 (Gradient of the Quadratic Form)**.** Let $A \in \mathbb{R}^{n\times n}$. In this example we will compute the gradient of the quadratic function $f\colon \mathbb{R}^n \to \mathbb{R}$ defined by $f(\vec{x}) = \vec{x}^\top A\vec{x}$. Indeed, we have

$$\begin{aligned} f(\vec{x}) &= \vec{x}^\top A\vec{x} && (3.28)\\ &= \sum_{i=1}^n \sum_{j=1}^n A_{ij}x_ix_j. && (3.29) \end{aligned}$$

Now fix $k \in \{1,\dots,n\}$, and we find the partial derivative with respect to x_k. We have

$$\begin{aligned} f(\vec{x}) &= \sum_{i=1}^n \sum_{j=1}^n A_{ij}x_ix_j \\ &= \sum_{j=1}^n A_{kj}x_kx_j + \sum_{\substack{i=1\\ i\neq k}}^n \sum_{j=1}^n A_{ij}x_ix_j \\ &= \sum_{j=1}^n A_{kj}x_kx_j + \sum_{\substack{i=1\\ i\neq k}}^n A_{ik}x_ix_k + \sum_{\substack{i=1\\ i\neq k}}^n \sum_{\substack{j=1\\ j\neq k}}^n A_{ij}x_ix_j \\ &= A_{kk}x_k^2 + \sum_{\substack{j=1\\ j\neq k}}^n A_{kj}x_kx_j + \sum_{\substack{i=1\\ i\neq k}}^n A_{ik}x_ix_k + \sum_{\substack{i=1\\ i\neq k}}^n \sum_{\substack{j=1\\ j\neq k}}^n A_{ij}x_ix_j \\ &= A_{kk}x_k^2 + x_k\sum_{\substack{j=1\\ j\neq k}}^n A_{kj}x_j + x_k\sum_{\substack{i=1\\ i\neq k}}^n A_{ik}x_i + \sum_{\substack{i=1\\ i\neq k}}^n \sum_{\substack{j=1\\ j\neq k}}^n A_{ij}x_ix_j. \end{aligned}$$

Then, taking the derivatives, we have

$$\begin{aligned} \frac{\partial f}{\partial x_k}(\vec{x}) &= \frac{\partial}{\partial x_k}\left(A_{kk}x_k^2 + x_k\sum_{\substack{j=1\\ j\neq k}}^n A_{kj}x_j + x_k\sum_{\substack{i=1\\ i\neq k}}^n A_{ik}x_i + \sum_{\substack{i=1\\ i\neq k}}^n \sum_{\substack{j=1\\ j\neq k}}^n A_{ij}x_ix_j\right) && (3.30)\\ &= 2A_{kk}x_k + \sum_{\substack{j=1\\ j\neq k}}^n A_{kj}x_j + \sum_{\substack{i=1\\ i\neq k}}^n A_{ik}x_i && (3.31)\\ &= \left(A_{kk}x_k + \sum_{\substack{j=1\\ j\neq k}}^n A_{kj}x_j\right) + \left(A_{kk}x_k + \sum_{\substack{i=1\\ i\neq k}}^n A_{ik}x_i\right) && (3.32)\\ &= \sum_{j=1}^n A_{kj}x_j + \sum_{i=1}^n A_{ik}x_i && (3.33)\\ &= (A\vec{x})_k + (A^\top\vec{x})_k && (3.34)\\ &= ((A + A^\top)\vec{x})_k. && (3.35) \end{aligned}$$

Thus computing the gradient via stacking the partial derivatives gets

$$\nabla f(\vec{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\vec{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\vec{x}) \end{bmatrix} \tag{3.36}$$

$$= \begin{bmatrix} ((A + A^\top)\vec{x})_1 \\ \vdots \\ ((A + A^\top)\vec{x})_n \end{bmatrix} \tag{3.37}$$

$$= (A + A^\top)\vec{x}. \tag{3.38}$$

That was a very involved computation! Luckily, in the near future, we will see ways to simplify the process of computing gradients.

3.1.3 Jacobian

We now have the tools to generalize the notion of derivatives to vector-valued functions $\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^m$.

Definition 58 (Jacobian)
Let $\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable function. The *Jacobian* of $\vec{f}$ is the function $D\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^{m\times n}$ defined as

$$D\vec{f}(\vec{x}) = \begin{bmatrix} \nabla f_1(\vec{x})^\top \\ \vdots \\ \nabla f_m(\vec{x})^\top \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\vec{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\vec{x}) & \cdots & \frac{\partial f_m}{\partial x_n}(\vec{x}) \end{bmatrix}. \tag{3.39}$$

One big thing to note is that *the Jacobian is different from the gradient!* If $f\colon \mathbb{R}^n \to \mathbb{R}^1 = \mathbb{R}$, then its Jacobian $Df\colon \mathbb{R}^n \to \mathbb{R}^{1\times n}$ is a function which outputs a *row vector*. This row vector is the *transpose* of the gradient.

We can develop a nice and general chain rule with the Jacobian.

Theorem 59 (Chain Rule for Vector-Valued Functions)
Let $\vec{f}\colon \mathbb{R}^p \to \mathbb{R}^m$ and $\vec{g}\colon \mathbb{R}^n \to \mathbb{R}^p$ be differentiable functions. Let $\vec{h}\colon \mathbb{R}^n \to \mathbb{R}^m$ be defined as $\vec{h}(\vec{x}) = \vec{f}(\vec{g}(\vec{x}))$ for all $\vec{x} \in \mathbb{R}^n$. Then $\vec{h}$ is differentiable, and

$$D\vec{h}(\vec{x}) = [D\vec{f}(\vec{g}(\vec{x}))] \cdot [D\vec{g}(\vec{x})]. \tag{3.40}$$

Here, as before, the notation $D\vec{f}(\vec{g}(\vec{x}))$ means that we compute $D\vec{f}$ and then evaluate it on the point $\vec{g}(\vec{x})$.

This is a broad chain rule which we can apply to many problems, but we must always remember that *for a function $f\colon \mathbb{R}^n \to \mathbb{R}$, its Jacobian is the transpose of its gradient.* One typical chain rule we can derive from the general one follows below.

Corollary 60. *Let $f\colon \mathbb{R}^p \to \mathbb{R}$ and $\vec{g}\colon \mathbb{R}^n \to \mathbb{R}^p$ be differentiable functions. Let $h\colon \mathbb{R}^n \to \mathbb{R}$ be defined as $h(\vec{x}) = f(\vec{g}(\vec{x}))$ for all $\vec{x} \in \mathbb{R}^n$. Then h is differentiable, and*

$$\nabla h(\vec{x}) = [D\vec{g}(\vec{x})]^\top \nabla f(\vec{g}(\vec{x})). \tag{3.41}$$

Example 61. In this example we will use the chain rule to compute the gradient of the function $f(\vec{x}) = \|A\vec{x} - \vec{y}\|_2^2$. It can be written as $f(\vec{x}) = g(\vec{h}(\vec{x}))$ where $g(\vec{x}) = \|\vec{x}\|_2^2$ and $\vec{h}(\vec{x}) = A\vec{x} - \vec{y}$. We have that

$$D\vec{h}(\vec{x}) = A \tag{3.42}$$

(the proof is in discussion or homework), and also from earlier

$$\nabla g(\vec{x}) = 2\vec{x}. \tag{3.43}$$

Thus applying the chain rule obtains

$$\nabla f(\vec{x}) = [D\vec{h}(\vec{x})]^\top \nabla g(\vec{h}(\vec{x})) \tag{3.44}$$

$$= 2A^\top (A\vec{x} - \vec{y}) \tag{3.45}$$

as desired. This gradient matches what would be computed if we had used the componentwise partial derivatives.

3.1.4 Hessian

So far, we have appropriately generalized the notion of first derivative to vector-valued functions, i.e., functions $\vec{f} \colon \mathbb{R}^n \to \mathbb{R}^m$. We now turn to doing the same with second derivatives.

It turns out that for general vector-valued functions $\vec{f}$, defining a second derivative is possible, but such an object will live in $\mathbb{R}^{m\times n\times n}$ and thus not be hard to work with using the linear algebra we have discussed in this class. However, for multivariate functions $f \colon \mathbb{R}^n \to \mathbb{R}$, defining this second derivative as a particular matrix becomes possible; this matrix, called the *Hessian*, has great conceptual and computational importance.

Recall that to find the second derivative of a scalar-valued function, we merely take the derivative of the derivative. Our notion of the gradient suffices as a first derivative; to take the derivative of this, we need to use the Jacobian. Indeed, the Hessian is exactly the Jacobian of the gradient, and defined precisely below.

Definition 62 (Hessian)
Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be twice differentiable. The *Hessian* of f is the function $\nabla^2 f \colon \mathbb{R}^n \to \mathbb{R}^{n\times n}$ defined by

$$\nabla^2 f(\vec{x}) = D(\nabla f)(\vec{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\vec{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \, \partial x_1}(\vec{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \, \partial x_n}(\vec{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\vec{x}) \end{bmatrix}. \tag{3.46}$$

It turns out that under some mild conditions, the Hessian is symmetric; this is called Clairaut's theorem.

Theorem 63 (Clairaut's Theorem)
Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable, and fix $\vec{x} \in \mathbb{R}^n$. Then $\nabla^2 f(\vec{x})$ is a symmetric matrix, i.e., for every $1 \le i, j \le n$ we have

$$\frac{\partial^2 f}{\partial x_i \, \partial x_j}(\vec{x}) = \frac{\partial^2 f}{\partial x_j \, \partial x_i}(\vec{x}). \tag{3.47}$$

The vast majority of functions we work with in this course are twice continuously differentiable, with some notable exceptions (i.e., the ℓ^1 norm is not even once-differentiable). Thus, in most cases, the Hessian is symmetric.

Example 64 (Hessian of Squared ℓ^2 Norm)**.** In this example we will compute the Hessian of the function $f(\vec{x}) = \|\vec{x}\|_2^2$ where $\vec{x} \in \mathbb{R}^2$. Recall the gradient of this function computed in Example 55 as

$$\nabla f(\vec{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}. \tag{3.48}$$

The Hessian can then be computed as

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}. \tag{3.49}$$

3.2 Taylor's Theorems

In this section, we will introduce Taylor approximation and Taylor's theorem for the familiar scalar function case. Then we will generalize the idea of Taylor approximation to multivariate functions. Taylor approximation is a tool to find polynomial approximation of functions using information about the function value at a point along with the value of its firs, second and higher order derivatives.

Definition 65 (Taylor Approximation)
Let $f\colon \mathbb{R} \to \mathbb{R}$ be a k-times continuously differentiable function, and fix $x_0 \in \mathbb{R}$. The k^{th} degree Taylor approximation around x_0 is the function $\widehat{f}_k(\cdot; x_0)\colon \mathbb{R} \to \mathbb{R}$ given by

$$\widehat{f}_k(x; x_0) = f(x_0) + \frac{1}{1!}\frac{\mathrm{d}f}{\mathrm{d}x}(x_0) \cdot (x - x_0) + \cdots + \frac{1}{k!}\frac{\mathrm{d}^k f}{\mathrm{d}x^k}(x_0) \cdot (x - x_0)^k \tag{3.50}$$

$$= \sum_{i=0}^{k} \frac{1}{i!}\frac{\mathrm{d}^i f}{\mathrm{d}x^i}(x_0) \cdot (x - x_0)^i. \tag{3.51}$$

In particular, the first-order and second-order Taylor approximations of f around x_0 are

$$\widehat{f}_1(x; x_0) = f(x_0) + \frac{\mathrm{d}f}{\mathrm{d}x}(x_0) \cdot (x - x_0) \tag{3.52}$$

$$\widehat{f}_2(x; x_0) = f(x_0) + \frac{\mathrm{d}f}{\mathrm{d}x}(x_0) \cdot (x - x_0) + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_0) \cdot (x - x_0)^2. \tag{3.53}$$

We will derive multivariable versions of these approximations later.

Example 66 (Taylor Approximation of Cubic Function)**.** Let us approximate the function $f(x) = x^3$ around the fixed point $x_0 = 1$ using Taylor approximations of different degrees.

$$\widehat{f}_1(x; 1) = f(x_0) + \frac{\mathrm{d}f}{\mathrm{d}x}(x_0) \cdot (x - x_0) \tag{3.54}$$

$$= x_0^3 + 3x_0^2 \cdot (x - x_0) \tag{3.55}$$

$$= 1^3 + 3 \cdot 1^2 \cdot (x - 1) \tag{3.56}$$

$$= 3(x - 1) + 1 \tag{3.57}$$

$$= 3x - 2. \tag{3.58}$$

$$\widehat{f}_2(x; 1) = \widehat{f}_1(x; 1) + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_0) \cdot (x - x_0)^2 \tag{3.59}$$

$$= 3x - 2 + 3 \cdot 1 \cdot (x - 1)^2 \tag{3.60}$$

$$= 3x^2 - 3x + 1. \tag{3.61}$$

$$\widehat{f}_3(x; 1) = \widehat{f}_2(x; 1) + \frac{1}{6}\frac{\mathrm{d}^3 f}{\mathrm{d}x^3}(x_0) \cdot (x - x_0)^3 \tag{3.62}$$

$$= 3x^2 - 3x + 1 + (x - 1)^3 \tag{3.63}$$

$$= x^3. \tag{3.64}$$

We notice the following take-aways:

- The first-order Taylor approximation $\widehat{f}_1(\cdot; x_0)$ is the *best linear approximation* to f around $x = x_0 = 1$. In particular, its graph is the tangent line to the graph of f around the point $(x_0, f(x_0)) = (1, 1)$, as observed in Figure 3.2.

- The second-order Taylor approximation $\widehat{f}_2(\cdot; x_0)$ is the *best quadratic approximation* to f around $x = x_0 = 1$. It is the parabola whose graph passes through the point $(x_0, f(x_0)) = (1, 1)$, as observed in Figure 3.2, and it has the same first and second derivatives as f at x_0. Using the intuition that the second derivative models curvature, we see that the second-order Taylor approximation captures the local curvature of the graph of the function. This intuition will be helpful later when discussing convexity.
- The third-degree Taylor approximation $\widehat{f}_3(\cdot; x_0)$ is the *best cubic approximation* to f; because f is just a cubic function, the best cubic approximation is just f itself, and indeed we have $\widehat{f}_3(\cdot; x_0) = f$.

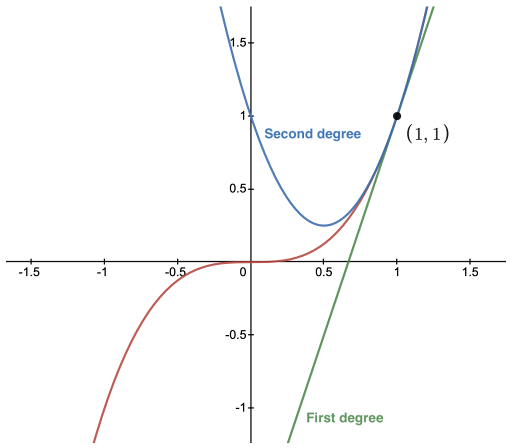


Figure 3.2: First and second degree Taylor approximations of the function $f(x) = x^3$.

Taylor approximation gives us the degree k polynomial that approximates the function $f(x)$ around the fixed point $x = x_0$. Taylor's theorem quantifies the bounds for the error of this approximation.

Theorem 67 (Taylor's Theorem)
Let $f: \mathbb{R} \to \mathbb{R}$ be a function which is k-times continuously differentiable, and fix $x_0 \in \mathbb{R}$. Then for all $x \in \mathbb{R}$ we have

$$f(x) = \widehat{f}_k(x; x_0) + o(|x - x_0|^k) \tag{3.65}$$

where the term $o(|x - x_0|^k)$ (i.e., the *remainder*) denotes a function, say $R_k(x; x_0)$, such that

$$\lim_{x \to x_0} \frac{R_k(x; x_0)}{|x - x_0|^k} = 0. \tag{3.66}$$

We use this remainder notation because we don't really care about what it is precisely, only its limiting behavior as $x \to x_0$, and the little-o notation allows us to not worry too much about the exact form of the remainder.

This theorem certifies that the Taylor approximations $\widehat{f}_k$ are good approximations to f. Another way to write this result is generally more useful or simpler:

$$f(x + \delta) = \underbrace{f(x) + \frac{\mathrm{d}f}{\mathrm{d}x}(x) \cdot \delta}_{= \widehat{f}_1(x + \delta; x)} + o(|\delta|) \tag{3.67}$$

$$= \underbrace{f(x) + \frac{\mathrm{d}f}{\mathrm{d}x}(x) \cdot \delta + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x) \cdot \delta^2}_{= \widehat{f}_2(x + \delta; x)} + o(\delta^2) \tag{3.68}$$

$$= \dots. \tag{3.69}$$

We will never need to quantitatively work with the remainder in this course; we will usually write $f \approx \widehat{f}_k$ and leave it at that.

3.2.1 Taylor Approximation of Multivariate Functions

Using the definitions we introduced for the gradient and Hessian of multivariate functions, we can generalize the idea of Taylor's approximation to these functions.

Definition 68 (Multivariate Taylor Approximations)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ and fix $\vec{x}_0 \in \mathbb{R}^n$.

- If f is continuously differentiable, then its first-order Taylor approximation around $\vec{x}_0$ is the function $\widehat{f}_1(\cdot; \vec{x}_0)\colon \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{f}_1(\vec{x}; \vec{x}_0) = f(\vec{x}_0) + [\nabla f(\vec{x}_0)]^\top (\vec{x} - \vec{x}_0). \tag{3.70}$$

- If f is twice continuously differentiable, then its second-order Taylor approximation around $\vec{x}_0$ is the function $\widehat{f}_2(\cdot; \vec{x}_0)\colon \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{f}_2(\vec{x}; \vec{x}_0) = f(\vec{x}_0) + [\nabla f(\vec{x}_0)]^\top (\vec{x} - \vec{x}_0) + \frac{1}{2}(\vec{x} - \vec{x}_0)^\top [\nabla^2 f(\vec{x}_0)](\vec{x} - \vec{x}_0). \tag{3.71}$$

The graph of the first-order Taylor approximation is the hyperplane tangent to the graph of f at the point $(\vec{x}_0, f(\vec{x}_0))$. This hyperplane has normal vector $\nabla f(\vec{x}_0)$.

We could define higher-order Taylor approximations $\widehat{f}_k$, but to express them concisely would require generalizations of matrices, called *tensors*. For example, the third derivative of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ is a rank-3 tensor, i.e., an object which lives in $\mathbb{R}^{n \times n \times n}$. These are out of scope for this course, and anyways we will only need the first two derivatives.

We can also state an analogous Taylor's theorem.

Theorem 69 (Taylor's Theorem)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a function which is k-times continuously differentiable, and fix $\vec{x}_0 \in \mathbb{R}^n$. Then for all $\vec{x} \in \mathbb{R}^n$ we have

$$f(\vec{x}) = \widehat{f}_k(\vec{x}; \vec{x}_0) + o(\|\vec{x} - \vec{x}_0\|_2^k). \tag{3.72}$$

We can re-write this result in the following, more useful, way for $k = 1$ and $k = 2$:

$$f(\vec{x} + \vec{\delta}) = \underbrace{f(\vec{x}) + [\nabla f(\vec{x})]^\top \vec{\delta}}_{\widehat{f}_1(\vec{x}+\vec{\delta}; \vec{x})} + o(\|\vec{\delta}\|_2) \tag{3.73}$$

$$= \underbrace{f(\vec{x}) + [\nabla f(\vec{x})]^\top \vec{\delta} + \frac{1}{2}\vec{\delta}^\top [\nabla^2 f(\vec{x})]\vec{\delta}}_{\widehat{f}_2(\vec{x}+\vec{\delta}; \vec{x})} + o(\|\vec{\delta}\|_2^2) \tag{3.74}$$

$$= \dots. \tag{3.75}$$

Example 70 (Taylor Approximation of the Squared ℓ^2 norm)**.** In this example we will compute and visualize the first and second degree Taylor approximations of the squared ℓ^2 norm function $f(\vec{x}) = \|\vec{x}\|_2^2$ for $\vec{x} \in \mathbb{R}^2$ around the vector $\vec{x} = \vec{x}_0$. First recall the gradient and hessian of the function which are computed in Examples 55 and 64, respectively.

- First degree approximation:

$$\begin{aligned}\widehat{f}_1(\vec{x};\vec{x}_0) &= f(\vec{x}_0) + [\nabla f(\vec{x}_0)]^\top(\vec{x}-\vec{x}_0) && (3.76)\\ &= \|\vec{x}_0\|_2^2 + [2\vec{x}_0]^\top(\vec{x}-\vec{x}_0) && (3.77)\\ &= 2\vec{x}_0^\top\vec{x} - \|\vec{x}_0\|_2^2\,. && (3.78)\end{aligned}$$

Recall that the graph of $f(\vec{x})$ has a paraboloid shape. Let us now evaluate and visualize the first degree approximation of the function around $\vec{x}_0 = \begin{bmatrix}1\\0\end{bmatrix}$.

$$\widehat{f}_1(\vec{x};\vec{x}_0) = 2x_1 - 1. \qquad (3.79)$$

We plot this function in Figure 3.3 and notice that the graph of the first order approximation is the plane tangent to the paraboloid at the point $(1, 0, f(1, 0)) = (1, 0, 1)$.

- Second degree approximation:

$$\begin{aligned}\widehat{f}_2(\vec{x};\vec{x}_0) &= \widehat{f}_1(\vec{x};\vec{x}_0) + \frac{1}{2}(\vec{x}-\vec{x}_0)^\top[\nabla^2 f(\vec{x}_0)](\vec{x}-\vec{x}_0) && (3.80)\\ &= \underbrace{2\vec{x}_0^\top\vec{x} - \|\vec{x}_0\|_2^2}_{=\widehat{f}_1(\vec{x};\vec{x}_0)} + \frac{1}{2}(\vec{x}-\vec{x}_0)^\top[2I](\vec{x}-\vec{x}_0) && (3.81)\\ &= 2\vec{x}_0^\top\vec{x} - \vec{x}_0^\top\vec{x}_0 + (\vec{x}-\vec{x}_0)^\top(\vec{x}-\vec{x}_0) && (3.82)\\ &= 2\vec{x}_0^\top\vec{x} - \vec{x}_0^\top\vec{x}_0 + \vec{x}^\top\vec{x} - \vec{x}_0^\top\vec{x} - \vec{x}^\top\vec{x}_0 + \vec{x}_0^\top\vec{x}_0 && (3.83)\\ &= 2\vec{x}_0^\top\vec{x} - \vec{x}_0^\top\vec{x}_0 + \vec{x}^\top\vec{x} - 2\vec{x}_0^\top\vec{x} + \vec{x}_0^\top\vec{x}_0 && (3.84)\\ &= \vec{x}^\top\vec{x} && (3.85)\\ &= \|\vec{x}\|_2^2\,. && (3.86)\end{aligned}$$

Thus $\widehat{f}_2 = f$ independently of the choice of $\vec{x}_0$, which makes sense since f is a quadratic function.

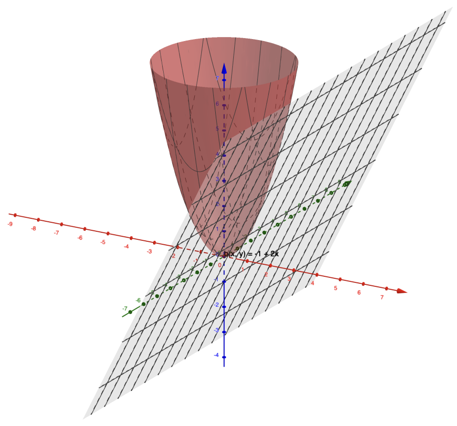


Figure 3.3: First degree Taylor approximation of the function $f(\vec{x}) = \|\vec{x}\|_2^2$

Example 71. We can also compute gradients using Taylor's theorem by pattern matching; this is sometimes much neater than taking componentwise gradients. At first glance this seems circular, but we will see how it is possible. Take for example the function $f\colon \mathbb{R}^n \to \mathbb{R}$ given by $f(\vec{x}) = \vec{x}^\top A\vec{x}$. We can perturb f around $\vec{x}$ to obtain

$$f(\vec{x}+\vec{\delta}) = (\vec{x}+\vec{\delta})^\top A(\vec{x}+\vec{\delta}) \qquad (3.87)$$

$$
\begin{aligned}
&= \vec{x}^\top A\vec{x} + \vec{\delta}^\top A\vec{x} + \vec{x}^\top A\vec{\delta} + \vec{\delta}^\top A\vec{\delta} && (3.88)\\
&= f(\vec{x}) + (\vec{x}^\top A^\top + \vec{x}^\top A)\vec{\delta} + \vec{\delta}^\top A\vec{\delta} && (3.89)\\
&= f(\vec{x}) + ((A + A^\top)\vec{x})^\top\vec{\delta} + \frac{1}{2}\vec{\delta}^\top (A + A^\top)\vec{\delta}. && (3.90)
\end{aligned}
$$

However, Taylor's theorem tells us that

$$f(\vec{x} + \vec{\delta}) = f(\vec{x}) + [\nabla f(\vec{x})]^\top\vec{\delta} + \frac{1}{2}\vec{\delta}^\top[\nabla^2 f(\vec{x})]\vec{\delta} + o(\|\vec{\delta}\|_2^2). \tag{3.91}$$

By pattern matching, we see that $\nabla f(\vec{x}) = (A + A^\top)\vec{x}$ (as obtained in a previous example), $\nabla^2 f(\vec{x}) = A + A^\top$, and the remainder term is $\vec{0}$.

One final note is that we changed $2A \to A + A^\top$ in Equation (3.90); this is to ensure that the Hessian is symmetric, which is a consequence of Theorem 63; and we are able to do this because

$$\vec{\delta}^\top(2A)\vec{\delta} = \vec{\delta}^\top A\vec{\delta} + \vec{\delta}^\top A\vec{\delta} = \vec{\delta}^\top A\vec{\delta} + (\vec{\delta}^\top A\vec{\delta})^\top = \vec{\delta}^\top A\vec{\delta} + \vec{\delta}^\top A^\top\vec{\delta} = \vec{\delta}^\top(A + A^\top)\vec{\delta}. \tag{3.92}$$

We conclude by introducing a more general version of a first-order Taylor approximation, a corresponding Taylor's theorem, and giving an example of when it is useful.

Definition 72 (Yet Another Taylor Approximation)
Let $\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^m$ and fix $\vec{x}_0 \in \mathbb{R}^n$. If $\vec{f}$ is continuously differentiable, then its first-order Taylor approximation around $\vec{x}_0$ is the function $\hat{\vec{f}}_1\colon \mathbb{R}^n \to \mathbb{R}^m$ given by

$$\hat{\vec{f}}_1(\vec{x}; \vec{x}_0) = \vec{f}(\vec{x}_0) + [D\vec{f}(\vec{x}_0)](\vec{x} - \vec{x}_0). \tag{3.93}$$

Again, higher-order approximations will require higher-order derivatives, which requires tensors.

Theorem 73 (Yet Another Taylor's Theorem)
Let $\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^m$ be a continuously differentiable function, and fix $\vec{x}_0 \in \mathbb{R}^n$. Then for all $\vec{x} \in \mathbb{R}^n$ we have

$$\vec{f}(\vec{x}) = \hat{\vec{f}}_1(\vec{x}; \vec{x}_0) + \vec{o}(\|\vec{x} - \vec{x}_0\|_2). \tag{3.94}$$

We can again re-write this result in a more workable form:

$$\vec{f}(\vec{x} + \vec{\delta}) = \vec{f}(\vec{x}) + [D\vec{f}(\vec{x})]\vec{\delta} + o(\|\vec{\delta}\|_2). \tag{3.95}$$

Example 74. Taylor's theorem can be used to compute gradients by pattern matching, even when the function is not linear or quadratic. For instance, we now use it to derive the chain rule (albeit with stronger assumptions on the functions). Let $\vec{f}\colon \mathbb{R}^p \to \mathbb{R}^m$ and $\vec{g}\colon \mathbb{R}^n \to \mathbb{R}^p$ be continuously differentiable. Let $\vec{h}\colon \mathbb{R}^n \to \mathbb{R}^m$ be defined as $\vec{h}(\vec{x}) = \vec{f}(\vec{g}(\vec{x}))$. Then we compute $\vec{h}$ on a perturbation around $\vec{x}$ and expand:

$$
\begin{aligned}
\vec{h}(\vec{x} + \vec{\delta}) &= \vec{f}(\vec{g}(\vec{x} + \vec{\delta})) && (3.96)\\
&\approx \vec{f}(\vec{g}(\vec{x}) + [D\vec{g}(\vec{x})]\vec{\delta})) && (3.97)\\
&\approx \vec{f}(\vec{g}(\vec{x})) + [D\vec{f}(\vec{g}(\vec{x}))]([D\vec{g}(\vec{x})]\vec{\delta}) && (3.98)\\
&\approx \vec{f}(\vec{g}(\vec{x})) + [D\vec{f}(\vec{g}(\vec{x}))][D\vec{g}(\vec{x})]\vec{\delta}. && (3.99)
\end{aligned}
$$

The first Taylor expansion is an expansion of $\vec{g}$ around the point $\vec{x}$ with perturbation $\vec{\delta}$; the second Taylor expansion is an expansion of $\vec{f}$ around the point $g(\vec{x})$ with perturbation $[D\vec{g}(\vec{x})]\vec{\delta}$.

Meanwhile, Taylor's theorem says that

$$\vec{h}(\vec{x}+\vec{\delta}) \approx \vec{h}(\vec{x}) + [D\vec{h}(\vec{x})]\vec{\delta}. \tag{3.100}$$

Thus by pattern matching we find that

$$D\vec{h}(\vec{x}) = [D\vec{f}(\vec{g}(\vec{x}))][D\vec{g}(\vec{x})] \tag{3.101}$$

which is precisely the chain rule!

Note that here we did not invoke the little-o notation because it turns out to be quite messy, but it is indeed possible to do the required rigorous manipulations and get the same result.

As a last practical note, remembering the formula for Taylor approximations helps us confirm our understanding of the dimensions of each vector. For instance, every term should multiply to a scalar. This makes it simpler to remember that, for a function $f\colon \mathbb{R}^n \to \mathbb{R}$, the gradient ∇f outputs column vectors in $\mathbb{R}^n$, the Hessian $\nabla^2 f$ outputs square matrices in $\mathbb{R}^{n\times n}$, etc.

3.3 The Main Theorem

In this section, we will use the concepts introduced in the previous sections to state and prove one of the fundamental ideas in optimization.

Theorem 75 (The Main Theorem [4])

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, and let $\Omega \subseteq \mathbb{R}^n$ be an open set.[a] Consider the optimization problem

$$\min_{\vec{x}\in\Omega} f(\vec{x}). \tag{3.102}$$

Let $\vec{x}^\star$ be a solution to this optimization problem. Then

$$\nabla f(\vec{x}^\star) = \vec{0}. \tag{3.103}$$

[a] "Open sets" are analogous to open intervals (a, b) — i.e., not containing boundary points.

This theorem gives a *necessary* condition for a point to be an optimal solution of this optimization problem. It says that any point that is optimal must necessarily have gradient equal to zero.

Proof. We prove this for scalar functions $f\colon \mathbb{R} \to \mathbb{R}$ only; the vector case is a bit more complicated and is left as an exercise.

Using Taylor approximation of the function around the optimal point:

$$f(x) = f(x^\star) + \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star)\cdot(x-x^\star) + o(|x-x^\star|). \tag{3.104}$$

Since $f(x^\star) \leq f(x)$ for all $x \in \Omega$, we have

$$f(x) \leq f(x) + \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star)\cdot(x-x^\star) + o(|x-x^\star|) \tag{3.105}$$

$$\implies 0 \leq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star)\cdot(x-x^\star) + o(|x-x^\star|). \tag{3.106}$$

Since Ω is an open set, there exists some ball of positive radius $r > 0$ around $x^\star$ such that $B_r(x^\star) \subseteq \Omega$. Formally,

$$B_r(x^\star) = \{x \in \mathbb{R} \mid |x - x^\star| \leq r\}. \tag{3.107}$$

Let us partition $B_r(x^\star)$ into B_+, the set of all $x \in B_r(x^\star)$ such that $x - x^\star \geq 0$, and B_-, the set of all $x \in B_r(x^\star)$ such that $x - x^\star < 0$.

For all $x \in B_+$, we have

$$\begin{aligned} 0 &\leq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) \cdot (x - x^\star) + o(|x - x^\star|) && (3.108) \\ &= \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) \cdot |x - x^\star| + o(|x - x^\star|) && (3.109) \\ \implies 0 &\leq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) + \frac{o(|x - x^\star|)}{|x - x^\star|}. && (3.110) \end{aligned}$$

Taking the limit as $x \to x^\star$ within B_+, we have

$$\begin{aligned} 0 &\leq \lim_{\substack{x \to x^\star \\ x \in B_+}} \left\{ \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) + \frac{o(|x - x^\star|)}{|x - x^\star|} \right\} && (3.111) \\ &= \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) + \lim_{\substack{x \to x^\star \\ x \in B_+}} \frac{o(|x - x^\star|)}{|x - x^\star|} && (3.112) \\ &= \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star). && (3.113) \end{aligned}$$

Thus we have $0 \leq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star)$. On the other hand, for all $x \in B_-$, we have

$$\begin{aligned} 0 &\leq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) \cdot (x - x^\star) + o(|x - x^\star|) && (3.114) \\ &= -\frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) \cdot |x - x^\star| + o(|x - x^\star|) && (3.115) \\ \implies 0 &\geq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) - \frac{o(|x - x^\star|)}{|x - x^\star|}. && (3.116) \end{aligned}$$

Taking the limit $x \to x^\star$ within B_-, we have

$$\begin{aligned} 0 &\geq \lim_{\substack{x \to x^\star \\ x \in B_-}} \left\{ \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) - \frac{o(|x - x^\star|)}{|x - x^\star|} \right\} && (3.117) \\ &= \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) - \lim_{\substack{x \to x^\star \\ x \in B_-}} \frac{o(|x - x^\star|)}{|x - x^\star|} && (3.118) \\ &= \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star). && (3.119) \end{aligned}$$

Thus we have $0 \geq \frac{\mathrm{d}f}{\mathrm{d}x}(x^\star)$ and so $\frac{\mathrm{d}f}{\mathrm{d}x}(x^\star) = 0$. □

3.4 Directional Derivatives

Recall the definition of the partial derivative of a multivariate function (Definition 47), which represents the rate of change of the function $f(\vec{x})$ along one of the standard basis vectors. We do not need to restrict our treatment to the standard basis vectors; in fact, we can compute the rate of change of the function in any arbitrary direction. This is called the directional derivative.

Definition 76 (Directional Derivative)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be differentiable, and fix $\vec{u} \in \mathbb{R}^n$ such that $\|\vec{u}\|_2 = 1$. The *directional derivative* of f along $\vec{u}$ is

the function $D_{\vec{u}}f\colon \mathbb{R}^n \to \mathbb{R}$ defined by

$$D_{\vec{u}}f(\vec{x}) = \lim_{h\to 0} \frac{f(\vec{x}+h\cdot\vec{u}) - f(\vec{x})}{h}. \tag{3.120}$$

If we know the directional derivative in any direction, we know the gradient; similarly, if we know the gradient, we know the directional derivative. The way to connect the two is given by the following proposition, whose proof is left as an exercise.

Proposition 77

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be differentiable, and fix $\vec{u} \in \mathbb{R}^n$ such that $\|\vec{u}\|_2 = 1$. Then

$$D_{\vec{u}}f(\vec{x}) = \vec{u}^\top[\nabla f(\vec{x})]. \tag{3.121}$$

In particular, $D_{\vec{e}_i}f(\vec{x}) = \frac{\partial f}{\partial x_i}(\vec{x})$.

3.5 (OPTIONAL) Matrix Calculus

So far we have only discussed derivatives of three types of function and all have either scalars or vectors and their input and output. We can think of a more general class of functions that also involve matrices. In this section, we will generalize the idea of derivatives to such functions. We will focus our attention on functions of the form $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$, which take a matrix $X \in \mathbb{R}^{m\times n}$ as input and produce a scalar $f(X)$ as output. Familiar examples of such functions include matrix norms, the determinant, and the trace.

Definition 78 (Gradient)

Let $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$ be differentiable. The *gradient* of f is the function $\nabla f\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ which is defined as

$$\nabla f(X) = \begin{bmatrix} \frac{\partial f}{\partial X_{11}}(X) & \cdots & \frac{\partial f}{\partial X_{1n}}(X) \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{m1}}(X) & \cdots & \frac{\partial f}{\partial X_{mn}}(X) \end{bmatrix} \tag{3.122}$$

There exists a general chain rule for matrix-valued functions, which is provable by flattening out all matrices into vectors and applying the vector chain rule.

Theorem 79 (Chain Rule)

Let $F\colon \mathbb{R}^{p\times q} \to \mathbb{R}^{r\times s}$ and $G\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q}$ be differentiable functions. Let $H\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{r\times s}$ be defined by $H(X) = F(G(X))$ for all $X \in \mathbb{R}^{m\times n}$. Then H is differentiable, and for all i, j, k, ℓ, we have

$$\frac{\partial H_{ij}}{\partial X_{k\ell}}(X) = \sum_a \sum_b \frac{\partial F_{ij}}{\partial G_{ab}}(G(X)) \frac{\partial G_{ab}}{\partial X_{k\ell}}(X) \tag{3.123}$$

As before, the notation $\frac{\partial F_{ij}}{\partial G_{ab}}$ means to take the derivative of the ij^{th} output of F by its ab^{th} input. A more specific version of this chain rule is given below for functions $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$.

Proposition 80

Let $w\colon \mathbb{R}^{p\times q} \to \mathbb{R}$ and $G\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q}$ be differentiable functions. Let $h\colon \mathbb{R}^{m\times n} \to \mathbb{R}$ be defined by $h(X) = f(G(X))$ for all $X \in \mathbb{R}^{m\times n}$. Then h is differentiable, and for all k, ℓ, we have

$$\frac{\partial h}{\partial X_{k\ell}}(X) = \sum_a \sum_b [\nabla f(G(X))]_{ab} \frac{\partial G_{ab}}{\partial X_{k\ell}}(X) \tag{3.124}$$

We also are able to define a first-order Taylor expansion without having to use tensor notation.

Definition 81 (Matrix Taylor Approximation)

Let $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$ and fix $X_0 \in \mathbb{R}^{m\times n}$. If f is continuously differentiable, then its first-order Taylor approximation around X_0 is the function $\widehat{f}_1(\cdot; X_0)\colon \mathbb{R}^{m\times n} \to \mathbb{R}$ given by

$$\widehat{f}_1(X; X_0) = f(X_0) + \operatorname{tr}\left([\nabla f(X_0)]^\top (X - X_0)\right). \tag{3.125}$$

There is a corresponding Taylor's theorem certifying the Taylor approximation accuracy, but we don't state it here.

Finally, note that the general recipe for computing all quantities such as the gradient, Jacobian, and gradient matrix is the same: consider each input component and each output component separately and organize their partial derivatives in vector or matrix form with a standard layout.

Chapter 4

Linear and Ridge Regression

Relevant sections of the textbooks:

- [2] Chapter 6.

4.1 Impact of Perturbations on Linear Regression

Before we start thinking about generic convex analysis, we will first study a particularly instructive, useful, and interesting linear-algebraic convex optimization problem.

Let $A \in \mathbb{R}^{n\times n}$ be invertible and $\vec{y} \in \mathbb{R}^{m}$. Consider the generic linear system $A\vec{x} = \vec{y}$, perhaps representing measurements of some physical system. There is exactly one $\vec{x}$ which solves this system — that being $\vec{x} = A^{-1}\vec{y}$. We want to understand how sensitive this system is to perturbations in the output. That is, if $\vec{y}$ is perturbed by $\vec{\delta}_{\vec{y}}$ for $\left\|\vec{\delta}_{\vec{y}}\right\|_2$ small (say, representing noise in the measurements), then the $\vec{x}$ that solves the system is *also* perturbed, say by $\vec{\delta}_{\vec{x}}$. So in the end, we have

$$A(\vec{x} + \vec{\delta}_{\vec{x}}) = (\vec{y} + \vec{\delta}_{\vec{y}}). \tag{4.1}$$

We want to compute the relative change in $\vec{x}$, that is, $\frac{\|\vec{\delta}_{\vec{x}}\|_2}{\|\vec{x}\|_2}$, in terms of $\left\|\vec{\delta}_{\vec{y}}\right\|_2$, as well as other properties of the system. In the context of our physical measurement system, we would much rather have this ratio be *small*; this means that the solutions to the equations governing our physical system are *robust* to measurement errors, thus assuring us that our model is relatively accurate to the real-life physical system. Thus, at the least we want to upper-bound $\frac{\|\vec{\delta}_{\vec{x}}\|_2}{\|\vec{x}\|_2}$.

The first part of upper-bounding $\frac{\|\vec{\delta}_{\vec{x}}\|_2}{\|\vec{x}\|_2}$ is to upper-bound $\left\|\vec{\delta}_{\vec{x}}\right\|_2$. We have

$$\begin{aligned} A(\vec{x} + \vec{\delta}_{\vec{x}}) &= \vec{y} + \vec{\delta}_{\vec{y}} &(4.2)\\ A\vec{x} + A\vec{\delta}_{\vec{x}} &= \vec{y} + \vec{\delta}_{\vec{y}} &(4.3)\\ A\vec{\delta}_{\vec{x}} &= \vec{\delta}_{\vec{y}} &(4.4)\\ \vec{\delta}_{\vec{x}} &= A^{-1}\vec{\delta}_{\vec{y}}. &(4.5)\end{aligned}$$

Then by taking norms on both sides,

$$\begin{aligned} \left\|\vec{\delta}_{\vec{x}}\right\|_2 &= \left\|A^{-1}\vec{\delta}_{\vec{y}}\right\|_2 &(4.6)\\ &\leq \max_{\substack{\vec{z}\in\mathbb{R}^n \\ \|\vec{z}\|_2 = \|\vec{\delta}_{\vec{y}}\|_2}} \left\|A^{-1}\vec{z}\right\|_2 &(4.7)\end{aligned}$$

$$= \left(\max_{\substack{\vec{z} \in \mathbb{R}^n \\ \|\vec{z}\|_2 = 1}} \left\|A^{-1}\vec{z}\right\|_2 \right) \left\|\vec{\delta}_{\vec{y}}\right\|_2 \tag{4.8}$$

$$= \left\|A^{-1}\right\|_2 \left\|\vec{\delta}_{\vec{y}}\right\|_2 . \tag{4.9}$$

In order to upper-bound $\frac{\|\vec{\delta}_{\vec{x}}\|_2}{\|\vec{x}\|_2}$, we also need to lower-bound $\|\vec{x}\|_2$. Applying the same matrix norm inequality to the regular linear system $A\vec{x} = \vec{y}$ gives

$$A\vec{x} = \vec{y} \tag{4.10}$$

$$\|A\vec{x}\|_2 = \|\vec{y}\|_2 \tag{4.11}$$

$$\|A\|_2 \|\vec{x}\|_2 \geq \|\vec{y}\|_2 \tag{4.12}$$

$$\|\vec{x}\|_2 \geq \frac{\|\vec{y}\|}{\|A\|_2} \tag{4.13}$$

where $\|A\|_2 \neq 0$ because A is invertible. Plugging in both bounds, we have

$$\frac{\left\|\vec{\delta}_{\vec{x}}\right\|_2}{\|\vec{x}\|_2} \leq \frac{\left\|A^{-1}\right\|_2 \left\|\vec{\delta}_{\vec{y}}\right\|_2}{\|\vec{y}\|_2 / \|A\|_2} \tag{4.14}$$

$$= \|A\|_2 \left\|A^{-1}\right\|_2 \cdot \frac{\left\|\vec{\delta}_{\vec{y}}\right\|_2}{\|\vec{y}\|_2}. \tag{4.15}$$

Thus we've bounded the relative change in $\vec{x}$ by the relative change in $\vec{y}$. If the relative change in $\vec{y}$ is small, then the relative change in $\vec{x}$ will be small, and so on. But we'd like to say something more about $\|A\|_2 \left\|A^{-1}\right\|_2$, and indeed we can:

$$\|A\|_2 \left\|A^{-1}\right\|_2 = \sigma_1\{A\} \cdot \sigma_1\{A^{-1}\} = \frac{\sigma_1\{A\}}{\sigma_n\{A\}}, \tag{4.16}$$

where again, $\sigma_n\{A\} \neq 0$ because A is invertible. This quantity

$$\kappa(A) \doteq \frac{\sigma_1\{A\}}{\sigma_n\{A\}} \tag{4.17}$$

is called the *condition number* of a matrix. In general, for non-invertible systems, this can be infinite, but has the same definition.

Definition 82 (Condition Number)
Let $A \in \mathbb{R}^{n \times n}$. The *condition number* of A, denoted $\kappa(A)$, is given by

$$\kappa(A) \doteq \frac{\sigma_1\{A\}}{\sigma_n\{A\}}. \tag{4.18}$$

If $\kappa(A)$ is large, then even a small change in our measurement $\vec{y}$ will result in a huge change in our variable $\vec{x}$. If $\kappa(A)$ is small, then large changes in our measurement $\vec{y}$ result in small changes to our variable $\vec{x}$.

It seems unlikely that in general, the equations that define our system will be square. Most likely we will have a least-squares type *tall* system. But this is resolved by using the so-called *normal equations* to represent the least squares solution:

$$A^\top A\vec{x} = A^\top \vec{y}. \tag{4.19}$$

The condition number of this linear system is $\kappa(A^\top A)$. Since $A^\top A$ is symmetric and positive semidefinite, its eigenvalues are also its singular values, and so we have

$$\kappa(A^\top A) = \frac{\lambda_{\max}\{A^\top A\}}{\lambda_{\min}\{A^\top A\}}. \tag{4.20}$$

4.2 Ridge Regression

Sometimes we have least squares systems with $\kappa(A^\top A) = \infty$, or even finite but very large. How do we make this system solvable, robust, or even better conditioned? The answer goes like the following: suppose we could add some number λ to all eigenvalues of $A^\top A$. Then, since $\lambda_1\{A^\top A\}$ and $\lambda_n\{A^\top A\}$ go up by the same amount, $\lambda_n\{A^\top A\}$ becomes a larger fraction of $\lambda_1\{A^\top A\}$, so the condition number $\kappa(A^\top A)$ becomes lower.

This is perhaps easier to see with a numerical example, which we provide now. Suppose that $A^\top A$ has $\lambda_1\{A^\top A\} = 5$ and $\lambda_n\{A^\top A\} = 0.01$. Then $\kappa(A^\top A) = 500$. But if we add 3 to all eigenvalues of $A^\top A$, then $\lambda_1\{A^\top A\} = 8$ and $\lambda_n\{A^\top A\} = 3.01$, so $\kappa(A) = \frac{8}{3.01} \approx 2.65$. This is a much better-conditioned problem.

The question is now how to add λ to all eigenvalues of $A^\top A$. Using the shift property of eigenvalues, we see that we can add λI to $A^\top A$, so that instead of solving the system $A^\top A\vec{x} = A^\top\vec{y}$ we instead solve the system $(A^\top A + \lambda I)\vec{x} = A^\top\vec{y}$. The problem of finding $\vec{x}$ which solves this system is called *ridge regression*. It turns out to be equivalent to the following formulation.

Theorem 83 (Ridge Regression)
Let $A \in \mathbb{R}^{m\times n}$, $\vec{y} \in \mathbb{R}^m$, and $\lambda > 0$. The unique solution to the *ridge regression* problem

$$\min_{\vec{x}\in\mathbb{R}^n} \left\{\|A\vec{x}-\vec{y}\|_2^2 + \lambda\|\vec{x}\|_2^2\right\} \tag{4.21}$$

is given by

$$\vec{x}^\star = (A^\top A + \lambda I)^{-1}A^\top\vec{y}. \tag{4.22}$$

Proof. Let $f(\vec{x}) \doteq \|A\vec{x}-\vec{y}\|_2^2 + \lambda\|\vec{x}\|_2^2$. By taking gradients, we get

$$\begin{aligned}
\nabla_{\vec{x}} f(\vec{x}) &= \nabla_{\vec{x}}\left\{\|A\vec{x}-\vec{y}\|_2^2 + \lambda\|\vec{x}\|_2^2\right\} && (4.23)\\
&= \nabla_{\vec{x}}\{\vec{x}^\top A^\top A\vec{x} - 2\vec{y}^\top A\vec{x} + \vec{y}^\top\vec{y} + \lambda\vec{x}^\top\vec{x}\} && (4.24)\\
&= 2A^\top A\vec{x} - 2A^\top\vec{y} + 2\lambda\vec{x} && (4.25)\\
&= 2(A^\top A + \lambda I)\vec{x} - 2A^\top\vec{y}. && (4.26)
\end{aligned}$$

Thus we get that the optimal point is determined by solving the linear system

$$(A^\top A + \lambda I)\vec{x} = A^\top\vec{y}. \tag{4.27}$$

Since $A^\top A$ is PSD and $\lambda > 0$, we have $A^\top A + \lambda I$ is PD and thus invertible. Therefore

$$\vec{x}^\star = (A^\top A + \lambda I)^{-1}A^\top\vec{y} \tag{4.28}$$

is the unique solution to the above linear system and therefore the unique solution to the optimization problem. □

Note that we haven't proved that a convex function (such as the above ridge regression objective) is minimized when its derivative is 0; we prove this in subsequent lectures, but for now let us take it for granted.

Proof. Another way to solve the same problem is to consider the augmented system

$$\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \vec{x} = \begin{bmatrix} \vec{y} \\ \vec{0} \end{bmatrix}. \tag{4.29}$$

This augmented matrix has full column rank, so we can use the least squares solution to get a unique solution for $\vec{x}$. We get

$$\begin{aligned}
\vec{x} &= \left(\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}^\top \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}^\top \begin{bmatrix} \vec{y} \\ \vec{0} \end{bmatrix} && (4.30) \\
&= \left(\begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} \vec{y} \\ \vec{0} \end{bmatrix} && (4.31) \\
&= \left(\begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \left(A^\top \vec{y} + \sqrt{\lambda} I \cdot \vec{0} \right) && (4.32) \\
&= \left(\begin{bmatrix} A^\top & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} A^\top \vec{y} && (4.33) \\
&= \left(A^\top A + \lambda I \right)^{-1} A^\top \vec{y}. && (4.34)
\end{aligned}$$

□

In the ridge regression objective

$$\left\| A\vec{x} - \vec{b} \right\|_2^2 + \lambda \left\| \vec{x} \right\|_2^2, \tag{4.35}$$

the second term $\lambda \left\| \vec{x} \right\|_2^2$ is called a *regularizer*; this is because it *regulates* or *regularizes* our problem by making it better-conditioned.

4.3 Principal Components Regression

We can gain more understanding of the ridge regression solution by looking at it through the SVD of A. Indeed, let $A = U\Sigma V^\top$. Then the ridge regression solution is

$$\begin{aligned}
x^\star &= \left(A^\top A + \lambda I \right)^{-1} A^\top \vec{y} && (4.36) \\
&= \left((U\Sigma V^\top)^\top (U\Sigma V^\top) + \lambda I \right)^{-1} (U\Sigma V^\top)^\top \vec{y} && (4.37) \\
&= \left(V\Sigma^\top U^\top U\Sigma V^\top + \lambda I \right)^{-1} V\Sigma^\top U^\top \vec{y} && (4.38) \\
&= \left(V\Sigma^\top \Sigma V^\top + \lambda I \right)^{-1} V\Sigma^\top U^\top \vec{y} && (4.39) \\
&= \left(V\Sigma^\top \Sigma V^\top + V(\lambda I)V^\top \right)^{-1} V\Sigma^\top U^\top \vec{y} && (4.40) \\
&= \left(V \left(\Sigma^\top \Sigma + \lambda I \right) V^\top \right)^{-1} V\Sigma^\top U^\top \vec{y} && (4.41) \\
&= V \left(\Sigma^\top \Sigma + \lambda I \right)^{-1} V^\top V \Sigma^\top U^\top \vec{y} && (4.42) \\
&= V \left(\Sigma^\top \Sigma + \lambda I \right)^{-1} \Sigma^\top U^\top \vec{y} && (4.43) \\
&= V \begin{bmatrix} (\Sigma_r^2 + \lambda I)^{-1} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} U^\top \vec{y}. && (4.44)
\end{aligned}$$

Looking at the middle matrix a bit more, we see that

$$(\Sigma_r^2 + \lambda I)^{-1}\Sigma_r = \begin{bmatrix} \frac{\sigma_1\{A\}}{\sigma_1\{A\}^2+\lambda} & & \\ & \ddots & \\ & & \frac{\sigma_n\{A\}}{\sigma_n\{A\}^2+\lambda} \end{bmatrix}.$$

Thus, we get

$$\vec{x}^\star = V \begin{bmatrix} (\Sigma_r^2 + \lambda I)^{-1}\Sigma_r & 0 \\ 0 & 0 \end{bmatrix} U^\top \vec{y} \tag{4.45}$$

$$= \left(\sum_{i=1}^{r} \frac{\sigma_i\{A\}}{\sigma_i\{A\}^2 + \lambda}\vec{v}_i\vec{u}_i^\top\right)\vec{y} \tag{4.46}$$

$$= \sum_{i=1}^{r} \frac{\sigma_i\{A\}}{\sigma_i\{A\}^2 + \lambda}(\vec{u}_i^\top\vec{y}) \cdot \vec{v}_i. \tag{4.47}$$

To understand what λ is doing here, we contrast two examples. Let $A \in \mathbb{R}^{n\times 3}$ for some large $n \gg 3$.

Suppose first that $\sigma_1\{A\} = \sigma_2\{A\} = \sigma_3\{A\} = 1$. Then

$$\vec{x}^\star = \frac{\sigma_1\{A\}}{\sigma_1\{A\}^2+\lambda}(\vec{u}_1^\top\vec{y})\vec{v}_1 + \frac{\sigma_2\{A\}}{\sigma_2\{A\}^2+\lambda}(\vec{u}_2^\top\vec{y})\vec{v}_2 + \frac{\sigma_3\{A\}}{\sigma_3\{A\}^2+\lambda}(\vec{u}_3^\top\vec{y})\vec{v}_3 \tag{4.48}$$

$$= \frac{1}{1+\lambda}\{(\vec{u}_1^\top\vec{y})\vec{v}_1 + (\vec{u}_2^\top\vec{y})\vec{v}_2 + (\vec{u}_3^\top\vec{y})\vec{v}_3\} \tag{4.49}$$

$$= \frac{1}{1+\lambda}\vec{\tilde{x}} \tag{4.50}$$

where $\vec{\tilde{x}}$ is the solution of the corresponding least squares linear regression problem, namely the ridge problem with $\lambda = 0$. In this way, the λ parameter decays the solution in each principal direction equally, pulling the whole $\vec{\tilde{x}}$ vector towards 0. This is interesting precisely because a first-level examination of the ridge regression objective function — and namely the $\|\vec{x}\|_2^2$ term, which by itself penalizes every direction of $\vec{x}$ equally — may make it seem like this is always the case, but it turns out to not be, as we will see shortly.

Now suppose that $\sigma_1\{A\} = 100$, $\sigma_2\{A\} = 10$, and $\sigma_3\{A\} = 1$. Then

$$\vec{x}^\star = \frac{\sigma_1\{A\}}{\sigma_1\{A\}^2+\lambda}(\vec{u}_1^\top\vec{y})\vec{v}_1 + \frac{\sigma_2\{A\}}{\sigma_2\{A\}^2+\lambda}(\vec{u}_2^\top\vec{y})\vec{v}_2 + \frac{\sigma_3\{A\}}{\sigma_3\{A\}^2+\lambda}(\vec{u}_3^\top\vec{y})\vec{v}_3 \tag{4.51}$$

$$= \frac{100}{10000+\lambda}(\vec{u}_1^\top\vec{y})\vec{v}_1 + \frac{10}{100+\lambda}(\vec{u}_2^\top\vec{y})\vec{v}_2 + \frac{1}{1+\lambda}(\vec{u}_3^\top\vec{y})\vec{v}_3 \tag{4.52}$$

$$= \frac{1}{100+\lambda/100}(\vec{u}_1^\top\vec{y})\vec{v}_1 + \frac{1}{10+\lambda/10}(\vec{u}_2^\top\vec{y})\vec{v}_2 + \frac{1}{1+\lambda}(\vec{u}_3^\top\vec{y})\vec{v}_3. \tag{4.53}$$

Thus, the different terms are now impacted differently based on λ; in particular, to impact the first term by a certain amount, one needs to change λ by 100 times the amount required to change the last term. Namely, if we set λ to be large, say $\lambda = 10000$, the coefficient of the first term becomes $1/110$, while the coefficient of the last term becomes $1/10001$ which is much lower. More generally, for a larger example, setting λ to be large effectively zeros out the last few terms while effectively not changing the first few terms. Thus, setting λ to be large effectively performs a "soft thresholding" of the singular values, making the terms associated with smaller singular values be nearly 0 while preserving the terms associated with larger singular values. More quantitatively, for large λ, we have

$$\vec{x}^\star = \frac{1}{100+\lambda/100}(\vec{u}_1^\top\vec{y})\vec{v}_1 + \frac{1}{10+\lambda/10}(\vec{u}_2^\top\vec{y})\vec{v}_2 + \frac{1}{1+\lambda}(\vec{u}_3^\top\vec{y})\vec{v}_3 \tag{4.54}$$

$$\approx \frac{1}{100+\lambda/100}(\vec{u}_1^\top \vec{y})\vec{v}_1 + \frac{1}{10+\lambda/10}(\vec{u}_2^\top \vec{y})\vec{v}_2 \tag{4.55}$$

and for even larger λ we simply have

$$\vec{x}^\star = \frac{1}{100+\lambda/100}(\vec{u}_1^\top \vec{y})\vec{v}_1 + \frac{1}{10+\lambda/10}(\vec{u}_2^\top \vec{y})\vec{v}_2 + \frac{1}{1+\lambda}(\vec{u}_3^\top \vec{y})\vec{v}_3 \tag{4.56}$$

$$\approx \frac{1}{100+\lambda/100}(\vec{u}_1^\top \vec{y})\vec{v}_1. \tag{4.57}$$

Since the terms form a linear combination of the $\vec{v}_i$, setting such terms associated with small singular values to (nearly) 0 is similar to performing PCA, where we only use the $\vec{v}_i$ associated with the largest few singular values. Thus our conclusion is — ridge regression behaves qualitatively similar to a soft form of PCA.

4.4 Tikhonov Regression

Recall that our earlier augmented system

$$\begin{bmatrix} A \\ \sqrt{\lambda}I \end{bmatrix} \vec{x} = \begin{bmatrix} \vec{y} \\ \vec{0} \end{bmatrix} \tag{4.58}$$

which had full column rank, tried to find a $\vec{x}$ such that $A\vec{x} \approx \vec{y}$ while $\vec{x} \approx \vec{0}$ — in other words, $\vec{x}$ that is small. Suppose that we wanted to instead enforce that $\vec{x}$ were close to some other vector $\vec{x}_0 \in \mathbb{R}^n$. Then we would set up the system

$$\begin{bmatrix} A \\ \sqrt{\lambda}I \end{bmatrix} \vec{x} = \begin{bmatrix} \vec{y} \\ \vec{x}_0 \end{bmatrix}. \tag{4.59}$$

This would yield the least-squares type objective function

$$\|A\vec{x} - \vec{y}\|_2^2 + \lambda \|\vec{x} - \vec{x}_0\|_2^2. \tag{4.60}$$

The final generalization of this is to put different weights on each row of $A\vec{x} - \vec{y}$ and $\vec{x} - \vec{x}_0$. If, for example, we really want to get row i of $A\vec{x}$ close to b_i, we can multiply the squared difference $(A\vec{x} - \vec{y})_i^2$ by a large weight in the loss function, and the solutions will bias towards ensuring that $(A\vec{x} - \vec{y})_i \approx 0$. Similarly, if we really are sure that the true $\vec{x}$ has i^{th} coordinate $(\vec{x}_0)_i$, then we can attach a large weight to the difference $(\vec{x} - \vec{x}_0)_i^2$ as well. Mathematically, this gives us the following objective function:

$$\|W_1(A\vec{x} - \vec{y})\|_2^2 + \|W_2(\vec{x} - \vec{x}_0)\|_2^2, \tag{4.61}$$

where $W_1 \in \mathbb{R}^{m \times m}$ and $W_2 \in \mathbb{R}^{n \times n}$ are diagonal matrices representing the weights. Notice how this is a generalization of ridge regression with $W_1 = I$, $W_2 = \sqrt{\lambda}I$, and $\vec{x}_0 = \vec{0}$. This general regression is called Tikhonov regression.

Theorem 84 (Tikhonov Regression)
Let $A \in \mathbb{R}^{m \times n}$, $\vec{x}_0 \in \mathbb{R}^n$, and $\vec{y} \in \mathbb{R}^m$, and let $W_1 \in \mathbb{R}^{m \times m}$ and $W_2 \in \mathbb{R}^{n \times n}$ be diagonal. Then the unique solution to the *Tikhonov regression* problem

$$\min_{\vec{x} \in \mathbb{R}^n} \left\{ \|W_1(A\vec{x} - \vec{y})\|_2^2 + \|W_2(\vec{x} - \vec{x}_0)\|_2^2 \right\} \tag{4.62}$$

is given by

$$\vec{x}^\star = (A^\top W_1^2 A + W_2^2)^{-1}(A^\top W_1^2 \vec{y} + W_2^2 \vec{x}_0). \tag{4.63}$$

Proof. Left as exercise. □

This expression looks complicated, so we do a sanity-check; if $W_1 = I$, $W_2 = \sqrt{\lambda}I$, and $\vec{x}_0 = \vec{0}$, then we get exactly the ridge regression solution.

4.5 Maximum Likelihood Estimation (MLE)

Previously, we talked about incorporating side information (like $\vec{x} = \vec{x}_0$) deterministically. Now we discuss a way to incorporate probabilistic information into our model.

Namely, suppose that the rows of our A matrix are vectors $\vec{a}_1, \dots, \vec{a}_m \in \mathbb{R}^n$, and that the entries of our $\vec{y}$ vector are $y_1, \dots, y_m \in \mathbb{R}$. Now suppose we have the probabilistic model

$$y_i = \vec{a}_i^\top \vec{x} + w_i, \qquad \forall i \in \{1, \dots, m\} \tag{4.64}$$

where $w_1, \dots, w_n$ are independent Gaussian random variables; in particular, $w_i \sim \mathcal{N}(0, \sigma_i^2)$. Or in short, we have

$$\vec{y} = A\vec{x} + \vec{w} \tag{4.65}$$

where $\vec{w} = \begin{bmatrix} w_1 & \cdots & w_m \end{bmatrix}^\top \in \mathbb{R}^m$. In this case, we say that $\vec{w} \sim \mathcal{N}(\vec{0}, \Sigma_{\vec{w}})$ where $\Sigma_{\vec{w}} \doteq \operatorname{diag}\left(\sigma_1^2, \dots, \sigma_m^2\right)$

In this setup, the *maximum likelihood estimate* (MLE) for $\vec{x}$ turns out to be exactly a solution to a Tikhonov regression problem. The maximum likelihood estimate is the parameter choice which makes the data most likely, in that it has the highest probability or probability density out of all choices of the parameter. It is a meaningful and popular statistical estimator; thus the fact that we can reduce its computation to a ridge regression-type problem is both interesting and useful.

Henceforth, we use p to denote probability densities, and use $p_{\vec{x}}$ to denote probability densities for a fixed value of $\vec{x}$. In the above model, $\vec{x}$ is not a random variable, so it doesn't quite make formal sense to condition on it (though — spoilers! — we will soon put a probabilistic prior on it, and then it makes sense to condition).

Proposition 85 (MLE as Tikhonov Regression)

In the above probabilistic model, we have

$$\operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} p_{\vec{x}}(\vec{y}) = \operatorname*{argmin}_{\vec{x} \in \mathbb{R}^n} \left\| \Sigma_{\vec{w}}^{-1/2}(A\vec{x} - \vec{y}) \right\|_2^2. \tag{4.66}$$

Proof. Since the logarithm is monotonically increasing, $\operatorname{argmax}_{\vec{x}} f(\vec{x}) = \operatorname{argmax}_{\vec{x}} \log(f(\vec{x}))$ for all functions f, and so

$$\operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} p_{\vec{x}}(\vec{y}) = \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \log(p_{\vec{x}}(\vec{y})) \tag{4.67}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \log\left(\prod_{i=1}^m p_{\vec{x}}(y_i)\right) \tag{4.68}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \sum_{i=1}^m \log(p_{\vec{x}}(y_i)) \tag{4.69}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \sum_{i=1}^m \log\left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{2\sigma_i^2}\right)\right) \tag{4.70}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \sum_{i=1}^{m} \left\{ \underbrace{\log\left(\frac{1}{\sqrt{2\pi\sigma_i^2}}\right)}_{\text{independent of } \vec{x}} + \log\left(\exp\left(-\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{2\sigma_i^2}\right)\right) \right\} \tag{4.71}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \sum_{i=1}^{m} \left\{ \log\left(\exp\left(-\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{2\sigma_i^2}\right)\right) \right\} \tag{4.72}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \sum_{i=1}^{m} \left\{ -\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{2\sigma_i^2} \right\} \tag{4.73}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \left\{ -\frac{1}{2} \sum_{i=1}^{m} \frac{(y_i - \vec{a}_i^\top \vec{x})^2}{\sigma_i^2} \right\} \tag{4.74}$$

$$= \operatorname*{argmin}_{\vec{x} \in \mathbb{R}^n} \sum_{i=1}^{m} \frac{(y_i - \vec{a}_i^\top \vec{x})^2}{\sigma_i^2} \tag{4.75}$$

$$= \operatorname*{argmin}_{\vec{x} \in \mathbb{R}^n} \left\| \Sigma_{\vec{w}}^{-1/2}(A\vec{x} - \vec{y}) \right\|_2^2. \tag{4.76}$$

□

4.6 Maximum A Posteriori Estimation (MAP)

Now we consider the same probabilistic model as above, except this time suppose that we believe $\vec{x}$ is also random, in the sense that

$$x_j = \mu_j + v_j, \qquad \forall j \in \{1, \ldots, n\} \tag{4.77}$$

where $v_1, \ldots, v_n$ are independent Gaussian random variables; in particular $v_j \sim \mathcal{N}(0, \tau_j^2)$. Or in short, we have

$$\vec{x} = \vec{x}_0 + \vec{v} \tag{4.78}$$

where $\vec{v} = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}^\top \in \mathbb{R}^n$ is distributed as $\vec{v} \sim \mathcal{N}(\vec{0}, \Sigma_{\vec{v}})$, where $\Sigma_{\vec{v}} \doteq \operatorname{diag}\left(\tau_1^2, \ldots, \tau_n^2\right)$.

In this setup, the maximum likelihood estimate may still be useful, but another quantity that is perhaps more relevant is the *maximum a posteriori estimate* (MAP). The MAP estimate is the value of $\vec{x}$ which is most likely, i.e., having the highest conditional probability or conditional probability density, conditioned on the observed data. It is also a meaningful and popular statistical estimator. It turns out that we can derive a similar result as in the MLE case.

Theorem 86 (MAP as Tikhonov Regression)
In the above probabilistic model, we have

$$\operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} p(\vec{x} \mid \vec{y}) = \operatorname*{argmin}_{\vec{x} \in \mathbb{R}^n} \left\{ \left\| \Sigma_{\vec{w}}^{-1/2}(A\vec{x} - \vec{y}) \right\|_2^2 + \left\| \Sigma_{\vec{v}}^{-1/2}(\vec{x} - \vec{x}_0) \right\|_2^2 \right\}. \tag{4.79}$$

Proof. Using Bayes' rule and the computations from before, we have

$$\operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} p(\vec{x} \mid \vec{y}) \tag{4.80}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \log(p(\vec{x} \mid \vec{y})) \tag{4.81}$$

$$= \operatorname*{argmax}_{\vec{x} \in \mathbb{R}^n} \log\left(\frac{p(\vec{y} \mid \vec{x})p(\vec{x})}{p(\vec{y})}\right) \tag{4.82}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} \left\{ \log(p(\vec{y} \mid \vec{x})) + \log(p(\vec{x})) - \underbrace{\log(p(\vec{y}))}_{\text{independent of } \vec{x}} \right\} \tag{4.83}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} \{\log(p(\vec{y} \mid \vec{x})) + \log(p(\vec{x}))\} \tag{4.84}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} \log\left(\left\{\prod_{i=1}^{m} p(y_i \mid \vec{x})\right\} \cdot \left\{\prod_{j=1}^{n} p(x_j)\right\}\right) \tag{4.85}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} \left\{\sum_{i=1}^{m} \log(p(y_i \mid \vec{x})) + \sum_{j=1}^{n} \log(p(x_j))\right\} \tag{4.86}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} \left\{\sum_{i=1}^{m} \log\left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{2\sigma_i^2}\right)\right) + \sum_{j=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\tau_j^2}} \exp\left(-\frac{(x_j - (\vec{x}_0)_j)^2}{2\tau_j^2}\right)\right)\right\} \tag{4.87}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} \left\{\sum_{i=1}^{m} \left(-\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{2\sigma_i^2}\right) + \sum_{j=1}^{n} \left(-\frac{(x_j - (\vec{x}_0)_j)^2}{2\tau_j^2}\right)\right\} \tag{4.88}$$

$$= \operatorname*{argmax}_{\vec{x}\in\mathbb{R}^n} -\frac{1}{2} \left\{\sum_{i=1}^{m} \left(\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{\sigma_i^2}\right) + \sum_{j=1}^{n} \left(\frac{(x_j - (\vec{x}_0)_j)^2}{\tau_j^2}\right)\right\} \tag{4.89}$$

$$= \operatorname*{argmin}_{\vec{x}\in\mathbb{R}^n} \left\{\sum_{i=1}^{m} \left(\frac{(y_i - \vec{a}_i^\top \vec{x})^2}{\sigma_i^2}\right) + \sum_{j=1}^{n} \left(\frac{(x_j - (\vec{x}_0)_j)^2}{\tau_j^2}\right)\right\} \tag{4.90}$$

$$= \operatorname*{argmin}_{\vec{x}\in\mathbb{R}^n} \left\{ \left\|\Sigma_{\vec{w}}^{-1/2}(A\vec{x} - \vec{y})\right\|_2^2 + \left\|\Sigma_{\vec{v}}^{-1/2}(\vec{x} - \vec{x}_0)\right\|_2^2 \right\} \tag{4.91}$$

as desired. □

Chapter 5

Convexity

Relevant sections of the textbooks:

- [1] Chapter 4.
- [2] Chapter 8.

5.1 Convex Sets

First, we want to define a special type of linear combination called a *convex combination*.

Definition 87 (Convex Combination)
Let $\vec{x}_1, \ldots, \vec{x}_k \in \mathbb{R}^n$. The sum

$$\vec{x} = \sum_{i=1}^{k} \theta_i \vec{x}_i \tag{5.1}$$

is a *convex combination* of $\vec{x}_1, \ldots, \vec{x}_k$ if each $\theta_i \geq 0$ and $\sum_{i=1}^{k} \theta_i = 1$.

We can think of each θ_i as a weight on the corresponding $\vec{x}_i$. Since they are non-negative numbers which sum to 1, we can also interpret them as probabilities.

Definition 88 (Convex Set)
Let $C \subseteq \mathbb{R}^n$. We say that C is a *convex set* if for all $\vec{x}_1, \vec{x}_2 \in C$ and all $\theta \in [0, 1]$, we have $\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \in C$.

Geometrically, a set C is convex if for every two points $\vec{x}_1, \vec{x}_2 \in C$, the line segment $\{\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \mid \theta \in [0, 1]\}$ is contained in C. This means that, for example, the midpoint between $\vec{x}_1$ and $\vec{x}_2$, i.e., $\frac{1}{2}\vec{x}_1 + \frac{1}{2}\vec{x}_2$, is contained in C, as well as the point $\frac{1}{3}$ of the way from $\vec{x}_1$ to $\vec{x}_2$, i.e., $\frac{1}{3}\vec{x}_1 + \frac{2}{3}\vec{x}_2$, etc. More generally, as we vary θ, we go along the line segment connecting $\vec{x}_1$ and $\vec{x}_2$.

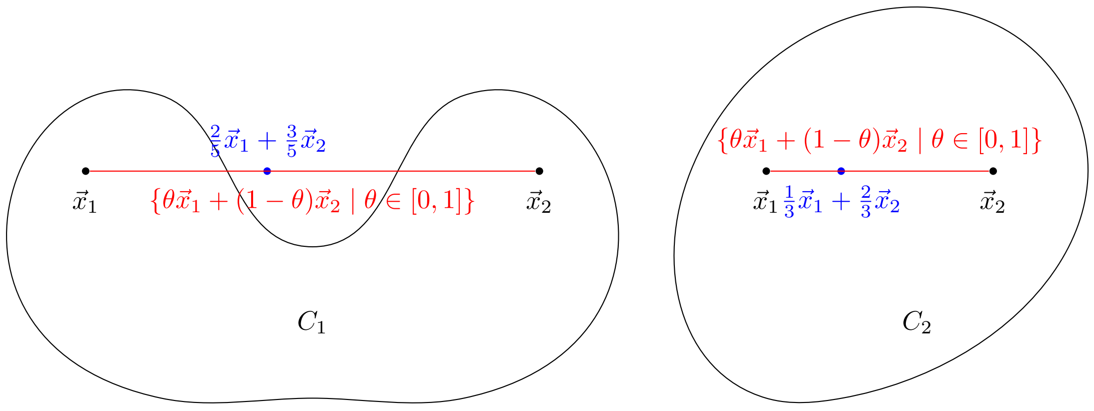

$\frac{2}{5}\vec{x}_1 + \frac{3}{5}\vec{x}_2$
$\vec{x}_1$
$\{\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \mid \theta \in [0,1]\}$
$\vec{x}_2$
C_1
$\{\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \mid \theta \in [0,1]\}$
$\vec{x}_1$
$\frac{1}{3}\vec{x}_1 + \frac{2}{3}\vec{x}_2$
$\vec{x}_2$
C_2

Figure 5.1: Two sets $C_1, C_2 \subseteq \mathbb{R}^2$. C_1 is not convex, but C_2 is. To visualize the behavior of the line segments, we also plot a point on each line segment along with its associated θ.

Algebraically, a set C is convex if for any $\vec{x}_1, \ldots, \vec{x}_k \in C$, any convex combination of $\vec{x}_1, \ldots, \vec{x}_k$ is contained in C.

One way to generate a convex set from any (possibly non-convex) set, including finite and infinite sets, is to take its convex hull.

Definition 89 (Convex Hull)

Let $S \subseteq \mathbb{R}^n$ be a set. The *convex hull* of S, denoted $\operatorname{conv}(S)$, is the set of all convex combinations of points in S, i.e.,

$$\operatorname{conv}(S) = \left\{ \sum_{i=1}^{k} \theta_i \vec{x}_i \;\middle|\; k \in \mathbb{N}, \theta_1, \ldots, \theta_k \geq 0, \sum_{i=1}^{k} \theta_i = 1, \vec{x}_1, \ldots, \vec{x}_k \in S \right\}. \tag{5.2}$$

Here are some properties of the convex hull; the proof is left as an exercise.

Proposition 90

Let $S \subseteq \mathbb{R}^n$ be a set.

(i) $\operatorname{conv}(S)$ is a convex set.

(ii) $\operatorname{conv}(S)$ is the minimal convex set which contains S, i.e.,

$$\operatorname{conv}(S) = \bigcap_{\substack{C \supseteq S \\ C \text{ is a convex set}}} C. \tag{5.3}$$

Thus if S is convex then $\operatorname{conv}(S) = S$.

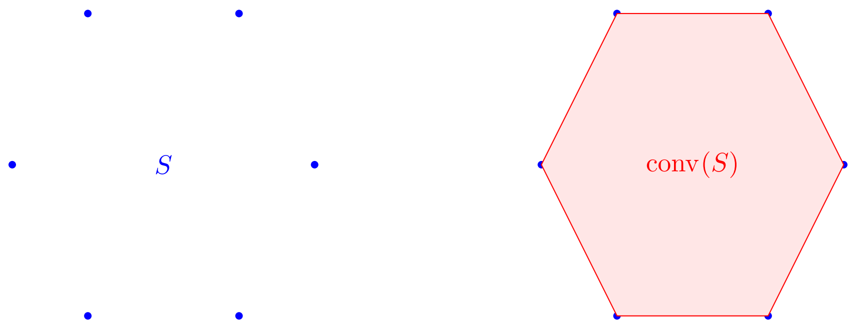
(iii) $\operatorname{conv}(S)$ is the union of convex hulls of all finite subsets of S, i.e.,

$$\operatorname{conv}(S) = \bigcup_{\substack{A \subseteq S \\ |A| < \infty}} \operatorname{conv}(A). \tag{5.4}$$

(iv) $\operatorname{conv}(S)$ can be equivalently expressed using convex combinations of only two points:

$$\operatorname{conv}(S) = \{\theta\vec{x} + (1-\theta)\vec{y} \mid \theta \in [0,1], \vec{x} \in S, \vec{y} \in S\}. \tag{5.5}$$

Below, we visualize the convex hull of a *finite set* S. By the above proposition, the convex hull of an infinite set S' is the union of convex hulls of all finite subsets of S'.



Definition 91 (Hyperplane)
Let $\vec{a}, \vec{x}_0 \in \mathbb{R}^n$ and $b \in \mathbb{R}$. A *hyperplane* is a set of the form

$$\{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top \vec{x} = b\} \tag{5.6}$$

or, equivalently, a set of the form

$$\{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top (\vec{x} - \vec{x}_0) = 0\}. \tag{5.7}$$

The equations $\vec{a}^\top \vec{x} = b$ and $\vec{a}^\top(\vec{x} - \vec{x}_0) = 0$ are connected, because if we define $b = \vec{a}^\top \vec{x}_0$, then the second equation resolves to the first equation; and if we take $\vec{x}_0$ to be any vector such that $\vec{a}^\top \vec{x}_0 = b$, then the first equation resolves to the second equation.

Example 92. Hyperplanes are convex. Consider a hyperplane $H \doteq \{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top \vec{x} = b\}$. Let $\vec{x}_1, \vec{x}_2 \in H$ and $\theta \in [0,1]$. Then

$$\begin{aligned} \vec{a}^\top(\theta\vec{x}_1 + (1-\theta)\vec{x}_2) &= \theta\vec{a}^\top\vec{x}_1 + (1-\theta)\vec{a}^\top\vec{x}_2 && (5.8)\\ &= \theta b + (1-\theta) b && (5.9)\\ &= b && (5.10) \end{aligned}$$

so $\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \in H$. Thus H is convex.

To show that a set C is convex, we need to show that for *every* $\vec{x}_1, \vec{x}_2 \in C$ and *every* $\theta \in [0,1]$, that $\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \in C$.

To show that C is *not* convex, we just need to come up with *one* choice of $\vec{x}_1, \vec{x}_2 \in C$ and *one* $\theta \in [0,1]$ such that $\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \notin C$. Note that even if C is non-convex, there could be *some* choices of $\vec{x}_1, \vec{x}_2 \in C, \theta \in [0,1]$ such that $\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \in C$; but if C is non-convex, there is at least one choice of $\vec{x}_1, \vec{x}_2 \in C, \theta \in [0,1]$ such that $\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \notin C$.

Definition 93 (Half-Space)
Let $\vec{a}, \vec{x}_0 \in \mathbb{R}^n$ and $b \in \mathbb{R}$. A *positive half-space* is a set of the form

$$\{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top \vec{x} \geq b\} \qquad \text{or} \qquad \{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top (\vec{x} - \vec{x}_0) \geq 0\}. \tag{5.11}$$

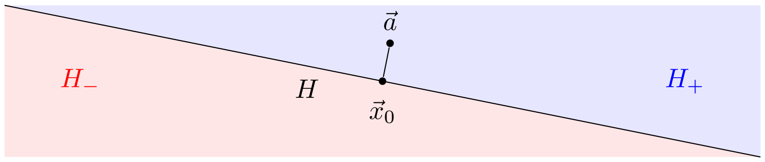
A *negative half-space* is a set of the form

$$\{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top \vec{x} \leq b\} \qquad \text{or} \qquad \{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top (\vec{x} - \vec{x}_0) \leq 0\}. \tag{5.12}$$

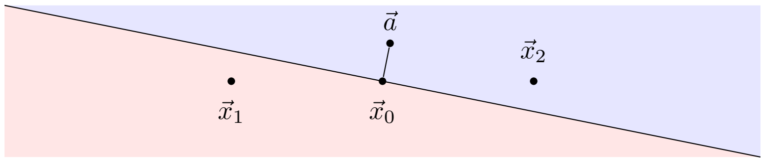
The mental picture we have for these hyperplanes and half-spaces is the following. Let $\vec{x}_0 \in \mathbb{R}^n$ and define

$$H \doteq \{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top (\vec{x} - \vec{x}_0) = 0\} \tag{5.13}$$
$$H_+ \doteq \{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top (\vec{x} - \vec{x}_0) \geq 0\} \tag{5.14}$$
$$H_- \doteq \{\vec{x} \in \mathbb{R}^n \mid \vec{a}^\top (\vec{x} - \vec{x}_0) \leq 0\}. \tag{5.15}$$

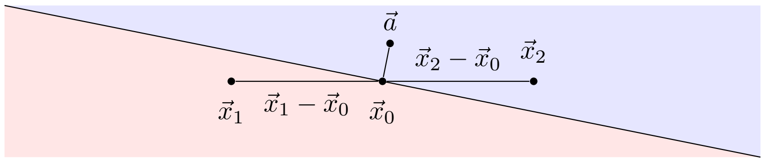
Then the alignment of these objects looks like the following:



In words, the positive and negative half-spaces partition $\mathbb{R}^n$. Looking at some individual vectors, say $\vec{x}_1 \in H_-$ and $\vec{x}_2 \in H_+$, we have the picture



If we draw lines connecting $\vec{x}_0$ with $\vec{x}_1$ and $\vec{x}_2$, they are not themselves representations of $\vec{x}_1$ and $\vec{x}_2$, unless $\vec{x}_0 = \vec{0}$. Instead, they are representations of the *displacements* of $\vec{x}_1$ and $\vec{x}_2$ from $\vec{x}_0$. Thus, we see the following picture:



And this gives us a clearer understanding of what's going on — $\vec{x}_1 - \vec{x}_0$ forms an obtuse angle with $\vec{a}$, indicating a negative dot product, whereas $\vec{x}_2 - \vec{x}_0$ forms an acute angle with $\vec{a}$, indicating a positive dot product. And this is how H_+ and H_- are computed.

This allows us to consider what it means for a hyperplane to separate two sets. It means that for every vector in the first set, the dot product is non-positive, and for every vector in the second set, the dot product is non-negative.

Example 94 (Set of PSD Matrices is Convex)**.** Consider $\mathbb{S}_+^n$, the set of all symmetric positive semidefinite (PSD) matrices. We want to show that $\mathbb{S}_+^n$ is convex. Take $A_1, A_2 \in \mathbb{S}_+^n$ and $\theta \in [0,1]$. We want to show that $\theta A_1 + (1-\theta) A_2 \in \mathbb{S}_+^n$.

One of the ways to tell if a matrix A is PSD is to check whether $\vec{x}^\top A\vec{x} \geq 0$ for all $\vec{x} \in \mathbb{R}^n$. Checking this for our convex combination, we get

$$\vec{x}^\top(\theta A_1 + (1-\theta)A_2)\vec{x} = \theta \underbrace{\vec{x}^\top A_1\vec{x}}_{\geq 0} + (1-\theta)\underbrace{\vec{x}^\top A_2\vec{x}}_{\geq 0} \tag{5.16}$$

$$\geq 0. \tag{5.17}$$

Note that it is possible to come up with linear combinations of PSD matrices that are not PSD; indeed, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ are PSD, yet their difference $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$ is not PSD. But all convex combinations of PSD matrices are PSD, as we have confirmed above.

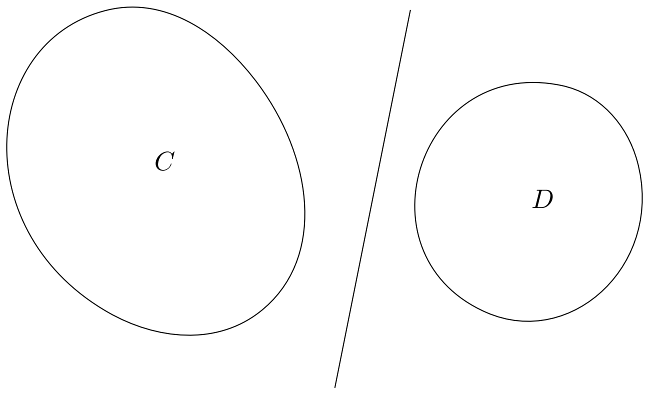
Theorem 95 (Separating Hyperplane Theorem)

Let $C, D \subseteq \mathbb{R}^n$ be two nonempty *disjoint* convex sets, i.e., $C \cap D = \emptyset$. Then there exists a hyperplane that separates C and D, i.e., there exists $\vec{a}, \vec{x}_0 \in \mathbb{R}^n$ such that

$$\vec{a}^\top(\vec{x} - \vec{x}_0) \geq 0, \qquad \forall \vec{x} \in C \tag{5.18}$$

$$\vec{a}^\top(\vec{x} - \vec{x}_0) \leq 0, \qquad \forall \vec{x} \in D. \tag{5.19}$$

The mental picture we want to have is the following.



Proof. We prove the theorem in the case C and D are compact. Compactness of a set is a topological property which, in $\mathbb{R}^n$, amounts to the set being bounded (contained in a ball of large finite radius) and containing its boundary points.

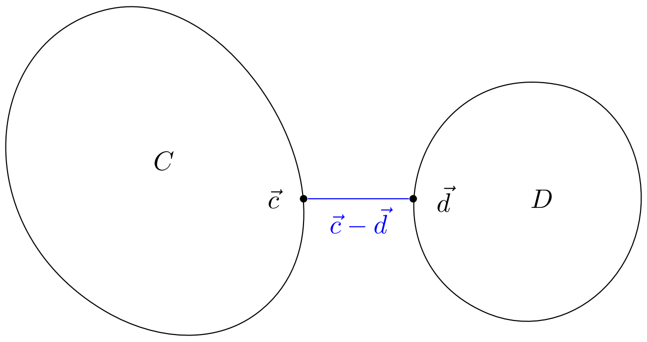
Even though our theorem statement concerns existence of such $\vec{a}$ and $\vec{x}_0$, we will prove it by construction, i.e., we will construct a $\vec{a}$ and $\vec{x}_0$ which separate C and D. This proof strategy is very powerful and will show up frequently.

Since C and D are disjoint, any points in C and D are separated by some positive distance; since they are compact, this distance has a finite lower bound.[1] Define

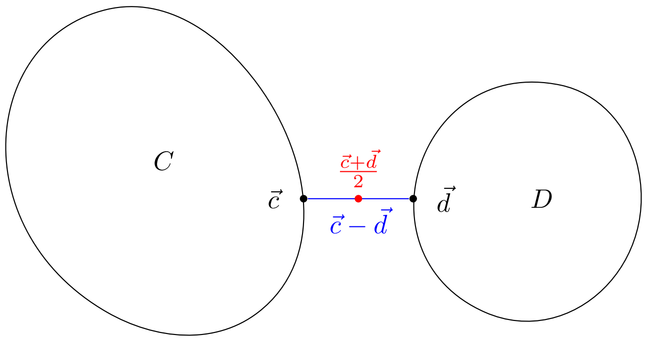
$$\operatorname{dist}(C, D) \doteq \min_{\substack{\vec{c} \in C \\ \vec{d} \in D}} \left\|\vec{c} - \vec{d}\right\|_2 . \tag{5.20}$$

By compactness, there exists some $c \in C$ and $d \in D$ such that $\left\|\vec{c} - \vec{d}\right\|_2 = \operatorname{dist}(C, D)$.[2]

[1]Proving this requires some mathematical analysis and is out of scope of the course.
[2]Same as footnote 1.



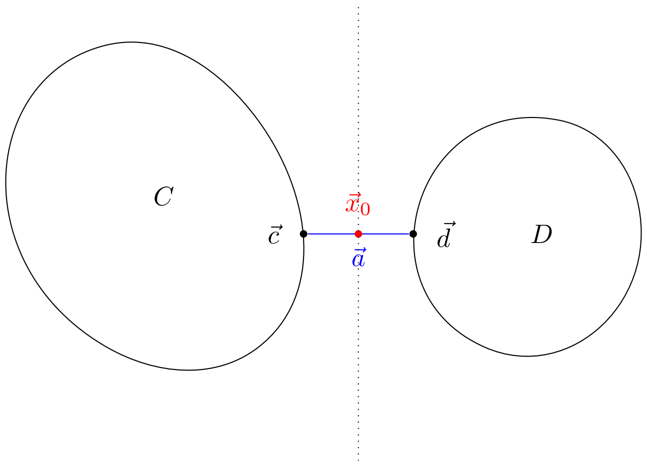
This signals that we want $\vec{c}-\vec{d}$ to be the *normal* vector of our hyperplane — that is, our $\vec{a}$ vector. To find the other point $\vec{x}_0$ which the hyperplane passes through, we can just have it pass through the midpoint of $\vec{c}$ and $\vec{d}$, i.e., $\frac{\vec{c}+\vec{d}}{2}$. This gives the following diagram.



Thus our proposed hyperplane has $\vec{a}$ and $\vec{x}_0$ equal to

$$\vec{a}=\vec{c}-\vec{d}, \qquad \vec{x}_0=\frac{\vec{c}+\vec{d}}{2}. \tag{5.21}$$

It yields the following picture, where the hyperplane is a dotted line.



Notice that there are many separating hyperplanes, such as the one discussed before the theorem. But we just need to prove that this hyperplane separates C and D.

The equation for this hyperplane is

$$\begin{aligned}
\vec{a}^\top(\vec{x}-\vec{x}_0) &= (\vec{c}-\vec{d})^\top\left(\vec{x}-\frac{\vec{c}+\vec{d}}{2}\right) && (5.22)\\
&= (\vec{c}-\vec{d})^\top\vec{x} - \frac{(\vec{c}-\vec{d})^\top(\vec{c}+\vec{d})}{2} && (5.23)\\
&= (\vec{c}-\vec{d})^\top\vec{x} - \frac{\vec{c}^\top\vec{c}-\vec{d}^\top\vec{d}}{2} && (5.24)\\
&= (\vec{c}-\vec{d})^\top\vec{x} - \frac{\|\vec{c}\|_2^2-\left\|\vec{d}\right\|_2^2}{2}. && (5.25)
\end{aligned}$$

Thus the given hyperplane is also available in $(\vec{a}, b)$ form as

$$\vec{a}=\vec{c}-\vec{d}, \qquad b=\frac{\|\vec{c}\|_2^2-\left\|\vec{d}\right\|_2^2}{2}. \tag{5.26}$$

Now we prove that it actually separates $\vec{c}$ and $\vec{d}$. Define $f\colon \mathbb{R}^n \to R$ by

$$f(\vec{x}) = (\vec{c}-\vec{d})^\top\left(\vec{x}-\frac{\vec{c}+\vec{d}}{2}\right). \tag{5.27}$$

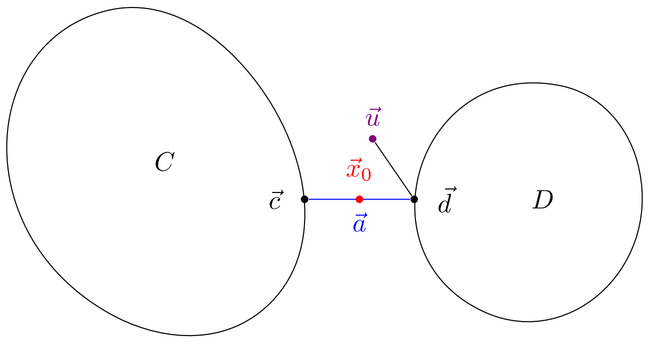
For the sake of contradiction, suppose there exists $\vec{u}\in D$ such that $f(\vec{u})>0$. We can write

$$\begin{aligned}
0 &< f(\vec{u}) && (5.28)\\
&= (\vec{c}-\vec{d})^\top\left(\vec{u}-\frac{\vec{c}+\vec{d}}{2}\right) && (5.29)\\
&= (\vec{c}-\vec{d})^\top\left(\vec{u}-\vec{d}-\frac{\vec{c}-\vec{d}}{2}\right) && (5.30)\\
&= (\vec{c}-\vec{d})^\top(\vec{u}-\vec{d}) - \frac{(\vec{c}-\vec{d})^\top(\vec{c}-\vec{d})}{2} && (5.31)\\
&= (\vec{c}-\vec{d})^\top(\vec{u}-\vec{d}) - \frac{1}{2}\left\|\vec{c}-\vec{d}\right\|_2^2. && (5.32)
\end{aligned}$$

Thus

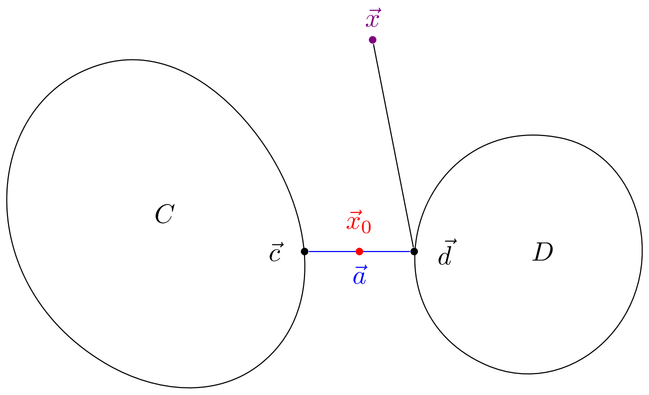
$$0 < (\vec{c}-\vec{d})^\top(\vec{u}-\vec{d}) - \frac{1}{2}\left\|\vec{c}-\vec{d}\right\|_2^2 < (\vec{c}-\vec{d})^\top(\vec{u}-\vec{d}). \tag{5.33}$$

This means that $\vec{c}-\vec{d}$ and $\vec{u}-\vec{d}$ form an *acute* angle. It also means that $\vec{u}\neq\vec{d}$, since otherwise the dot product would be 0. Going back to our picture, this means that $\vec{u}$ would have to be positioned similarly to the following:



At least from the diagram, it seems hard to imagine a $\vec{u} \in D$ such that $\vec{u} - \vec{d}$ and $\vec{c} - \vec{d}$ form an acute angle. Namely, any vector $\vec{x} \in \mathbb{R}^n$ (of reasonably small norm, such as the $\vec{u}$ in the figure) such that $\vec{x} - \vec{d}$ and $\vec{c} - \vec{d}$ form an acute angle, seems to be closer to $\vec{c}$ than $\vec{d}$ is to $\vec{c}$.

Why do we need the "reasonably small norm" condition? Consider the following possible $\vec{x}$:



Certainly, this $\vec{x}$ is farther from $\vec{c}$ than $\vec{d}$ is, and so no contradiction would be derived.

If we can prove that our $\vec{u}$, which we assume is in D, is closer to $\vec{c}$ than $\vec{d}$ is, then we can derive a contradiction with the fact that $\vec{d}$ is the closest vector in D to $\vec{c}$. But we can't prove this for our $\vec{u}$ directly, because $\left\|\vec{u} - \vec{d}\right\|_2$ may be large as in the above figure, so instead we take another vector $\vec{x}$ which is close to $\vec{d}$, where the displacement between $\vec{x}$ and $\vec{d}$ points in the direction of $\vec{u}$. We will show that this $\vec{x}$ is in D yet is closer to $\vec{c}$ than $\vec{d}$ is, thus deriving a contradiction.

Here are the details. Let $\vec{p}\colon [0,1] \to \mathbb{R}^n$ trace out the line from $\vec{d}$ to $\vec{u}$; namely, let $\vec{p}(t) = \vec{d} + t(\vec{u} - \vec{d}) = t\vec{u} + (1-t)\vec{d}$. Since $\vec{u}, \vec{d} \in D$ by assumption, and D is convex, we have that $\vec{p}(t) \in D$ for all $t \in [0,1]$. Now we see that

$$\begin{aligned}
\|\vec{p}(t) - \vec{c}\|_2^2 &= \left\|\vec{d} + t(\vec{u} - \vec{d}) - \vec{c}\right\|_2^2 \\
&= \left\|(\vec{d} - \vec{c}) + t(\vec{u} - \vec{d})\right\|_2^2 \\
&= \left\|\vec{d} - \vec{c}\right\|_2^2 + 2t(\vec{d} - \vec{c})^\top(\vec{u} - \vec{d}) + t^2\left\|\vec{u} - \vec{d}\right\|_2^2 \\
&= \left\|\vec{c} - \vec{d}\right\|_2^2 - 2t(\vec{c} - \vec{d})^\top(\vec{u} - \vec{d}) + t^2\left\|\vec{u} - \vec{d}\right\|_2^2.
\end{aligned}$$

We want to show that there exists t such that $\|\vec{p}(t) - \vec{c}\|_2^2 < \left\|\vec{c} - \vec{d}\right\|_2^2$, i.e.,

$$-2t(\vec{c} - \vec{d})^\top(\vec{u} - \vec{d}) + t^2\left\|\vec{u} - \vec{d}\right\|_2^2 < 0, \tag{5.34}$$

i.e.,

$$2\underbrace{(\vec{c} - \vec{d})^\top(\vec{u} - \vec{d})}_{>0} - t\left\|\vec{u} - \vec{d}\right\|_2^2 > 0. \tag{5.35}$$

Now for all $0 < t < \frac{2(\vec{c} - \vec{d})^\top(\vec{u} - \vec{d})}{\left\|\vec{u} - \vec{d}\right\|_2^2}$, i.e., t small enough, the above inequality holds, so we have for this t that

$$\begin{aligned}
\|\vec{p}(t) - \vec{c}\|_2^2 &= \left\|\vec{c} - \vec{d}\right\|_2^2 - 2t(\vec{c} - \vec{d})^\top(\vec{u} - \vec{d}) + t^2\left\|\vec{u} - \vec{d}\right\|_2^2 \\
&< \left\|\vec{c} - \vec{d}\right\|_2^2.
\end{aligned}$$

However, $\vec{p}(t) \in D$, a contradiction. □

5.2 Convex Functions

Definition 96 (Convex Function)

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a function. We say that f is *convex* if the domain of f, say Ω, is convex, and, for any $\vec{x}_1, \vec{x}_2 \in \Omega$ and $\theta \in [0,1]$, we have

$$f(\theta\vec{x}_1 + (1-\theta)\vec{x}_2) \leq \theta f(\vec{x}_1) + (1-\theta)f(\vec{x}_2). \tag{5.36}$$

A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *concave* if $-f$ is convex.

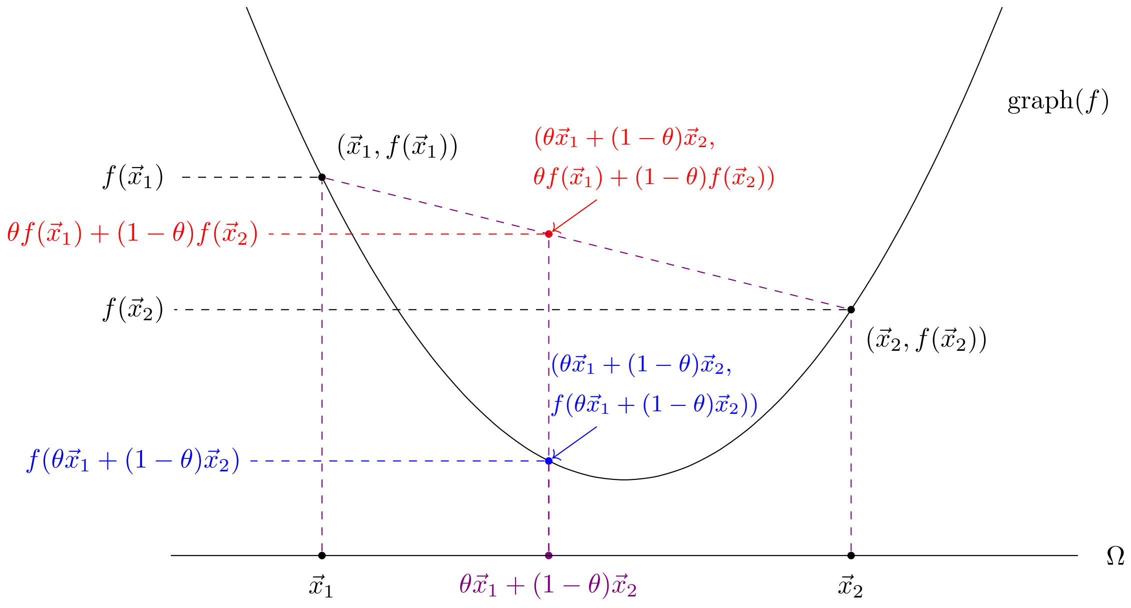
Equation (5.36) is also called *Jensen's inequality* and is equivalent to the following, seemingly more general statement.

Theorem 97 (Jensen's Inequality)

Let Ω be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a convex function. Let $\vec{x}_1, \dots, \vec{x}_k \in \Omega$, and let $\theta_1, \dots, \theta_k \in [0,1]$ such that $\sum_{i=1}^k \theta_i = 1$. Then

$$f\left(\sum_{i=1}^k \theta_i \vec{x}_i\right) \leq \sum_{i=1}^k \theta_i f(\vec{x}_i). \tag{5.37}$$

We can think about this result in terms of a picture.


graph(f)
$(\vec{x}_1, f(\vec{x}_1))$
$(\theta\vec{x}_1 + (1-\theta)\vec{x}_2,$
$\theta f(\vec{x}_1) + (1-\theta)f(\vec{x}_2))$
$f(\vec{x}_1)$
$\theta f(\vec{x}_1) + (1-\theta)f(\vec{x}_2)$
$f(\vec{x}_2)$
$(\vec{x}_2, f(\vec{x}_2))$
$(\theta\vec{x}_1 + (1-\theta)\vec{x}_2,$
$f(\theta\vec{x}_1 + (1-\theta)\vec{x}_2))$
$f(\theta\vec{x}_1 + (1-\theta)\vec{x}_2)$
Ω
$\vec{x}_1$
$\theta\vec{x}_1 + (1-\theta)\vec{x}_2$
$\vec{x}_2$

The prototypical convex function has a "bowl-shaped" graph, and taking a weighted average of two (or any finite number) of points will mean we land in the bowl. In particular, taking a weighted average of any number of function values $f(\vec{x}_1), \dots, f(\vec{x}_k)$ will always give a larger number than applying the function f to the same weighted averages of the points $\vec{x}_1, \dots, \vec{x}_k$. Put more simply, if f is convex then the chord joining the points $(\vec{x}_1, f(\vec{x}_1))$ and $(\vec{x}_2, f(\vec{x}_2))$ always lies *above* the graph of f. Similarly, if f is concave then the chord joining the points $(\vec{x}_1, f(\vec{x}_1))$ and $(\vec{x}_2, f(\vec{x}_2))$ always lies *below* the graph of f.

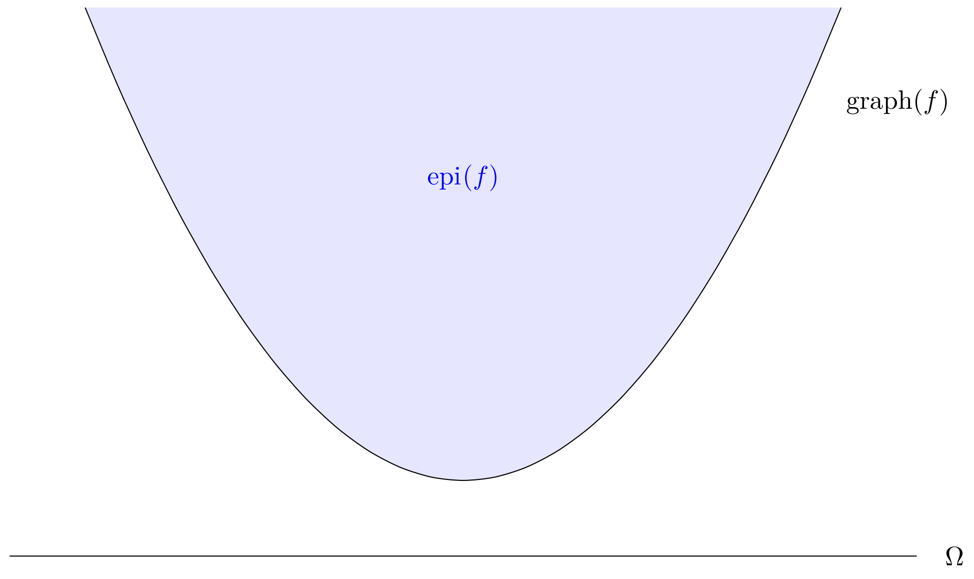
From the picture, it may be intuitively clear that it is hard to construct convex functions f with multiple global minima. We will come back to this idea later.

It may be useful to connect the notion of convex function and convex set. For this, we will define the epigraph.

Definition 98 (Epigraph)
Let Ω be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a convex function. The *epigraph* of f, denoted $\operatorname{epi}(f) \subseteq \Omega \times \mathbb{R}$, is defined as

$$\operatorname{epi}(f) = \{(\vec{x}, t) \mid \vec{x} \in \Omega, t \geq f(\vec{x})\}. \tag{5.38}$$

Geometrically, the epigraph is all points in $\Omega \times \mathbb{R}$ that lie above the graph of f.



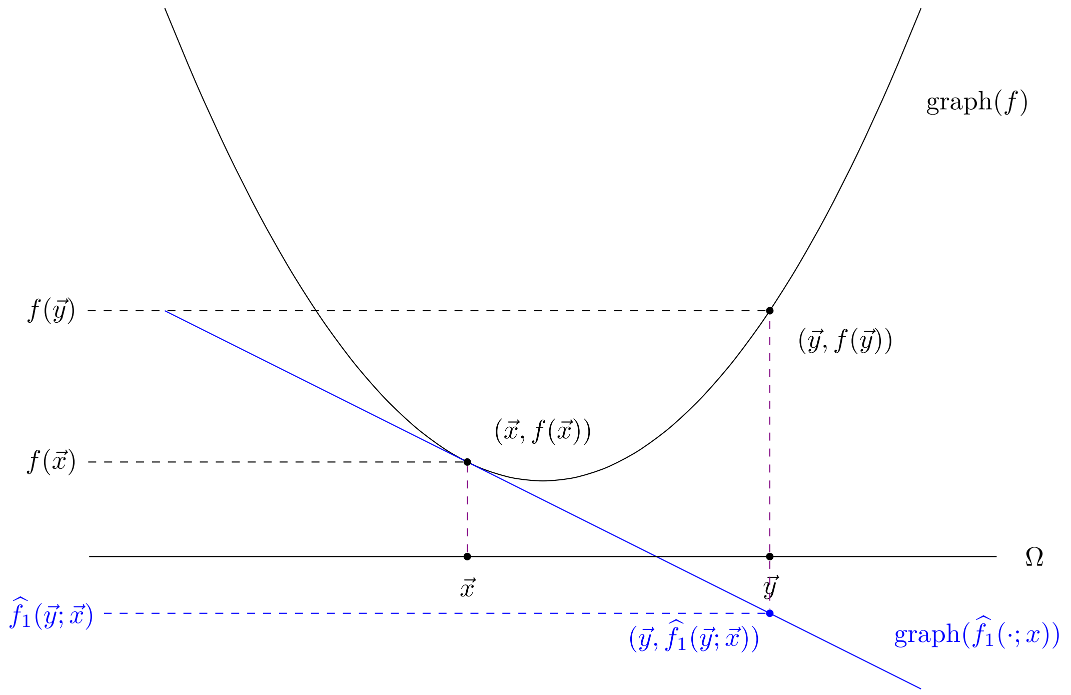
Proposition 99
Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a function. Then f is a convex function if and only if $\operatorname{epi}(f)$ is a convex set.

Proof. Left as an exercise. $\square$

Theorem 100 (First-Order Condition for Convexity)
Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a differentiable function. Then f is convex if and only if for all $\vec{x}, \vec{y} \in \Omega$, we have

$$f(\vec{y}) \geq f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x}). \tag{5.39}$$

Note that this latter term is the first-order Taylor expansion of f around $\vec{x}$ evaluated at $\vec{y}$, i.e., $\widehat{f}_1(\vec{y}; \vec{x}) = f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x})$. The graph of $\widehat{f}_1(\cdot; \vec{x})$ is the tangent line to the graph of f at the point $(\vec{x}, f(\vec{x}))$. So another characterization of convex functions is that their graphs lie above their tangent lines.


graph(f)
$f(\vec{y})$
$(\vec{y}, f(\vec{y}))$
$(\vec{x}, f(\vec{x}))$
$f(\vec{x})$
Ω
$\vec{x}$
$\vec{y}$
$\widehat{f}_1(\vec{y};\vec{x})$
$(\vec{y}, \widehat{f}_1(\vec{y};\vec{x}))$
graph$(\widehat{f}_1(\cdot;x))$

Proof. First suppose f is convex. Then for any $h \in (0,1)$, we have

$$\begin{aligned}
f(h\vec{y}+(1-h)\vec{x}) &\leq hf(\vec{y})+(1-h)f(\vec{x}) && (5.40)\\
&= hf(\vec{y})+f(\vec{x})-hf(\vec{x}) && (5.41)\\
&= f(\vec{x})+h(f(\vec{y})-f(\vec{x})) && (5.42)\\
\implies f(h\vec{y}+(1-h)\vec{x})-f(\vec{x}) &\leq h(f(\vec{y})-f(\vec{x})) && (5.43)\\
\implies \frac{f(h\vec{y}+(1-h)\vec{x})-f(\vec{x})}{h} &\leq f(\vec{y})-f(\vec{x}) && (5.44)\\
\implies f(\vec{y}) &\geq f(\vec{x})+\frac{f(h\vec{y}+(1-h)\vec{x})-f(\vec{x})}{h} && (5.45)\\
&= f(\vec{x})+\frac{f(\vec{x}+h(\vec{y}-\vec{x}))}{h}. && (5.46)
\end{aligned}$$

To summarize, for any $h \in (0,1)$ we have that

$$f(\vec{y}) \geq f(\vec{x})+\frac{f(\vec{x}+h(\vec{y}-\vec{x}))}{h}. \tag{5.47}$$

Taking the limit $h \to 0$ on both sides, we get

$$\begin{aligned}
f(\vec{y}) &\geq \lim_{h\to 0}\left\{f(\vec{x})+\frac{f(\vec{x}+h(\vec{y}-\vec{x}))}{h}\right\} && (5.48)\\
&= f(\vec{x})+\lim_{h\to 0}\frac{f(\vec{x}+h(\vec{y}-\vec{x}))}{h} && (5.49)\\
&= f(\vec{x})+[\nabla f(\vec{x})]^\top(\vec{y}-\vec{x}) && (5.50)
\end{aligned}$$

as desired.

For the other direction, let $\theta \in [0,1]$ and let $\vec{z} = \theta\vec{x}+(1-\theta)\vec{y}$. We have

$$f(\vec{x}) \geq f(\vec{z})+[\nabla f(\vec{z})]^\top(\vec{x}-\vec{z}) \tag{5.51}$$

$$f(\vec{y}) \geq f(\vec{z}) + [\nabla f(\vec{z})]^\top (\vec{y} - \vec{z}). \tag{5.52}$$

Adding θ times the first equation to $(1-\theta)$ times the second equation, we get

$$\begin{aligned}
\theta f(\vec{x}) + (1-\theta) f(\vec{y}) &\geq \theta f(\vec{z}) + (1-\theta) f(\vec{z}) + \theta [\nabla f(\vec{z})]^\top (\vec{x} - \vec{z}) + (1-\theta)[\nabla f(\vec{z})]^\top (\vec{y} - \vec{z}) && (5.53)\\
&= f(\vec{z}) + [\nabla f(\vec{z})]^\top (\theta \vec{x} + (1-\theta)\vec{y} - \vec{z}) && (5.54)\\
&= f(\vec{z}) + [\nabla f(\vec{z})]^\top (\vec{z} - \vec{z}) && (5.55)\\
&= f(\vec{z}) + [\nabla f(\vec{z})]^\top \vec{0} && (5.56)\\
&= f(\vec{z}) && (5.57)\\
&= f(\theta \vec{x} + (1-\theta)\vec{y}). && (5.58)
\end{aligned}$$

□

We also have a corresponding second-order condition.

Theorem 101 (Second-Order Condition for Convexity)

Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f : \Omega \to \mathbb{R}$ be a twice-differentiable function. Then f is convex if and only if for all $\vec{x} \in \Omega$, we have

$$\nabla^2 f(\vec{x}) \succeq 0, \tag{5.59}$$

i.e., $\nabla^2 f(\vec{x})$ is PSD for each $\vec{x} \in \Omega$.

Corollary 102. *Let $Q \in \mathbb{S}^n$ be a symmetric matrix, let $\vec{b} \in \mathbb{R}^n$, and let $c \in \mathbb{R}$. The quadratic form*

$$f(\vec{x}) = \vec{x}^\top Q \vec{x} + \vec{b}^\top \vec{x} + c \tag{5.60}$$

is convex if and only if Q is PSD.

Lastly, we identify a strengthened condition of convexity which allows for stronger guarantees later down the line.

Definition 103 (Strictly Convex Function)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We say that f is *strictly convex* if the domain of f, say Ω, is convex, and, for any $\vec{x}_1 \neq \vec{x}_2 \in \Omega$ and $\theta \in (0,1)$, we have

$$f(\theta \vec{x}_1 + (1-\theta)\vec{x}_2) < \theta f(\vec{x}_1) + (1-\theta) f(\vec{x}_2). \tag{5.61}$$

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *strictly concave* if $-f$ is strictly convex.

And correspondingly, we have the first-order and second-order conditions.

Theorem 104 (First-Order Condition for Strict Convexity)

Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f : \Omega \to \mathbb{R}$ be a differentiable function. Then f is strictly convex if and only if for all $\vec{x} \neq \vec{y} \in \Omega$, we have

$$f(\vec{y}) > f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x}). \tag{5.62}$$

Theorem 105 (Second-Order Condition for Strict Convexity)
Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a twice-differentiable function such that $\nabla^2 f(\vec{x}) \succ 0$ for all $\vec{x} \in \Omega$, i.e., $\nabla^2 f(\vec{x})$ is PD for all $\vec{x} \in \Omega$. Then f is strictly convex.

Notice that this is *not* an if-and-only-if! As an example, take the scalar function $f(x) = x^4$. Then $f''(0) = 0$, so it is not true that $f''(x) \succeq 0$ for all $x \in \Omega = \mathbb{R}$, but f is strictly convex.

5.3 Convex Optimization Problems

This section will lay out some of the key properties of the main class of optimization problems we are interested in — convex optimization problems.

Definition 106 (Convex Optimization Problem)
Let $\Omega \subseteq \mathbb{R}^n$ be a set and let $f\colon \Omega \to \mathbb{R}$ be a function. We say that the problem

$$\min_{\vec{x}\in\Omega} f(\vec{x}) \tag{5.63}$$

is a *convex optimization problem* if Ω is a convex set and f is a convex function.

Note that this applies to other kinds of constraint sets too — in particular, those of the "standard" form "$f_i(\vec{x}) \leq 0$ for all i and $h_i(\vec{x}) = 0$ for all j" still define a feasible region Ω and thus can furnish a convex optimization problem. In particular, we have the following result.

Theorem 107 (When is Standard Form Optimization Problem Convex?)
Let $f_1, \ldots, f_m, h_1, \ldots, h_p\colon \mathbb{R}^n \to \mathbb{R}$ be functions. The feasible set

$$\Omega = \left\{ \vec{x} \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} f_i(\vec{x}) \leq 0, & \forall i \in \{1, \ldots, m\} \\ h_j(\vec{x}) = 0, & \forall j \in \{1, \ldots, p\} \end{array} \right\} \tag{5.64}$$

is convex if each f_i is convex and each h_j is affine. Consequently, the problem

$$\begin{aligned} \min_{\vec{x}\in\mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \leq 0, \qquad \forall i \in \{1, \ldots, m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1, \ldots, p\}. \end{aligned} \tag{5.65}$$

is a convex optimization problem if $f_0, f_1, \ldots, f_m$ are convex functions and $h_1, \ldots, h_p$ are affine functions.

Proof. Left as exercise. □

Now, we can establish the first-order condition for optimality within a convex problem. This is one of the main theorems of convex analysis.

Theorem 108 (First-Order Conditions for Optimality in Convex Problem)

Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f \colon \Omega \to \mathbb{R}$ be a differentiable convex function. Let $\vec{x}^\star \in \Omega$ be such that $\nabla f(\vec{x}^\star) = \vec{0}$. Then $\vec{x}^\star \in \operatorname{argmin}_{\vec{x} \in \Omega} f(\vec{x})$, i.e., $\vec{x}^\star$ is a global minimizer of f.

Proof. Let $\vec{y}$ be any other point in Ω. Then

$$f(\vec{y}) \geq f(\vec{x}^\star) + [\nabla f(\vec{x}^\star)]^\top (\vec{y} - \vec{x}^\star) \tag{5.66}$$
$$= f(\vec{x}^\star) + \vec{0}^\top (\vec{y} - \vec{x}^\star) \tag{5.67}$$
$$= f(\vec{x}^\star). \tag{5.68}$$

This proves that $\vec{x}^\star \in \operatorname{argmin}_{\vec{x} \in \Omega} f(\vec{x})$ as desired. □

A generalization of this statement is that all local minimizers are global minimizers.

Theorem 109 (For Convex Functions, Local Minima are Global Minima)
Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f \colon \Omega \to \mathbb{R}$ be a convex function. Let $\vec{x}^\star \in \Omega$ be such that there exists some $\epsilon > 0$ such that if $\vec{x} \in \Omega$ has $\|\vec{x} - \vec{x}^\star\|_2 \leq \epsilon$ then $f(\vec{x}^\star) \leq f(\vec{x})$. Then $\vec{x}^\star$ is a minimizer of f over Ω.

Proof. Left as exercise. □

When is the global minimizer unique? We can justify this using strict convexity.

Theorem 110 (Strictly Convex Functions have Unique Minimizers)
Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f \colon \Omega \to \mathbb{R}$ be a strictly convex function. Then f has *at most* one global minimizer.

Proof. Left as exercise. □

For an example of a strictly convex function with one global minimizer, take $f(x) = x^4$, which is minimized at $x = 0$. For an example of a strictly convex function with no global minimizers, take $f(x) = e^x$.

5.4 Solving Convex Optimization Problems

To construct a first attempt at systematically solving convex optimization problems, we first need to define active and inactive constraints.

Definition 111 (Types of Constraints)
Consider a problem of the form

$$\begin{aligned} \min_{\vec{x} \in \mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \leq 0, \qquad \forall i \in \{1, \ldots, m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1, \ldots, p\}. \end{aligned} \tag{5.69}$$

Fix a feasible $\vec{x}_0 \in \mathbb{R}^n$. The *inequality* constraint $f_k(\vec{x}) \leq 0$ is *active at* $\vec{x}_0$ if $f_k(\vec{x}_0) = 0$, and *inactive at* $\vec{x}_0$ otherwise, i.e., $f_k(\vec{x}_0) < 0$.

We can use this to formulate a strategy for solving convex optimization problems. Recall that for a convex problem $\min_{\vec{x}\in\Omega} f_0(\vec{x})$ which has a solution $\vec{x}^\star$, either $\nabla f(\vec{x}^\star) = \vec{0}$ or $\vec{x}^\star$ is on the boundary of Ω. The boundary of Ω is any point in which any inequality constraint is active. This allows us to systematically find solutions to convex optimization problems.

Problem Solving Strategy 112. *To solve a convex optimization program, we can do the following.*

(i) Iterate through all 2^m subsets $S \subseteq \{1, \dots, m\}$ of constraints which might be active at optimum.

(ii) For each S:

1. *Solve the modified problem*

$$\begin{aligned} \min_{\vec{x}\in\mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) = 0, \qquad \forall i \in S \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1, \dots, p\}, \end{aligned} \tag{5.70}$$

   i.e., solve the problem where you pretend that all inequality constraints in S are met with equality and pretend that the other inequality constraints don't exist. This gives some solutions $\vec{x}_S^\star$.

2. *If there is a solution $\vec{x}_S^\star$ which is feasible for the original problem, write down the value of $f_0(\vec{x}_S^\star)$. Otherwise, ignore it.*

3. *After iterating through all $\vec{x}_S^\star$ which are feasible for the original problem, take the one(s) with the best objective value $f_0(\vec{x}_S^\star)$ as the optimal solution(s) to the original problem.*

Predictably, this problem solving strategy is exponentially hard as the number of inequality constraints increases. Even if solving the "inner" equality-constrained minimization problems is easy (as it often is), the whole procedure is untenable for large-scale problems. In future chapters, we will develop better analytic and algorithmic ways to solve convex optimization problems.

5.5 Problem Transformations and Reparameterizations

In this section, we discuss various transformations, or reparameterizations, of generic optimization problems, which allow us to go between many equivalent formulations of a problem. Some of them will even allow us to turn non-convex problems into convex problems.

5.5.1 Monotone Transformations of the Objective Function

The core of this technique is the following result.

Proposition 113 (Positive Monotone Transformations Don't Affect Optimizers)
Let Ω be any set, let $f_0 \colon \Omega \to \mathbb{R}$ be any function. Define $f_0(\Omega) \doteq \{f_0(\vec{x}) \mid \vec{x} \in \Omega\} \subseteq \mathbb{R}$.

(a) Let $\phi \colon f_0(\Omega) \to \mathbb{R}$ be any monotonically increasing function. Then

$$\operatorname*{argmin}_{\vec{x}\in\Omega} f_0(\vec{x}) = \operatorname*{argmin}_{\vec{x}\in\Omega} \phi(f_0(\vec{x})) \qquad \text{and} \qquad \operatorname*{argmax}_{\vec{x}\in\Omega} f_0(\vec{x}) = \operatorname*{argmax}_{\vec{x}\in\Omega} \phi(f_0(\vec{x})). \tag{5.71}$$

(b) Let $\psi\colon f_0(\Omega) \to \mathbb{R}$ be any monotonically decreasing function. Then

$$\operatorname*{argmin}_{\vec{x}\in\Omega} f_0(\vec{x}) = \operatorname*{argmax}_{\vec{x}\in\Omega} \psi(f_0(\vec{x})) \qquad \text{and} \qquad \operatorname*{argmax}_{\vec{x}\in\Omega} f_0(\vec{x}) = \operatorname*{argmin}_{\vec{x}\in\Omega} \psi(f_0(\vec{x})). \tag{5.72}$$

Proof. We only prove the very first equality; the rest follow similarly. We have

$$\vec{x}^\star \in \operatorname*{argmin}_{\vec{x}\in\Omega} f_0(\vec{x}) \tag{5.73}$$

$$\iff f_0(\vec{x}^\star) \leq f_0(\vec{x}), \qquad \forall \vec{x}\in\Omega \tag{5.74}$$

$$\iff \phi(f_0(\vec{x}^\star)) \leq \phi(f_0(\vec{x})), \qquad \forall \vec{x}\in\Omega \tag{5.75}$$

$$\iff \vec{x}^\star \in \operatorname*{argmin}_{\vec{x}\in\Omega} \phi(f_0(\vec{x})). \tag{5.76}$$

□

The first equality will be by far the most important for us, though the others might also be situationally useful. This proposition is why doing things like squaring the norm in least squares won't affect the solutions we get, i.e.,

$$\operatorname*{argmin}_{\vec{x}\in\mathbb{R}^n} \left\|A\vec{x}-\vec{b}\right\|_2 = \operatorname*{argmin}_{\vec{x}\in\mathbb{R}^n} \left\|A\vec{x}-\vec{b}\right\|_2^2. \tag{5.77}$$

But the fact that ϕ is monotonic *when restricted to* $f_0(\Omega)$ is quite crucial; indeed, we have that

$$\operatorname*{argmin}_{x\in\mathbb{R}} x = \emptyset \qquad \text{but} \qquad \operatorname*{argmin}_{x\in\mathbb{R}} x^2 = \{0\}. \tag{5.78}$$

This is because the function $u \mapsto u^2$ is not monotonic in general, although it is monotonic on the non-negative real numbers $\mathbb{R}_+$. It just so happens that $\|\cdot\|_2$ only outputs non-negative numbers, so actually in the case of least squares we have $f_0(\Omega) = \mathbb{R}_+$ and the proposition applies.

Example 114 (Logistic Regression)**.** A more non-trivial example is logistic regression. First, define $\sigma\colon \mathbb{R} \to (0,1)$ by

$$\sigma(x) \doteq \frac{1}{1+\mathrm{e}^{-x}}. \tag{5.79}$$

Suppose we have data points $\vec{x}_1,\dots,\vec{x}_n \in \mathbb{R}^d$ and accompanying labels $y_1,\dots,y_n \in \{0,+1\}$. Suppose that the conditional probability that $y_i = 1$ given $\vec{x}_i$ is given by

$$\mathbb{P}_{\vec{w}_0}[\vec{y}_i = 1 \mid \vec{x}_i] = \sigma(\vec{x}_i^\top \vec{w}_0) \tag{5.80}$$

for some $\vec{w}_0 \in \mathbb{R}^d$. We wish to recover $\vec{w}_0$, and thus recover the generative model $\mathbb{P}_{\vec{w}_0}[y \mid \vec{x}]$. We do this by maximum likelihood estimation. Writing out the problem, we get

$$\vec{w}^\star \in \operatorname*{argmax}_{\vec{w}\in\mathbb{R}^d} \mathbb{P}_{\vec{w}}[y_1,\dots,y_n \mid \vec{x}_1,\dots,\vec{x}_n] \tag{5.81}$$

$$= \operatorname*{argmax}_{\vec{w}\in\mathbb{R}^d} \prod_{i=1}^n \mathbb{P}_{\vec{w}}[y_i \mid \vec{x}_i] \tag{5.82}$$

$$= \operatorname*{argmax}_{\vec{w}\in\mathbb{R}^d} \left(\prod_{\substack{i=1\\ y_i=0}}^n \mathbb{P}_{\vec{w}}[y_i = 0 \mid \vec{x}_i]\right)\left(\prod_{\substack{i=1\\ y_i=1}}^n \mathbb{P}_{\vec{w}}[y_i = 1 \mid \vec{x}_i]\right) \tag{5.83}$$

$$= \operatorname*{argmax}_{\vec{w}\in\mathbb{R}^d} \prod_{i=1}^n \mathbb{P}_{\vec{w}}[y_i = 1 \mid \vec{x}_i]^{y_i}\, \mathbb{P}_{\vec{w}}[y_i = 0 \mid \vec{x}_i]^{1-y_i} \tag{5.84}$$

$$= \operatorname*{argmax}_{\vec{w} \in \mathbb{R}^d} \prod_{i=1}^{n} \sigma(\vec{x}_i^\top \vec{w})^{y_i} (1 - \sigma(\vec{x}_i^\top \vec{w}))^{1-y_i}. \tag{5.85}$$

Now, the σ function is *very* non-convex. The product of σs is *also* very non-convex. Thus, it seems intractable to solve this problem. But note that the objective function takes values in $(0, 1)$. We use the above proposition with the function $x \mapsto \log(x)$, which is monotonically increasing on $(0, 1)$. We obtain

$$\begin{aligned}
&\operatorname*{argmax}_{\vec{w} \in \mathbb{R}^d} \prod_{i=1}^{n} \sigma(\vec{x}_i^\top \vec{w})^{y_i} (1 - \sigma(\vec{x}_i^\top \vec{w}))^{1-y_i} && (5.86)\\
&= \operatorname*{argmax}_{\vec{w} \in \mathbb{R}^d} \log\left(\prod_{i=1}^{n} \sigma(\vec{x}_i^\top \vec{w})^{y_i} (1 - \sigma(\vec{x}_i^\top \vec{w}))^{1-y_i} \right) && (5.87)\\
&= \operatorname*{argmax}_{\vec{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i \log\left(\sigma(\vec{x}_i^\top \vec{w})\right) + (1 - y_i) \log\left(1 - \sigma(\vec{x}_i^\top \vec{w})\right) \right) && (5.88)\\
&= \operatorname*{argmin}_{\vec{w} \in \mathbb{R}^d} \left\{ -\sum_{i=1}^{n} \left(y_i \log\left(\sigma(\vec{x}_i^\top \vec{w})\right) + (1 - y_i) \log\left(1 - \sigma(\vec{x}_i^\top \vec{w})\right) \right) \right\} && (5.89)\\
&= \operatorname*{argmin}_{\vec{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \left(-y_i \log\left(\sigma(\vec{x}_i^\top \vec{w})\right) - (1 - y_i) \log\left(1 - \sigma(\vec{x}_i^\top \vec{w})\right) \right). && (5.90)
\end{aligned}$$

In the penultimate line we used another one of the equalities in the proposition with the monotonically decreasing function $\psi(x) = -x$. Thus, logistic regression reduces to minimizing the objective function

$$f_0(\vec{w}) = \sum_{i=1}^{n} \left(-y_i \log\left(\sigma(\vec{x}_i^\top \vec{w})\right) - (1 - y_i) \log\left(1 - \sigma(\vec{x}_i^\top \vec{w})\right) \right). \tag{5.91}$$

This is the so-called *cross-entropy* loss.

Computing the gradient and Hessian of this function is an exercise, but the result is

$$\nabla f_0(\vec{w}) = \sum_{i=1}^{n} \vec{x}_i (\sigma(\vec{x}_i^\top \vec{w}) - y_i) \tag{5.92}$$

$$\nabla^2 f_0(\vec{w}) = \sum_{i=1}^{n} \sigma(\vec{x}_i^\top \vec{w})(1 - \sigma(\vec{x}_i^\top \vec{w})) \vec{x}_i \vec{x}_i^\top. \tag{5.93}$$

Because $\sigma(x) \in (0, 1)$, the above Hessian is a non-negative weighted sum of positive semidefinite matrices $\vec{x}_i \vec{x}_i^\top$ and is thus positive semidefinite. By the second order conditions, f_0 is convex! Thus we have turned an extremely non-convex problem into an unconstrained convex minimization problem just by a neat application of monotone functions. We can efficiently solve this problem algorithmically using convex optimization solvers such as gradient descent.

Actually, we can do better than gradient descent for this particular example! If we define

$$X = \begin{bmatrix} \vec{x}_1^\top \\ \vdots \\ \vec{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}, \qquad \vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n, \qquad \vec{p}(\vec{w}) = \begin{bmatrix} \sigma(\vec{x}_1^\top \vec{w}) \\ \vdots \\ \sigma(\vec{x}_n^\top \vec{w}) \end{bmatrix} \in \mathbb{R}^n \tag{5.94}$$

then the gradient looks like

$$\nabla f_0(\vec{w}) = X^\top (\vec{p}(\vec{w}) - \vec{y}). \tag{5.95}$$

Notice the similarity to the gradient of least squares, $X^\top (X\vec{w} - \vec{y})$. In fact, we can exploit this similarity to obtain a specialized optimization algorithm called *iteratively reweighted least squares*, which can efficiently solve for maximum likelihood estimates within this type of statistical model (i.e., a *generalized linear model*).

5.5.2 Slack Variables

The purpose of slack variables is to turn equality constraints into inequality constraints and vice-versa.

Definition 115 (Slack Variables)
Consider a problem of the form

$$\begin{aligned} \min_{\vec{x}\in\mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \le 0, \qquad \forall i \in \{1,\dots,m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1,\dots,p\}. \end{aligned} \tag{5.96}$$

Let $\mathcal{S} \subseteq \{1,\dots,m\}$ be any subset. The above problem is equivalent to the following problem:

$$\begin{aligned} \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \vec{s}\in\mathbb{R}^{\mathcal{S}}_+}} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) + s_i = 0, \qquad \forall i \in \mathcal{S} \\ & f_i(\vec{x}) \le 0, \qquad \forall i \in \{1,\dots,m\} \setminus \mathcal{S} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1,\dots,p\}. \end{aligned} \tag{5.97}$$

Here the notation $\mathbb{R}^{\mathcal{S}}_+ = \{(x_i)_{i\in\mathcal{S}} \mid x_i \ge 0 \; \forall i \in \mathcal{S}\}$, and $\vec{s}$ is called a *slack variable*.

One can choose to create slack variables s_i for only a subset of the inequality constraints, or all of them. When we work with more advanced optimization algorithms later, sometimes this parameterization is crucial (e.g. for equality-constrained Newton's method).

Example 116. If we have a problem of the form

$$\min_{\vec{x}\in\mathbb{R}^n} \quad f_0(\vec{x}) \tag{5.98}$$

$$\text{s.t.} \quad 3x_1^2 + 4x_2^2 \le 0 \tag{5.99}$$

$$2x_1^2 + 5x_2^2 \le 0 \tag{5.100}$$

but our solver could only handle equality constraints, then it would be equivalent to solve the problem

$$\min_{\substack{\vec{x}\in\mathbb{R}^2 \\ \vec{s}\in\mathbb{R}^2_+}} \quad f_0(\vec{x}) \tag{5.101}$$

$$\text{s.t.} \quad 3x_1^2 + 4x_2^2 + s_1 = 0 \tag{5.102}$$

$$2x_1^2 + 5x_2^2 + s_2 = 0 \tag{5.103}$$

and, upon solving this problem and obtaining $(\vec{x}^\star, \vec{s}^\star)$, the solution to the original problem would be this same $\vec{x}^\star$.

5.5.3 Epigraph Reformulation

The epigraph reformulation is a way to convert between non-linearity in the objective and a constraint.

Definition 117

Consider a problem of the form

$$\begin{aligned} \min_{\vec{x}\in\mathbb{R}^n} \quad & f_0(\vec{x}) && (5.104) \\ \text{s.t.} \quad & f_i(\vec{x}) \le 0, \qquad \forall i \in \{1,\dots,m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1,\dots,p\}. \end{aligned}$$

Its *epigraph reformulation* is the problem

$$\begin{aligned} \min_{\substack{t\in\mathbb{R}\\ \vec{x}\in\mathbb{R}^n}} \quad & t && (5.105) \\ \text{s.t.} \quad & t \ge f_0(\vec{x}) \\ & f_i(\vec{x}) \le 0, \qquad \forall i \in \{1,\dots,m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1,\dots,p\}. \end{aligned}$$

The epigraph objective is always a linear and differentiable function of the decision variables $(t, \vec{x})$. However, the constraint can become complicated if $f_0(\vec{x})$ is non-linear. This transformation is especially useful in the case of quadratically-constrained quadratic programs (QCQP).

Example 118 (Elastic-Net Regularization)**.** This example uses the two previously-discussed techniques in tandem to figure out how to handle a regularizer with both smooth and non-smooth components (i.e., ℓ^1 and ℓ^2 norms).

Let $A \in \mathbb{R}^{m\times n}$, and $\vec{y} \in \mathbb{R}^n$. Suppose that we have a problem of the form

$$\min_{\vec{x}\in\mathbb{R}^n} \left\{ \|A\vec{x}-\vec{y}\|_2^2 + \alpha\|\vec{x}\|_2^2 + \beta\|\vec{x}\|_1 \right\}. \qquad (5.106)$$

The regularizer $\alpha\|\vec{x}\|_2^2 + \beta\|\vec{x}\|_1$ is called the *elastic net regularizer* and encourages "sparse" and small $\vec{x}$; this regularizer has some use in the analysis of high-dimensional and structured data.

Suppose that our solver can only handle differentiable objectives, but is able to handle constraints so long as they are also differentiable. Then we cannot solve the problem out-right using our solver, so we need to reformulate it. We can first start by using a modification of the epigraph reformulation:

$$\begin{aligned} \min_{\substack{t\in\mathbb{R}\\ \vec{x}\in\mathbb{R}^n}} \quad & \|A\vec{x}-\vec{y}\|_2^2 + \alpha\|\vec{x}\|_2^2 + \beta t && (5.107) \\ \text{s.t.} \quad & t \ge \|\vec{x}\|_1 . && (5.108) \end{aligned}$$

Now the constraint is non-differentiable, so we are no longer able to exactly solve this constrained problem. However, the objective is now convex and differentiable. Let us rewrite this problem using the $|x_i|$:

$$\begin{aligned} \min_{\substack{t\in\mathbb{R}\\ \vec{x}\in\mathbb{R}^n}} \quad & \|A\vec{x}-\vec{y}\|_2^2 + \alpha\|\vec{x}\|_2^2 + \beta t && (5.109) \\ \text{s.t.} \quad & t \ge \sum_{i=1}^{n} |x_i| . && (5.110) \end{aligned}$$

The main insight that goes into resolving the non-differentiability of this constraint is that $|x_i| = \max\{x_i, -x_i\}$ and in particular $s_i \ge |x_i|$ if and only if $s_i \ge x_i$ and $s_i \ge -x_i$. Thus we add more "slack-type" variables and obtain

$$\min_{\substack{t\in\mathbb{R}\\ \vec{x}\in\mathbb{R}^n\\ \vec{s}\in\mathbb{R}^n_+}} \quad \|A\vec{x}-\vec{y}\|_2^2 + \alpha\|\vec{x}\|_2^2 + \beta t \qquad (5.111)$$

$$\text{s.t.} \quad t \geq \sum_{i=1}^{n} s_i \tag{5.112}$$

$$s_i \geq x_i, \qquad \forall i \in \{1, \ldots, n\} \tag{5.113}$$

$$s_i \geq -x_i, \qquad \forall i \in \{1, \ldots, n\}. \tag{5.114}$$

Now the constraints are all differentiable (in fact, affine) and the problem may be solved. This problem is exactly equivalent to the above problem because t is being minimized in the objective, so each s_i is being minimized by way of the first constraint, meaning that $s_i = |x_i|$ and this is equivalent to the original elastic-net regression problem.

Chapter 6

Gradient Descent

Relevant sections of the textbooks:

- [2] Section 12.2.

6.1 Strong Convexity and Smoothness

In this section, we will introduce two properties of functions that will become useful in analyzing optimization algorithms. The first property is a notion of convexity of functions that is stronger than the ones previously introduced.

Definition 119 (μ-Strongly Convex Function)
Let $\mu \geq 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. We say that f is *μ-strongly convex* if the domain of f, say Ω, is convex, and, for any $\vec{x}, \vec{y} \in \Omega$, we have

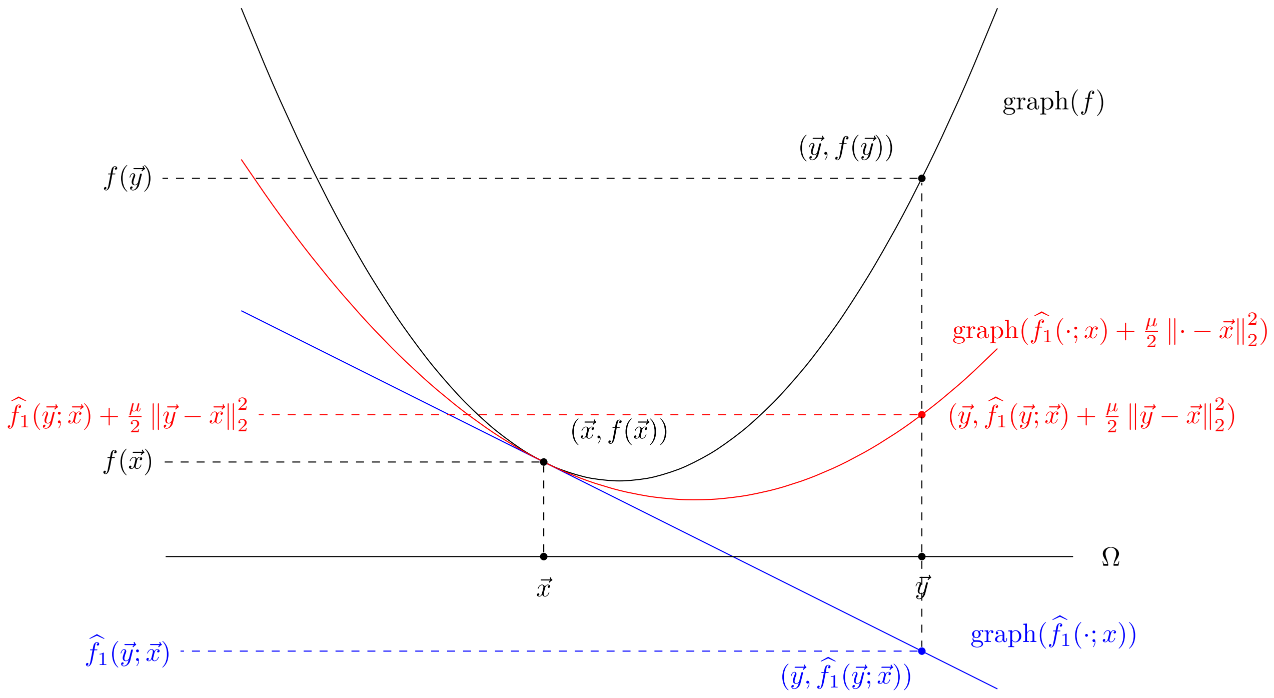
$$f(\vec{y}) \geq f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x}) + \frac{\mu}{2} \|\vec{y} - \vec{x}\|_2^2 . \tag{6.1}$$

A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is *μ-strongly concave* if $-f$ is μ-strongly convex.

Recall from Theorem 100 that the first order condition for (usual) convexity requires the function to be bigger than its linear (first order) Taylor approximation centered at any point. μ-strong convexity imposes a stronger requirement on the function: it needs to be bigger than its linear approximation plus a non-negative quadratic term that has a Hessian matrix μI. This becomes more obvious if we write Equation (6.1) in the equivalent form

$$f(\vec{y}) \geq \underbrace{f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x})}_{\text{first-order Taylor approximation}} + \underbrace{(\vec{y} - \vec{x})^\top \left(\frac{\mu}{2} I\right) (\vec{y} - \vec{x})}_{\text{non-negative quadratic term}} . \tag{6.2}$$

Below, we visualize the μ-strong convexity property and compare it to the first order condition for convexity.


graph(f)
$(\vec{y}, f(\vec{y}))$
$f(\vec{y})$
graph$(\widehat{f}_1(\cdot; x) + \frac{\mu}{2}\|\cdot - \vec{x}\|_2^2)$
$\widehat{f}_1(\vec{y};\vec{x}) + \frac{\mu}{2}\|\vec{y}-\vec{x}\|_2^2$
$(\vec{x}, f(\vec{x}))$
$(\vec{y}, \widehat{f}_1(\vec{y};\vec{x}) + \frac{\mu}{2}\|\vec{y}-\vec{x}\|_2^2)$
$f(\vec{x})$
Ω
$\vec{x}$
$\vec{y}$
$\widehat{f}_1(\vec{y};\vec{x})$
graph$(\widehat{f}_1(\cdot; x))$
$(\vec{y}, \widehat{f}_1(\vec{y};\vec{x}))$

Therefore, μ-strong convexity of the function is a very important feature. It guarantees that the function will always have enough *curvature* (at least as much as its quadratic lower bound) and thus will never become too flat anywhere on its domain.

If the function f is twice-differentiable we can formalize this notion by giving the following equivalent condition for μ-strong convexity.

Theorem 120 (Second Order Condition for μ-Strong Convexity)
Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a twice-differentiable function. Then f is μ-strongly convex if and only if for all $\vec{x} \in \Omega$ we have

$$\nabla^2 f(\vec{x}) - \mu I \succeq 0, \tag{6.3}$$

i.e., $\nabla^2 f(\vec{x}) - \mu I$ is PSD for each $\vec{x} \in \Omega$.

An important property of μ-strongly convex functions is that they are strictly convex and thus they have at most one minimizer. In fact, one can show that they have exactly one minimizer.

Theorem 121 (Strongly Convex Functions have Unique Global Minimizers)
Let $\mu > 0$. Let $\Omega \subseteq \mathbb{R}^n$ be a convex set and let $f\colon \Omega \to \mathbb{R}$ be a μ-strongly convex function. Then f has *exactly one* global minimizer.

The second property we want to introduce is L-smoothness, which describes quadratic *upper* bounds on the function f.

Definition 122 (L-Smooth Function)
Let $L \geq 0$. Let $\Omega \subseteq \mathbb{R}^n$ be a set. Let $f\colon \Omega \to \mathbb{R}$ be a differentiable function. We say that f is *L-smooth* if, for

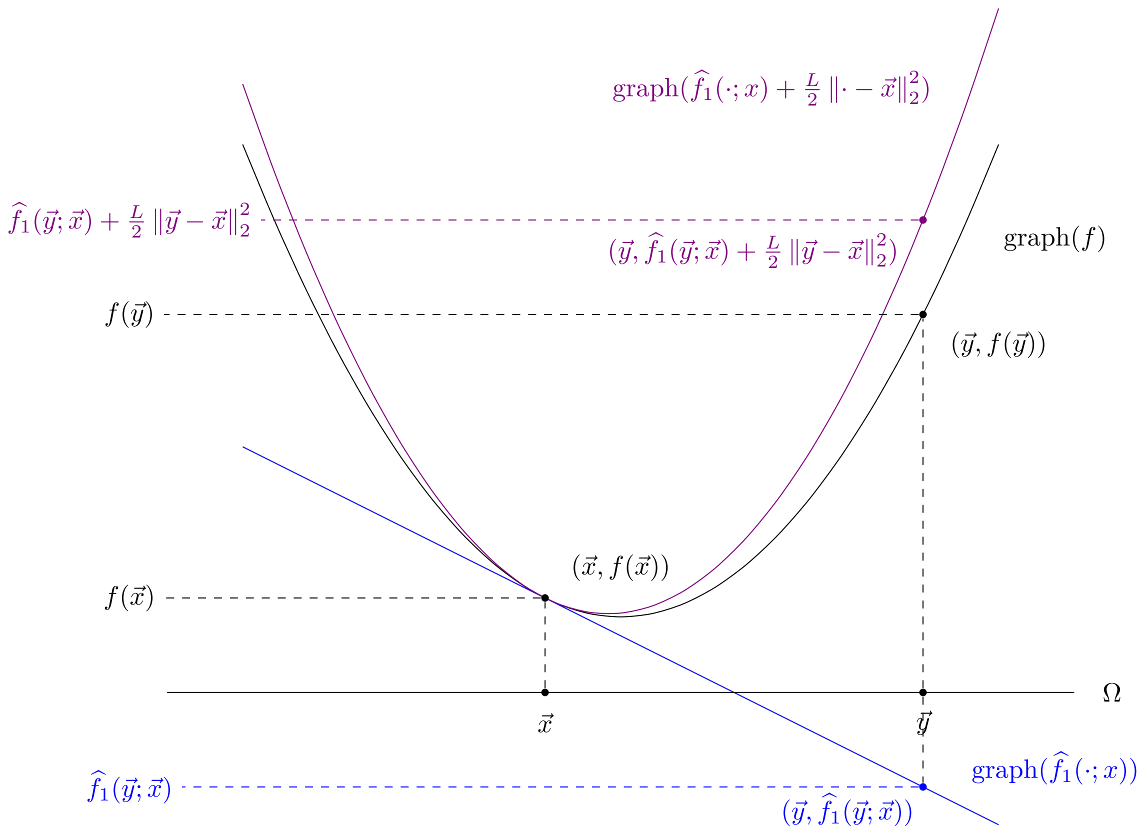
any $\vec{x}, \vec{y} \in \Omega$, we have

$$f(\vec{y}) \leq f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x}) + \frac{L}{2} \|\vec{y} - \vec{x}\|_2^2 . \tag{6.4}$$

If f is L-smooth, then the function f is *upper* bounded by its first order Taylor approximation plus a non-negative quadratic term:

$$f(\vec{y}) \leq \underbrace{f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x})}_{\text{first-order Taylor approximation}} + \underbrace{(\vec{y} - \vec{x})^\top \left(\frac{L}{2} I\right) (\vec{y} - \vec{x})}_{\text{non-negative quadratic term}} . \tag{6.5}$$

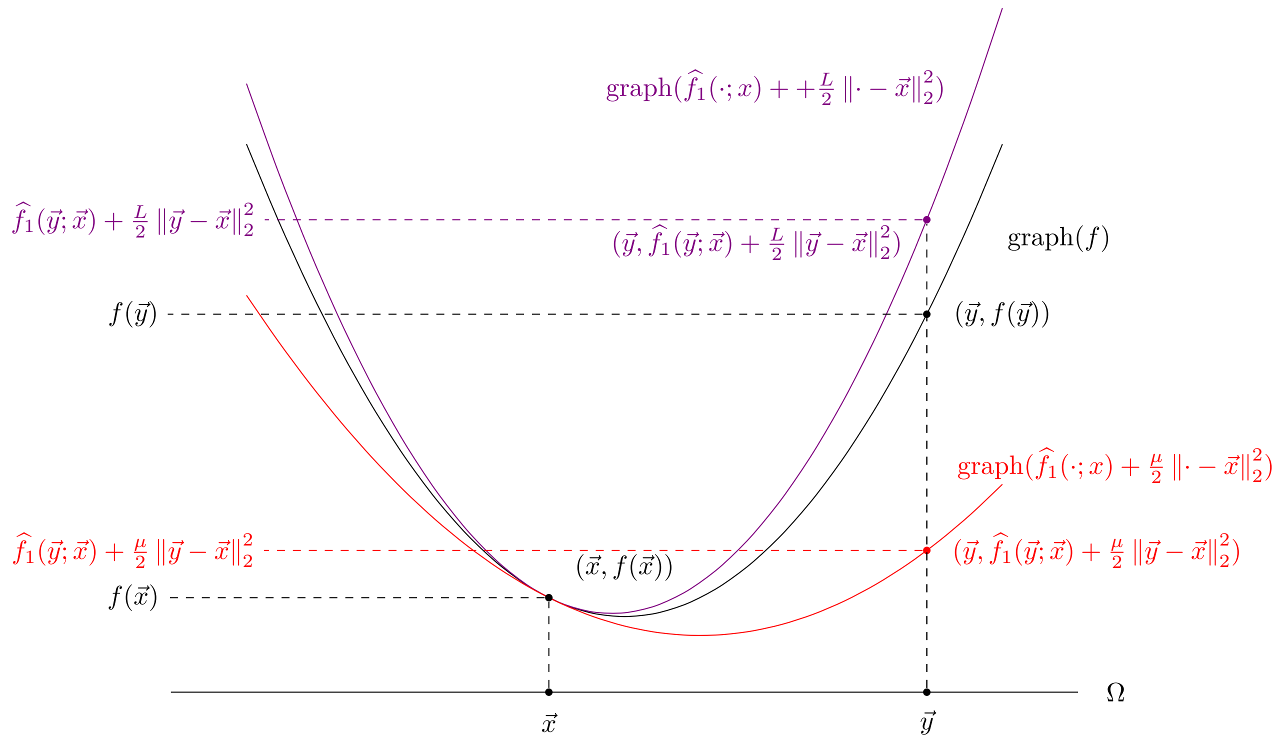
We can visualize the L-smoothness condition similarly to the strongly convex condition, as below:



$\text{graph}(\widehat{f}_1(\cdot; x) + \frac{L}{2} \|\cdot - \vec{x}\|_2^2)$
$\widehat{f}_1(\vec{y}; \vec{x}) + \frac{L}{2} \|\vec{y} - \vec{x}\|_2^2$
$(\vec{y}, \widehat{f}_1(\vec{y}; \vec{x}) + \frac{L}{2} \|\vec{y} - \vec{x}\|_2^2)$
$\text{graph}(f)$
$f(\vec{y})$
$(\vec{y}, f(\vec{y}))$
$(\vec{x}, f(\vec{x}))$
$f(\vec{x})$
Ω
$\vec{x}$
$\vec{y}$
$\text{graph}(\widehat{f}_1(\cdot; x))$
$\widehat{f}_1(\vec{y}; \vec{x})$
$(\vec{y}, \widehat{f}_1(\vec{y}; \vec{x}))$

L-smoothness provides a quadratic upper bound on f. This upper bound ensures that the function doesn't have too much curvature anywhere on its domain (at most as much as its upper bound). We will see later in this chapter that this actually translates into an upper bound on the rate at which the gradient of the function changes.

Finally, we visualize the behavior of the μ-strongly convex and L-smooth bounds together.


$\mathrm{graph}(\widehat{f}_1(\cdot;x) + +\frac{L}{2}\|\cdot - \vec{x}\|_2^2)$
$\widehat{f}_1(\vec{y};\vec{x}) + \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2$
$(\vec{y}, \widehat{f}_1(\vec{y};\vec{x}) + \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2)$
$\mathrm{graph}(f)$
$f(\vec{y})$
$(\vec{y}, f(\vec{y}))$
$\mathrm{graph}(\widehat{f}_1(\cdot;x) + \frac{\mu}{2}\|\cdot - \vec{x}\|_2^2)$
$\widehat{f}_1(\vec{y};\vec{x}) + \frac{\mu}{2}\|\vec{y} - \vec{x}\|_2^2$
$(\vec{x}, f(\vec{x}))$
$(\vec{y}, \widehat{f}_1(\vec{y};\vec{x}) + \frac{\mu}{2}\|\vec{y} - \vec{x}\|_2^2)$
$f(\vec{x})$
Ω
$\vec{x}$
$\vec{y}$

6.2 Gradient Descent

We now have all the tools we need to introduce, understand, and analyze gradient descent - one of the most ubiquitous optimization algorithms. The algorithm is used to numerically solve differentiable and unconstrained optimization problems.

More formally, let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. We attempt to solve the problem:

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} f(\vec{x}). \tag{6.6}$$

The general idea behind the gradient descent algorithm is that it starts with some *initial guess* $\vec{x}_0 \in \mathbb{R}^n$ and produces a sequence of refined guesses $\vec{x}_1, \vec{x}_2, \ldots$, called *iterates*. In each iteration $t = 0, 1, 2, \ldots$, the algorithm updates its guess according to the following rule:

$$\vec{x}_{t+1} = \vec{x}_t + \eta\vec{v}_t \tag{6.7}$$

for some $\vec{v}_t \in \mathbb{R}^n$ and $\eta \geq 0$.

There are two quantities that we need to specify to complete the definition of the algorithm:

- The vector $\vec{v}_t$, or the *search direction*, which specifies a good direction to move.
- The scalar η, or the *step size*, which specifies how far we move in the direction of $\vec{v}_t$.

For the gradient descent algorithm, we assume that at every point $\vec{x} \in \mathbb{R}^n$ we can get two pieces of information about the function we are optimizing: the value of the function $f(\vec{x}) \in \mathbb{R}$ as well as its gradient $\nabla f(\vec{x}) \in \mathbb{R}^n$. Next, we will use this available information to come up with a good search direction $\vec{v}_t$. The choice of the step size η is a more difficult task. There is no universal choice of η that is good for all problems and a good choice of η is problem-specific. We will discuss the choice of the step size later in the section and show the important role it plays in the algorithm.

6.2.1 Search Direction

To choose $\vec{v}_t$, we want to ensure that $f(\vec{x}_t + \eta\vec{v}_t) \leq f(\vec{x}_t)$ for some small η. This motivates taking $\vec{v}_t$ as the *direction of steepest descent*, i.e., the direction in which f decays the fastest. The degree (i.e., the steepness) of the rate of change can be characterized by the directional derivative $D_{\vec{v}}f(\vec{x}) = \vec{v}^\top[\nabla f(\vec{x})]$.

Theorem 123 (Negative Gradient is Direction of Steepest Descent)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, and let $\vec{x} \in \mathbb{R}^n$. Then

$$-\frac{\nabla f(\vec{x})}{\|\nabla f(\vec{x})\|_2} \in \operatorname*{argmin}_{\substack{\vec{v}\in\mathbb{R}^n \\ \|\vec{v}\|_2=1}} D_{\vec{v}}f(\vec{x}). \tag{6.8}$$

Proof. Left as exercise. □

We also want to use the norm of the gradient in our update. For example, if $f(x) = x^2$ then $f'(x) = 2x$, which has large norm when x is far from the optimal point $x^\star = 0$. In this way, if the gradient is large then we are usually far away from an optimum, and we want our update $\eta\vec{v}_t$ to be large. This motivates choosing $\vec{v}_t = -\nabla f(\vec{x}_t)$.

6.2.2 Defining Gradient Descent

This choice of $\vec{v}_t$ gives the famous gradient descent iteration scheme:

$$\vec{x}_{t+1} = \vec{x}_t - \eta\nabla f(\vec{x}_t), \qquad \forall t = 0, 1, 2, \dots. \tag{6.9}$$

We can formalize this in an algorithm which terminates after T iterations for a user-set T.

Algorithm 3 Gradient Descent.

```
1: function GradientDescent($f, \vec{x}_0, \eta, T$)
2:     for $t = 0, 1, \dots, T-1$ do
3:         $\vec{x}_{t+1} \leftarrow \vec{x}_t - \eta\nabla f(\vec{x}_t)$
4:     end for
5:     return $\vec{x}_T$
6: end function
```

It is important to note that with the choice $\vec{v}_t = -\nabla f(\vec{x}_t)$, *descent is not guaranteed*. Namely, for a fixed $\eta > 0$, it is *not* true in general that

$$f(\vec{x}_{t+1}) = f(\vec{x}_t - \eta\nabla f(\vec{x}_t)) \leq f(\vec{x}_t). \tag{6.10}$$

Rather, from the proof of the above theorem, we only have that, for any given t there *exists* $\eta_t > 0$ such that

$$f(\vec{x}_t - \eta_t\nabla f(\vec{x}_t)) \leq f(\vec{x}_t). \tag{6.11}$$

This η_t, in general, is very small, and depends heavily on t and the local geometry of f around $\vec{x}_t$. None of these details are known by any realistic implementation of the gradient descent algorithm. In what follows, we will study gradient descent with a constant step size; this setting is most common in practice.

6.2.3 Convergence Analysis of Gradient Descent

When applying GD algorithm to a problem we need to make sure that we're making progress and that we eventually converge to the optimal solution. This question is very related to the choice of the step size. For the reminder of the section we will show that, for certain classes of functions $f: \mathbb{R}^n \to \mathbb{R}$ and certain chosen step sizes $\eta > 0$, the gradient descent algorithm is guaranteed to make progress in every iteration and converge to the optimal solution. We will first explore these questions through a familiar example.

Example 124 (Gradient Descent for Least Squares)**.** In this example we explore the convergence properties of the gradient descent algorithm by applying it to the least squares problem. Let $A \in \mathbb{R}^{m\times n}$ have full column rank, and $\vec{y} \in \mathbb{R}^m$.

$$\min_{\vec{x}\in\mathbb{R}^n} \|A\vec{x} - \vec{y}\|_2^2 . \tag{6.12}$$

Recall that, in this setting, the least squares problem has the unique closed form solution

$$\vec{x}^\star = (A^\top A)^{-1} A^\top \vec{y}. \tag{6.13}$$

We will use our knowledge of the true solution to analyze the convergence of the gradient descent algorithm. To apply gradient descent to this problem, let us first compute the gradient, which we see as

$$\nabla f(\vec{x}) = 2A^\top (A\vec{x} - \vec{b}). \tag{6.14}$$

Now let $\vec{x}_0 \in \mathbb{R}^n$ be the initial guess. For t a non-negative integer, we can write the gradient descent step:

$$\begin{aligned} \vec{x}_{t+1} &= \vec{x}_t - \eta \nabla f(\vec{x}_t) && (6.15)\\ &= \vec{x}_t - 2\eta A^\top (A\vec{x}_t - \vec{y}) && (6.16)\\ &= (I - 2\eta A^\top A)\vec{x}_t + 2\eta A^\top \vec{y}. && (6.17) \end{aligned}$$

We aim to set η which achieves the following two desired properties for the gradient descent iterates $\vec{x}_t$:

- We make progress, i.e., in every iteration we get closer to the optimal solution:

$$\|\vec{x}_{t+1} - \vec{x}^\star\|_2 \le \|\vec{x}_t - \vec{x}^\star\|_2 . \tag{6.18}$$

- We eventually converge to the optimal solution:

$$\lim_{t\to\infty} \vec{x}_t = \vec{x}^\star \iff \lim_{t\to\infty} \|\vec{x}_t - \vec{x}^\star\|_2 = 0. \tag{6.19}$$

To study this, let us write write out the relationship between $\vec{x}_{t+1} - \vec{x}^\star$ and $\vec{x}_t - \vec{x}^\star$. We do this by subtracting $\vec{x}^\star$ from both sides of Equation (6.17) and do some algebraic manipulations.

$$\begin{aligned} \vec{x}_{t+1} - \vec{x}^\star &= (I - 2\eta A^\top A)\vec{x}_t + 2\eta A^\top \vec{y} - \vec{x}^\star && (6.20)\\ &= (I - 2\eta A^\top A)\vec{x}_t + 2\eta A^\top \vec{y} - (A^\top A)^{-1} A^\top \vec{y} && (6.21)\\ &= (I - 2\eta A^\top A)\vec{x}_t + 2\eta (A^\top A)(A^\top A)^{-1} A^\top \vec{y} - (A^\top A)^{-1} A^\top \vec{y} && (6.22)\\ &= (I - 2\eta A^\top A)\vec{x}_t + 2\eta A^\top A\vec{x}^\star - \vec{x}^\star && (6.23)\\ &= (I - 2\eta A^\top A)\vec{x}_t - (I - 2\eta A^\top A)\vec{x}^\star && (6.24)\\ &= (I - 2\eta A^\top A)(\vec{x}_t - \vec{x}^\star). && (6.25) \end{aligned}$$

Here we used the trick of introducing $I = (A^\top A)(A^\top A)^{-1}$; this is because we wanted to group terms with $A^\top A$, and we also wanted to introduce instances of $\vec{x}^\star = (A^\top A)^{-1} A^\top \vec{y}$. Thus, while this was a trick, it was a motivated trick. In the end we get the following relationship:

$$\vec{x}_{t+1} - \vec{x}^\star = (I - 2\eta A^\top A)(\vec{x}_t - \vec{x}^\star). \tag{6.26}$$

If we take the norm of both sides, we have

$$\begin{aligned} \|\vec{x}_{t+1} - \vec{x}^\star\|_2 &= \left\|(I - 2\eta A^\top A)(\vec{x}_t - \vec{x}^\star)\right\|_2 && (6.27) \\ &\leq \left\|I - 2\eta A^\top A\right\|_2 \|\vec{x}_t - \vec{x}^\star\|_2 && (6.28) \\ &= \sigma_{\max}\{I - 2\eta A^\top A\} \|\vec{x}_t - \vec{x}^\star\|_2 . && (6.29) \end{aligned}$$

Thus, if $\sigma_{\max}\{I - 2\eta A^\top A\} < 1$, then we have

$$\|\vec{x}_{t+1} - \vec{x}^\star\|_2 \leq \sigma_{\max}\{I - 2\eta A^\top A\} \|\vec{x}_t - \vec{x}^\star\|_2 < \|\vec{x}_t - \vec{x}^\star\|_2 . \tag{6.30}$$

Thus we are guaranteed to make progress at each step. For the convergence guarantee, we recursively apply Equation (6.29) to get

$$\begin{aligned} \|\vec{x}_{t+1} - \vec{x}^\star\|_2 &\leq \sigma_{\max}\{I - 2\eta A^\top A\} \|\vec{x}_t - \vec{x}^\star\|_2 && (6.31) \\ &\leq \sigma_{\max}\{I - 2\eta A^\top A\}^2 \|\vec{x}_{t-1} - \vec{x}^\star\|_2 && (6.32) \\ &\leq \cdots && (6.33) \\ &\leq \sigma_{\max}\{I - 2\eta A^\top A\}^{t+1} \|\vec{x}_0 - \vec{x}^\star\|_2 . && (6.34) \end{aligned}$$

Thus taking the limit we get

$$\begin{aligned} \lim_{t\to\infty} \|\vec{x}_t - \vec{x}^\star\|_2 &\leq \lim_{t\to\infty} \sigma_{\max}\{I - 2\eta A^\top A\}^t \|\vec{x}_0 - \vec{x}^\star\|_2 && (6.35) \\ &= \|\vec{x}_0 - \vec{x}^\star\|_2 \cdot \underbrace{\lim_{t\to\infty} \sigma_{\max}\{I - 2\eta A^\top A\}^t}_{=0} && (6.36) \\ &= 0. && (6.37) \end{aligned}$$

Thus our convergence guarantee holds as well, so long as $\sigma_1\{I - 2\eta A^\top A\} < 1$.

Let us now generalize this analysis to a larger class of functions that includes least squares, as well as more general functions. We will focus our attention on the class of L-smooth and μ-strongly convex functions. Similar to what we did for least squares, we will use the optimal solution $\vec{x}^\star$ in our analysis. For a general function f, the quantity $\vec{x}^\star$ is almost always not known. But in the case of L-smooth and μ-strongly convex functions, a unique global minimizer exists. We just need the fact that it exists to show that for some small enough choice of the step size η, the gradient descent algorithm converges to this optimal solution. Before we formally state and prove this, we want to introduce one property of L-smooth functions that will become useful in our following proof.

Lemma 125 (Gradient Bound for L-Smooth Functions)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-smooth function. For all $\vec{x} \in \mathbb{R}^n$, we have

$$\|\nabla f(\vec{x})\|_2^2 \leq 2L \left(f(\vec{x}) - \min_{\vec{x}' \in \mathbb{R}^n} f(\vec{x}') \right). \tag{6.38}$$

This lemma says that the magnitude of the gradient gets smaller as we get closer to the optimal solution.

Proof. Recall the definition of L-smooth functions:

$$f(\vec{y}) \leq f(\vec{x}) + [\nabla f(\vec{x})]^\top (\vec{y} - \vec{x}) + \frac{L}{2} \|\vec{y} - \vec{x}\|_2^2 . \tag{6.39}$$

This is true for all points $\vec{x}, \vec{y} \in \mathbb{R}^n$, so let us fix $\vec{x}$ and set $\vec{y} = \vec{x} - \frac{\nabla f(\vec{x})}{L}$. We get

$$\begin{aligned} f\left(\vec{x} - \frac{\nabla f(\vec{x})}{L}\right) &\leq f(\vec{x}) + [\nabla f(\vec{x})]^\top \left(-\frac{\nabla f(\vec{x})}{L}\right) + \frac{L}{2} \left\|-\frac{\nabla f(\vec{x})}{L}\right\|_2^2 && (6.40) \\ &= f(\vec{x}) - \frac{1}{L} \|\nabla f(\vec{x})\|_2^2 + \frac{1}{2L} \|\nabla f(\vec{x})\|_2^2 && (6.41) \\ &= f(\vec{x}) - \frac{1}{2L} \|\nabla f(\vec{x})\|_2^2 . && (6.42) \end{aligned}$$

Now we can use the fact that $\min_{\vec{x}'} f(\vec{x}') \leq f(\vec{z})$ for all $\vec{z} \in \mathbb{R}^n$ to get

$$\begin{aligned} \min_{\vec{x}' \in \mathbb{R}^n} f(\vec{x}') &\leq f\left(\vec{x} - \frac{\nabla f(\vec{x})}{L}\right) && (6.43) \\ &\leq f(\vec{x}) - \frac{1}{2L} \|\nabla f(\vec{x})\|_2^2 . && (6.44) \end{aligned}$$

Rearranging, we have

$$\|\nabla f(\vec{x})\|_2^2 \leq 2L \left(f(\vec{x}) - \min_{\vec{x}' \in \mathbb{R}^n} f(\vec{x}')\right) \tag{6.45}$$

as desired. □

Now we have all the needed tools to prove the following property of gradient descent: for any μ-strongly convex and L-smooth function, the gradient descent algorithm converges exponentially fast to the true solution $\vec{x}^\star$.

Theorem 126 (Convergence of Gradient Descent for Smooth Strongly Convex Functions)
Let $\mu, L > 0$. Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be an L-smooth, μ-strongly convex function. Consider the following optimization problem:

$$p^\star = \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \tag{6.46}$$

which has optimal solution $\vec{x}^\star$. Then the constant step size $\eta = \frac{1}{L}$ is such that, if applying the gradient descent algorithm generates the sequence of points

$$\vec{x}_{t+1} = \vec{x}_t - \eta \nabla f(\vec{x}_t), \tag{6.47}$$

then for all initializations $\vec{x}_0$ and non-negative integers t,

$$\|\vec{x}_{t+1} - \vec{x}^\star\|_2^2 \leq \left(1 - \frac{\mu}{L}\right) \|\vec{x}_t - \vec{x}^\star\|_2^2 . \tag{6.48}$$

Proof. We want to upper bound $\|\vec{x}_{t+1} - \vec{x}^\star\|_2^2$, so let us write it out as

$$\begin{aligned} \|\vec{x}_{t+1} - \vec{x}^\star\|_2^2 &= \|\vec{x}_t - \eta \nabla f(\vec{x}_t) - \vec{x}^\star\|_2^2 && (6.49) \\ &= \|[\vec{x}_t - \vec{x}^\star] - \eta \nabla f(\vec{x}_t)\|_2^2 && (6.50) \\ &= \|\vec{x}_t - \vec{x}^\star\|_2^2 - 2\eta [\nabla f(\vec{x}_t)]^\top (\vec{x}_t - \vec{x}^\star) + \eta^2 \|\nabla f(\vec{x}_t)\|_2^2 && (6.51) \end{aligned}$$

$$= \|\vec{x}_t - \vec{x}^\star\|_2^2 + 2\eta[\nabla f(\vec{x}_t)]^\top(\vec{x}^\star - \vec{x}_t) + \eta^2 \|\nabla f(\vec{x}_t)\|_2^2 . \tag{6.52}$$

Because f is L-smooth, the lemma gives

$$\|\nabla f(\vec{x}_t)\|_2^2 \leq 2L(f(\vec{x}_t) - f(\vec{x}^\star)). \tag{6.53}$$

Because f is μ-strongly convex, we can write

$$f(\vec{x}^\star) \geq f(\vec{x}_t) + [\nabla f(\vec{x}_t)]^\top(\vec{x}^\star - \vec{x}_t) + \frac{\mu}{2}\|\vec{x}^\star - \vec{x}_t\|_2^2 \tag{6.54}$$

$$\implies [\nabla f(\vec{x}_t)]^\top(\vec{x}_t - \vec{x}^\star) \geq f(\vec{x}_t) - f(\vec{x}^\star) + \frac{\mu}{2}\|\vec{x}_t - \vec{x}^\star\|_2^2 \tag{6.55}$$

$$\implies [\nabla f(\vec{x}_t)]^\top(\vec{x}^\star - \vec{x}_t) \leq f(\vec{x}^\star) - f(\vec{x}_t) - \frac{\mu}{2}\|\vec{x}_t - \vec{x}^\star\|_2^2 . \tag{6.56}$$

Plugging these two estimates in, we get

$$\|\vec{x}_{t+1} - \vec{x}^\star\|_2^2 = \|\vec{x}_t - \vec{x}^\star\|_2^2 + 2\eta[\nabla f(\vec{x}_t)]^\top(\vec{x}^\star - \vec{x}_t) + \eta^2\|\nabla f(\vec{x}_t)\|_2^2 \tag{6.57}$$

$$\leq \|\vec{x}_t - \vec{x}^\star\|_2^2 + 2\eta\left[f(\vec{x}^\star) - f(\vec{x}_t) - \frac{\mu}{2}\|\vec{x}_t - \vec{x}^\star\|_2^2\right] + \eta^2\left[2L(f(\vec{x}_t) - f(\vec{x}^\star))\right] \tag{6.58}$$

$$= (1 - \eta\mu)\|\vec{x}_t - \vec{x}^\star\|_2^2 + 2\eta(\eta L - 1)(f(\vec{x}_t) - f(\vec{x}^\star)). \tag{6.59}$$

So as to make the second term 0, we choose $\eta = \frac{1}{L}$. This gives

$$\|\vec{x}_{t+1} - \vec{x}^\star\|_2^2 \leq \left(1 - \frac{\mu}{L}\right)\|\vec{x}_t - \vec{x}^\star\|_2^2 . \tag{6.60}$$

□

Corollary 127. *Consider the setting of Theorem 126. We have that* $0 \leq 1 - \frac{\mu}{L} < 1$*, and consequently:*

(i) (Descent at every step.) $\|\vec{x}_{t+1} - \vec{x}^\star\|_2 \leq \|\vec{x}_t - \vec{x}^\star\|_2$ *for all non-negative integers* t *and initializations* $\vec{x}_0$*.*

(ii) (Convergence.) $\lim_{t\to\infty} \vec{x}_t = \vec{x}^\star$*.*

Proof. We need to show that $1 - \frac{\mu}{L} \in [0, 1)$, or in other words that $0 < \frac{\mu}{L} \leq 1$. By assumption $\mu > 0$ so $\frac{\mu}{L} > 0$. The upper bound is given by the fact that f is μ-strongly convex and L-smooth, so

$$f(\vec{x}) + [\nabla f(\vec{x})]^\top(\vec{y} - \vec{x}) + \frac{\mu}{2}\|\vec{y} - \vec{x}\|_2^2 \leq f(\vec{y}) \leq f(\vec{x}) + [\nabla f(\vec{x})]^\top(\vec{y} - \vec{x}) + \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2 \tag{6.61}$$

which implies that

$$f(\vec{x}) + [\nabla f(\vec{x})]^\top(\vec{y} - \vec{x}) + \frac{\mu}{2}\|\vec{y} - \vec{x}\|_2^2 \leq f(\vec{x}) + [\nabla f(\vec{x})]^\top(\vec{y} - \vec{x}) + \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2 \tag{6.62}$$

which implies that

$$\frac{\mu}{2}\|\vec{y} - \vec{x}\|_2^2 \leq \frac{L}{2}\|\vec{y} - \vec{x}\|_2^2 \tag{6.63}$$

which finally implies that $\mu \leq L$, i.e., $0 < \frac{\mu}{L} \leq 1$. This immediately yields the first claim since $1 - \frac{\mu}{L} \in [0, 1)$ so $\sqrt{1 - \frac{\mu}{L}} \in [0, 1)$, therefore

$$\|\vec{x}_{t+1} - \vec{x}^\star\|_2 \leq \sqrt{1 - \frac{\mu}{L}}\|\vec{x}_t - \vec{x}^\star\|_2 < \|\vec{x}_t - \vec{x}^\star\|_2 . \tag{6.64}$$

The second claim follows since

$$\|\vec{x}_t - \vec{x}^\star\|_2^2 \leq \left(1 - \frac{\mu}{L}\right)\|\vec{x}_{t-1} - \vec{x}^\star\|_2^2 \tag{6.65}$$

$$\le \left(1-\frac{\mu}{L}\right)^2 \|\vec{x}_{t-2}-\vec{x}^\star\|_2^2 \tag{6.66}$$

$$\le \cdots \tag{6.67}$$

$$\le \left(1-\frac{\mu}{L}\right)^t \|\vec{x}_0-\vec{x}^\star\|_2^2 \tag{6.68}$$

and so

$$\lim_{t\to\infty} \|\vec{x}_t-\vec{x}^\star\|_2^2 = \lim_{t\to\infty}\left(1-\frac{\mu}{L}\right)^t \|\vec{x}_0-\vec{x}^\star\|_2^2 \tag{6.69}$$

$$= \|\vec{x}_0-\vec{x}^\star\|_2^2 \cdot \underbrace{\lim_{t\to\infty}\left(1-\frac{\mu}{L}\right)^t}_{=0} \tag{6.70}$$

$$= 0. \tag{6.71}$$

□

In this case the quantity $c=\sqrt{1-\frac{\mu}{L}}$ is important. If we wanted to run gradient descent for T iterations to within accuracy ϵ, we would have

$$\|\vec{x}_T-\vec{x}^\star\|_2 \le c^T \|\vec{x}_0-\vec{x}^\star\|_2 . \tag{6.72}$$

If we set $D \doteq \|\vec{x}_0-\vec{x}^\star\|$, then we can ensure that $\|\vec{x}_T-\vec{x}^\star\|_2 \le \epsilon$ by bounding the right-hand side $c^T\|\vec{x}_0-\vec{x}^\star\|_2 = c^T D$ by ϵ, getting

$$c^T D \le \epsilon \tag{6.73}$$

$$\implies c^T \le \frac{\epsilon}{D} \tag{6.74}$$

$$\implies T\log(c) \le \log(\epsilon)-\log(D) \tag{6.75}$$

$$\implies T \ge \frac{\log(\epsilon)-\log(D)}{\log(c)} \tag{6.76}$$

$$= \frac{\log(1/\epsilon)+\log(D)}{\log(1/c)}. \tag{6.77}$$

Thus, the lower the value of c is, the faster we get convergence towards a given accuracy.

Finally, it is important to point out that this is not the only class of functions for which the gradient descent algorithm converges. For example for the class of functions that are L-smooth and convex (i.e., not μ-strictly convex), the gradient descent algorithm does converge; in particular it converges to within accuracy ϵ after $T \ge O(1/\epsilon)$ iterations. This is vastly slower than the convergence achieved for μ-strongly convex functions.

6.3 Variations: Stochastic Gradient Descent

In this section we will cover the stochastic gradient descent (SGD) algorithm. This variant is motivated by optimization problems in which computing the gradient can be computationally expensive. The idea behind SGD is to use random sampling to find an estimate of the gradient that is easy to compute. Let us consider the class of unconstrained differentiable optimization problems that take the form

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} f(\vec{x}) = \min_{\vec{x}\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} f_i(\vec{x}). \tag{6.78}$$

The gradient of the objective function is given by

$$\nabla f(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(\vec{x}), \tag{6.79}$$

and the gradient descent step is given by

$$\vec{x}_{t+1} = \vec{x}_t - \eta \cdot \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(\vec{x}), \tag{6.80}$$

This class of functions is very common in applications that involve learning from data, one example being the least squares problem. Recall that we can write the least squares problem as

$$\min_{\vec{x}\in\mathbb{R}^n} \underbrace{\frac{1}{m}\|A\vec{x} - \vec{y}\|}_{=f(\vec{x})} = \min_{\vec{x}\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^{m} \underbrace{(\vec{a}_i^\top \vec{x} - b_i)^2}_{=f_i(\vec{x})} \tag{6.81}$$

Note that we multiplied the objective function with the constant $\frac{1}{m}$, which does not change the solutions to the optimization problem. We will see shortly that this constant is important in the SGD setting.

If the number of functions m is large and if each gradient $\nabla f_i(\vec{x})$ is complex to evaluate, computing the full gradient in Equation (6.79) can become prohibitively expensive. To overcome this, SGD proposes to take a random sample from the set of functions $\{f_1, f_2, \ldots, f_m\}$, evaluate the gradient at that sample only, and take the step

$$\vec{x}_{t+1} = \vec{x}_t - \eta\nabla f_i(\vec{x}). \tag{6.82}$$

If this index i is drawn uniformly at random, we can see that the expected value of the estimated gradient is equal to the full gradient.

$$\mathbb{E}[\nabla f_i(\vec{x})] = \sum_{i=1}^{m} \frac{1}{m}\nabla f_i(\vec{x}) \tag{6.83}$$

$$= \nabla f(\vec{x}). \tag{6.84}$$

Note that the direction $-\nabla f_i(\vec{x})$ is not guaranteed to be the direction of steepest descent of the function $f(\vec{x})$. In fact, it might not be a descent direction at all, and the value of the function f might even *increase* by taking a step in this direction. Further, it is not guaranteed to satisfy the first order optimality conditions; that is, when $\nabla f(\vec{x}) = \vec{0}$, the gradient of a single function $\nabla f_i(\vec{x})$ is generally not zero. This already tells us that the notion of convergence we studied (which is called "last-iterate convergence") is practically impossible to achieve — and in particular, the type of convergence proof that we studied for gradient descent will not work here. Instead the type of convergence guarantees we can hope for in this setting is for the *averaged* point across all iterations will converge to the optimal solution, i.e.,

$$\lim_{T\to\infty} f\left(\frac{1}{T}\sum_{t=1}^{T} \vec{x}_t\right) = \min_{\vec{x}\in\mathbb{R}^n} f(\vec{x}). \tag{6.85}$$

Further, to prove convergence of SGD we need to use a variable step size η_t. Using a fixed step size (like we did for gradient descent) doesn't guarantee convergence. The intuition behind this is that the gradient directions $\nabla f_i(\vec{x})$ for different i might be competing with each other and want to pull the solution in different directions. This will cause the sequence $\vec{x}_t$ to bounce back and forth. Using a variable step size η_t such that $\lim_{t\to\infty} \eta_t = 0$ makes it possible to converge to a single point. It is not trivial to show that the point that the algorithm will converge to is the optimal solution, but this can be proven under some assumptions on the objective function. For formal proofs, a good reference is [5].

Example 128 (Finding the Centroid Using SGD)**.** Fix points $\vec{p}_1, \dots, \vec{p}_m \in \mathbb{R}^n$. Let us consider the problem of finding their *centroid*, which we formulate as

$$\min_{\vec{x} \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{1}{2} \|\vec{x} - \vec{p}_i\|_2^2}_{=f_i(\vec{x})}. \tag{6.86}$$

The optimal solution for this problem is just the average of the points

$$\vec{x}^\star = \frac{1}{m} \sum_{i=1}^{m} \vec{p}_i. \tag{6.87}$$

Let us apply SGD on this problem with the time varying step size $\eta_t = \frac{1}{t}$ and initial guess $\vec{x}_0 = \vec{0}$. We first compute the gradients

$$\nabla f_i(\vec{x}) = \vec{x} - \vec{p}_i. \tag{6.88}$$

To simplify the notation we will apply SGD by selecting the indices in order.

$$\text{Iteration } 1: \quad \vec{x}_1 = \vec{x}_0 - \frac{1}{1}(\vec{x}_0 - \vec{p}_1) = \vec{p}_1 \tag{6.89}$$

$$\text{Iteration } 2: \quad \vec{x}_2 = \vec{x}_1 - \frac{1}{2}(\vec{x}_1 - \vec{p}_2) = \frac{\vec{p}_1 + \vec{p}_2}{2} \tag{6.90}$$

$$\text{Iteration } 3: \quad \vec{x}_3 = \vec{x}_3 - \frac{1}{3}(\vec{x}_2 - \vec{p}_3) = \frac{\vec{p}_1 + \vec{p}_2 + \vec{p}_3}{3} \tag{6.91}$$

$$\vdots \tag{6.92}$$

$$\text{Iteration } m: \quad \vec{x}_m = \vec{x}_{m-1} - \frac{1}{m}(\vec{x}_{m-1} - \vec{p}_m) = \frac{\sum_{i=1}^{m} \vec{p}_i}{m}. \tag{6.93}$$

So the SGD algorithm which takes a step along the gradient of a single $\nabla f_i(\vec{x})$ in every iteration converges to the true solution in m iterations. It is important to note that this is a toy example that is used to illustrate the application of SGD. As we discussed earlier, this type of convergence behavior is not common or guaranteed when applying SGD to more complicated real world problems.

Finally, we note that this set up of the SGD algorithm allows us to think of the optimization scheme as an online algorithm. Instead of randomly selecting an index to compute the gradient, assume that the data points (i.e., in the above example the $\vec{p}_i$) that define the functions $f_i(\vec{x})$ arrive one at a time. Following the same idea of SGD, we can perform an optimization step as each new data point becomes available. In this setting it is important to normalize by the number of data points m in the objective function; otherwise the objective function will keep growing as we get more data, irregardless of whether our optimization algorithm finds a good solution.

6.4 Variations: Gradient Descent for Constrained Optimization

So far we have seen how gradient descent can be applied to different problems all of which are unconstrained. There are variants of the gradient descent algorithm that can be used to solve problems with *simple* constraints. Let $\Omega \subseteq \mathbb{R}^n$ be a *convex* set, let $f\colon \Omega \to \mathbb{R}$ be a differentiable function, and consider the problem

$$p^\star = \min_{\vec{x} \in \Omega} f(\vec{x}). \tag{6.94}$$

For these types of problems we want to limit our search to points inside the set Ω. If we start with an initial guess $\vec{x}_0 \in \Omega$ and apply the (unconstrained) gradient descent algorithm

$$\vec{x}_{t+1} = \vec{x}_t - \eta \nabla f(\vec{x}), \tag{6.95}$$

the point $\vec{x}_{t+1}$ might end up outside of the feasible set Ω, even when $\vec{x}_t$ is in Ω. In this section we will consider two variants of the gradient descent algorithm that propose techniques to deal with constraints.

6.4.1 Projected Gradient Descent

This first variant is projected gradient descent, proposes the idea of *projecting* the point back into the feasible set.

Definition 129 (Projection onto a Convex Set)
Let Ω be a compact and convex set. Let $\vec{y} \in \mathbb{R}^n$ be any vector. We define the projection of the vector $\vec{y}$ onto the set Ω as

$$\operatorname{proj}_{\Omega}(\vec{y}) = \operatorname*{argmin}_{\vec{x} \in \Omega} \|\vec{x} - \vec{y}\|_2^2 . \tag{6.96}$$

Because of the compactness assumption, the projection is unique. In words, $\operatorname{proj}_{\Omega}(\vec{y})$ is the closest point in Ω to $\vec{y}$. From this definition one sees that if $\vec{y} \in \Omega$ then $\operatorname{proj}_{\Omega}(\vec{y}) = \vec{y}$. Using this definition we can write the step of the projected gradient descent algorithm as

$$\vec{x}_{t+1} = \operatorname{proj}_{\Omega}(\vec{x}_t - \eta \nabla f(\vec{x}_t)). \tag{6.97}$$

Algorithm 4 Projected Gradient Descent

```
1: function ProjectedGradientDescent($f, \vec{x}_0, \Omega, \eta, T$)
2:     for $t = 0, 1, \ldots, T-1$ do
3:         $\vec{x}_{t+1} \leftarrow \operatorname{proj}_{\Omega}(\vec{x}_t - \eta \nabla f(\vec{x}_t))$
4:     end for
5:     return $\vec{x}_T$
6: end function
```

Note that the projection operator itself solves a convex optimization problem. It is only meaningful to consider the projected gradient descent algorithm if this projection problem is simple enough to be solved in every iteration. We have seen examples of projections onto simple sets. For example, the least squares problem computes the projection of a vector onto the subspace $\mathcal{R}(A)$, and we know how to efficiently solve this projection problem. However, that's not always the case, and the projection problem might be nearly as difficult to solve as our original optimization problem.

6.4.2 Conditional Gradient Descent

We introduce another variant of the gradient descent algorithm for constrained problems called conditional gradient descent (also known as Frank-Wolfe method). Recall that in our development of the choice of the "best" search direction for the (regular) gradient descent algorithm, we considered the quantity

$$D_{\vec{v}_t} f(\vec{x}_t) = \lim_{\eta \to 0} \frac{f(\vec{x}_t + \eta \vec{v}_t) - f(\vec{x}_t)}{\eta} = [\nabla f(\vec{x}_t)]^\top \vec{v}_t. \tag{6.98}$$

This quantity, known as the directional derivative, is the (instantaneous) rate of change of the function f at point $\vec{x}_t$ along direction $\vec{v}_t$. We also saw that the choice $\vec{v}_t = -\nabla f(\vec{x}_t)$ is the direction that minimizes this rate of change. This is a reasonable choice for the unconstrained case since we can freely move in any direction. The idea behind the conditional gradient descent algorithm is to limit our search direction to the feasible set Ω and find a vector inside the set that minimizes $\nabla \left[f(\vec{x}_t)\right]^\top \vec{v}_t$. Thus, given $\vec{x}_t \in \Omega$, we can define the search direction of the conditional gradient descent as

$$\vec{v}_t = \operatorname*{argmin}_{\vec{v} \in \Omega} [\nabla f(\vec{x}_t)]^\top \vec{v}. \tag{6.99}$$

Once we find this direction $\vec{v}_t$, we need to use it in a way that ensures that we don't leave the set. Taking a step along this direction $\vec{x}_{t+1} = \vec{x}_t + \eta \vec{v}_t$ doesn't guarantee this. But we can do something different and take the convex combination between the current point and the search direction. That is, for $\delta_t \in [0, 1]$ we can update

$$\vec{x}_{t+1} = (1 - \delta_t)\vec{x}_t + \delta_t \vec{v}_t \tag{6.100}$$

By convexity of the set, this guarantees that if we start with a feasible initial guess $\vec{x}_0 \in \Omega$, we get a sequence of points $\vec{x}_t \in \Omega$ for all $t \geq 0$. While this property holds for any choice of $\delta_t \in [0, 1]$, to prove convergence of the algorithm we require $\lim_{t\to\infty} \delta_t = 0$; a conventional choice is $\delta_t = \frac{1}{t}$.

Algorithm 5 Conditional Gradient Descent (Frank-Wolfe)

```
1: function ConditionalGradientDescent($f, \vec{x}_0, \Omega, \delta, T$)
2:     for $t = 0, 1, \ldots, T - 1$ do
3:         $\vec{v}_t \leftarrow \operatorname{argmin}_{\vec{v} \in \Omega} [\nabla f(\vec{x}_t)]^\top \vec{v}$
4:         $\vec{x}_{t+1} \leftarrow (1 - \delta_t)\vec{x}_t + \delta_t \vec{v}_t$
5:     end for
6:     return $\vec{x}_T$
7: end function
```

As a last note, we point out that the problem of finding a search direction $\vec{v}_t$ is a constrained optimization problem within Ω.

Chapter 7

Duality

Relevant sections of the textbooks:

- [1] Chapter 5.
- [2] Section 8.5.

7.1 Lagrangian

This chapter develops the theory of duality for optimization problems. This is a technique that can help us solve or bound the optimal values for constrained optimization problems by solving the related dual problem. The dual problem might not always give us a direct solution to our original ("primal") problem, but it will always give us a bound. For this, we first define the Lagrangian for a problem.

Let us start with our generic constrained optimization problem, which we will call problem $\mathcal{P}$ (which stands for *primal*):

$$\begin{aligned} \text{problem } \mathcal{P}: \qquad p^\star = \min_{\vec{x}\in\mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \leq 0, \qquad \forall i \in \{1,\dots,m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1,\dots,p\}. \end{aligned} \tag{7.1}$$

Let us denote its feasible set by

$$\Omega \doteq \left\{ \vec{x} \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} f_i(\vec{x}) \leq 0, & \forall i \in \{1,\dots,m\} \\ h_j(\vec{x}) = 0, & \forall j \in \{1,\dots,p\} \end{array} \right\} \qquad \text{so that} \qquad p^\star = \min_{\vec{x}\in\Omega} f_0(\vec{x}). \tag{7.2}$$

We know how to algorithmically approach unconstrained problems, say for instance using gradient descent, which we discussed in the previous chapter. Thus, our question is whether there is a way to incorporate the constraints of the problem $\mathcal{P}$ into the objective function itself. Trying to understand if this is possible is one motivation for the Lagrangian.

To this end, we construct the *indicator function* $\mathbb{1}[\cdot]$, which for a condition $C(\vec{x})$ defined as

$$\mathbb{1}[C(\vec{x})] \doteq \begin{cases} 0, & \text{if } C(\vec{x}) \text{ is true,} \\ +\infty, & \text{if } C(\vec{x}) \text{ is false.} \end{cases} \tag{7.3}$$

Then the problem $\mathcal{P}$ in (7.1) is equivalent to the following *unconstrained* problem:

$$p^\star = \min_{\vec{x}\in\Omega} f_0(\vec{x}) \tag{7.4}$$
$$= \min_{\vec{x}\in\mathbb{R}^n} \{f_0(\vec{x}) + \mathbb{1}[\vec{x}\in\Omega]\} \tag{7.5}$$
$$= \min_{\vec{x}\in\mathbb{R}^n} \left\{ f_0(\vec{x}) + \sum_{i=1}^{m} \mathbb{1}[f_i(\vec{x}) \leq 0] + \sum_{j=1}^{p} \mathbb{1}[h_j(\vec{x}) = 0] \right\}. \tag{7.6}$$

To explain the chain of equalities leading to (7.6), say we consider the function

$$F_0(\vec{x}) \doteq f_0(\vec{x}) + \sum_{i=1}^{m} \mathbb{1}[f_i(\vec{x}) \leq 0] + \sum_{j=1}^{p} \mathbb{1}[h_j(\vec{x}) = 0] \tag{7.7}$$

evaluated at an $\vec{x} \notin \Omega$, i.e., there is some i such that $f_i(\vec{x}) > 0$ or some j such that $h_j(\vec{x}) \neq 0$. Then $F_0(\vec{x}) = \infty$. Therefore no solution to the minimization in (7.6) will ever be outside of Ω, i.e., all solutions to (7.6) will be contained in Ω. And for $\vec{x} \in \Omega$, we have $F_0(\vec{x}) = f_0(\vec{x})$, so that $\min_{\vec{x}\in\Omega} f_0(\vec{x}) = \min_{\vec{x}\in\mathbb{R}^n} F_0(\vec{x})$ (and the argmins are equal too). Hence the problem in (7.6) is equivalent to the original problem $\mathcal{P}$.

Thus through this reformulation, we have removed the explicit constraints of the problem and incorporated them into the objective function of the problem, and we can nominally write the problem as an unconstrained problem. If we had some magic algorithm which allowed us to solve *all* unconstrained problems, then this reduction would let us solve constrained problems as well.

Unfortunately, we do *not* have such a magic algorithm. Our usual algorithms for solving unconstrained problems, such as gradient descent, require differentiable objective functions that are well-defined. Thus, it falls to us to express our indicator functions in a differentiable way.

For this we consider *approximating* the indicator functions by other functions that are differentiable.

The key idea here is that, for a given indicator $\mathbb{1}[f_i(\vec{x}) \leq 0]$, we can express it as

$$\mathbb{1}[f_i(\vec{x}) \leq 0] = \max_{\lambda_i\in\mathbb{R}_+} \lambda_i f_i(\vec{x}), \tag{7.8}$$

where $\mathbb{R}_+$ is the set of non-negative real numbers. Why does this equality hold? Let's break it up into cases. Suppose that $f_i(\vec{x}) > 0$. Then λ_i being more positive would make $\lambda_i f_i(\vec{x})$ more positive, so the maximization will send $\lambda_i \to \infty$, making $\lambda_i f_i(\vec{x}) = \infty$. Now suppose that $f_i(\vec{x}) \leq 0$. Then λ_i being more positive would make $\lambda_i f_i(\vec{x})$ more negative, so the maximization will keep λ_i as its lower bound 0, making $\lambda_i f_i(\vec{x}) = 0$. For a similar reason, for an indicator $\mathbb{1}[h_j(\vec{x}) = 0]$, we have

$$\mathbb{1}[h_j(\vec{x}) = 0] = \max_{\nu_j\in\mathbb{R}} \nu_j h_j(\vec{x}). \tag{7.9}$$

Thus, we can rewrite the problem $\mathcal{P}$ as

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} \left\{ f_0(\vec{x}) + \sum_{i=1}^{m} \max_{\lambda_i\in\mathbb{R}_+} \lambda_i f_i(\vec{x}) + \sum_{j=1}^{p} \max_{\nu_j\in\mathbb{R}} \nu_j h_j(\vec{x}) \right\} \tag{7.10}$$
$$= \min_{\vec{x}\in\mathbb{R}^n} \left\{ f_0(\vec{x}) + \max_{\vec{\lambda}\in\mathbb{R}^m_+} \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \max_{\vec{\nu}\in\mathbb{R}^p} \sum_{j=1}^{p} \nu_j h_j(\vec{x}) \right\} \tag{7.11}$$
$$= \min_{\vec{x}\in\mathbb{R}^n} \max_{\substack{\vec{\lambda}\in\mathbb{R}^m_+ \\ \vec{\nu}\in\mathbb{R}^p}} \left\{ f_0(\vec{x}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \sum_{j=1}^{p} \nu_j h_j(\vec{x}) \right\}. \tag{7.12}$$

This interior function is called the *Lagrangian*, and is much easier to work with than the original unconstrained problem with indicator functions. Thus we have made our constrained problem into a (mostly) unconstrained problem,[1] at the expense of adding an extra $\max$. We will see how to cope with this extra difficulty shortly.

More formally, we have the following definition:

Definition 130 (Lagrangian)

The *Lagrangian* of problem $\mathcal{P}$ in (7.1) is the function $L\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ given by

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) \doteq f_0(\vec{x}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \sum_{j=1}^{p} \nu_j h_j(\vec{x}). \tag{7.13}$$

Additionally, $\lambda_i, \nu_j \in \mathbb{R}$ are called *Lagrange multipliers*.

The important intuition behind the Lagrange multipliers is that they are *penalties* for violating their corresponding constraint. In particular, just like how the indicator function $\mathbb{1}[f_i(\vec{x}) \leq 0]$ assigns an infinite penalty for violating the constraint $f_i(\vec{x}) \leq 0$, the term $\lambda_i f_i(\vec{x})$ assigns a penalty $\lambda_i f_i(\vec{x})$ for violating the constraint $f_i(\vec{x}) \leq 0$. One can show that $\lambda_i f_i(\vec{x}) \leq \mathbb{1}[f_i(\vec{x}) \leq 0]$, so the Lagrangian penalty is a smooth lower bound to the indicator penalty, which is something like a hard-threshold. We can do similar analysis for the equality constraints $h_j(\vec{x}) = 0$.

As derived before, the primal problem can be expressed in terms of the Lagrangian as

$$p^\star = \min_{\vec{x} \in \mathbb{R}^n} \max_{\substack{\vec{\lambda} \in \mathbb{R}^m_+ \\ \vec{\nu} \in \mathbb{R}^p}} L(\vec{x}, \vec{\lambda}, \vec{\nu}). \tag{7.14}$$

We conclude with two very important properties of the Lagrangian.

Proposition 131

For every $\vec{x} \in \mathbb{R}^n$, the function $(\vec{\lambda}, \vec{\nu}) \mapsto L(\vec{x}, \vec{\lambda}, \vec{\nu})$ is an affine function, and hence a concave function. (We also say that L is affine (resp. concave) in $\vec{\lambda}$ and $\vec{\nu}$.)

Proof. The proof follows from the definition of the Lagrangian:

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = f_0(\vec{x}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \sum_{j=1}^{p} \nu_j h_j(\vec{x}) \tag{7.15}$$

is affine in $\vec{\lambda}$ and $\vec{\nu}$. □

Example 132. Consider the optimization problem

$$\begin{aligned} p^\star = \min_{x \in \mathbb{R}} \quad & 3x^2 \\ \text{s.t.} \quad & 2x^3 \leq 8. \end{aligned} \tag{7.16}$$

This problem is not convex, but we can still find its Lagrangian as

$$L(x, \lambda) = 3x^2 + \lambda(2x^3 - 8). \tag{7.17}$$

[1]Not quite — the $\max$ is actually over $\mathbb{R}^m_+ \times \mathbb{R}^p$ which is technically a constrained set. But this non-negativity constraint is usually much easier to handle than arbitrary constraints, in part because the constraint set is convex.

7.2 Weak Duality

Our foray into duality starts by considering the primal problem:

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} \max_{\substack{\vec{\lambda}\in\mathbb{R}^m_+ \\ \vec{\nu}\in\mathbb{R}^p}} L(\vec{x},\vec{\lambda},\vec{\nu}). \tag{7.18}$$

The *dual problem* is obtained by swapping the $\min$ and $\max$:

$$d^\star = \max_{\substack{\vec{\lambda}\in\mathbb{R}^m_+ \\ \vec{\nu}\in\mathbb{R}^p}} \min_{\vec{x}\in\mathbb{R}^n} L(\vec{x},\vec{\lambda},\vec{\nu}). \tag{7.19}$$

In general, the quantities $p^\star$ and $d^\star$ *do not have to be equal*! And swapping the $\min$ and the $\max$ is the generic definition of "duality".

Definition 133 (Dual Problem)
For a primal problem $\mathcal{P}$ as described in (7.1), its *dual problem* $\mathcal{D}$ is defined as

$$\text{problem } \mathcal{D}: \qquad d^\star = \max_{\substack{\lambda\in\mathbb{R}^m \\ \nu\in\mathbb{R}^p}} \quad g(\vec{\lambda},\vec{\nu}) \qquad \text{s.t.} \quad \lambda_i \geq 0, \qquad \forall i \in \{1,\ldots,m\}. \tag{7.20}$$

Here the function $g\colon \mathbb{R}^m_+ \times \mathbb{R}^p \to \mathbb{R}$ is the *dual function* and defined as

$$g(\vec{\lambda},\vec{\nu}) = \min_{\vec{x}\in\mathbb{R}^n} L(\vec{x},\vec{\lambda},\vec{\nu}). \tag{7.21}$$

Thus, $g(\vec{\lambda},\vec{\nu})$ can be computed as an unconstrained optimization problem over $\vec{x}$, in particular $g(\vec{\lambda},\vec{\nu})$ is the minimum value of $L(\vec{x},\vec{\lambda},\vec{\nu})$ over all $\vec{x}$.

There are several important properties of the dual problem and dual function, which we summarize below.

Proposition 134
The dual function g is a concave function of $(\vec{\lambda},\vec{\nu})$, regardless of any properties of $\mathcal{P}$.

Proof. We already showed that $L(\vec{x},\vec{\lambda},\vec{\nu})$ is an affine (and thus concave) function of $\vec{\lambda}$ and $\vec{\nu}$. Thus the function

$$g(\vec{\lambda},\vec{\nu}) = \min_{\vec{x}\in\mathbb{R}^n} L(\vec{\lambda},\vec{\nu}) \tag{7.22}$$

is a pointwise minimum of concave functions and is thus concave. □

Corollary 135. *The dual problem $\mathcal{D}$ is always a convex problem, no matter what the primal problem $\mathcal{P}$ is.*

Note that the minimization of a convex function or the maximization of a concave function are both considered "convex" problems.

This means we have analytic and algorithmic ways to solve the dual problem $\mathcal{D}$. If we can connect the solutions to the dual problem $\mathcal{D}$ to the primal problem $\mathcal{P}$, this means that we have reduced the process of solving $\mathcal{P}$ to the process of solving a convex optimization problem, and this is something we know how to do. In the rest of this chapter, we discuss when this reduction is possible.

Example 136. Let us compute the dual of the optimization problem

$$p^\star = \min_{x \in \mathbb{R}} \quad 5x^2 \tag{7.23}$$

$$\text{s.t.} \quad 3x \leq 5. \tag{7.24}$$

The Lagrangian is

$$L(x, \lambda) = 5x^2 + \lambda(3x - 5). \tag{7.25}$$

The dual function is

$$g(\lambda) = \min_x L(x, \lambda) = \min_x \left\{5x^2 + \lambda(3x - 5)\right\}. \tag{7.26}$$

For each $\lambda > 0$, the function $L(x, \lambda) = 5x^2 + \lambda(3x - 5)$ is bounded below (in x), convex, and differentiable, so to minimize it we set its derivative to 0. The optimal $x^\star$ is a function of the Lagrange multiplier λ, so we write it as $x^\star(\lambda)$. We have

$$0 = \nabla_x L(x^\star(\lambda), \lambda) \tag{7.27}$$

$$= 10x^\star(\lambda) + 3\lambda \tag{7.28}$$

$$\implies x^\star(\lambda) = -\frac{3}{10}\lambda. \tag{7.29}$$

Thus we can plug this optimal point back in to get g:

$$g(\lambda) = L(x^\star(\lambda), \lambda) \tag{7.30}$$

$$= 5(x^\star(\lambda))^2 + \lambda(3x^\star(\lambda) - 5) \tag{7.31}$$

$$= \frac{5 \cdot 9}{100}\lambda^2 + \lambda\left(-\frac{3 \cdot 3}{10}\lambda - 5\right) \tag{7.32}$$

$$= -\frac{9}{20}\lambda^2 - 5\lambda. \tag{7.33}$$

Thus the dual problem is

$$d^\star = \max_{\lambda \in \mathbb{R}_+}\left(-\frac{9}{20}\lambda^2 - 5\lambda\right). \tag{7.34}$$

One may solve this problem directly to obtain $\lambda^\star = 0$ and $d^\star = 0$.

We also have some more bounds for the Lagrangian in terms of f_0, g, $p^\star$, and $d^\star$.

Proposition 137

Let $\vec{x} \in \Omega$, let $\vec{\lambda} \in \mathbb{R}^m_+$, and let $\vec{\nu} \in \mathbb{R}^p$, so that $\vec{x}$ is feasible for $\mathcal{P}$ and $(\vec{\lambda}, \vec{\nu})$ is feasible for $\mathcal{D}$. Then we have:

(a) $f_0(\vec{x}) \geq p^\star$ and $g(\vec{\lambda}, \vec{\nu}) \leq d^\star$;

(b) $f_0(\vec{x}) \geq L(\vec{x}, \vec{\lambda}, \vec{\nu}) \geq g(\vec{\lambda}, \vec{\nu})$;

(c) $f_0(\vec{x}) \geq d^\star$ and $g(\vec{\lambda}, \vec{\nu}) \leq p^\star$.

Proof. The inequalities in (a) follow from the characterizations

$$p^\star = \min_{\vec{x}' \in \Omega} f_0(\vec{x}') \leq f_0(\vec{x}) \tag{7.35}$$

$$d^\star = \max_{\substack{\vec{\lambda}' \in \mathbb{R}^m_+ \\ \vec{\nu}' \in \mathbb{R}^p}} g(\vec{\lambda}', \vec{\nu}') \geq g(\vec{\lambda}, \vec{\nu}). \tag{7.36}$$

The inequalities in (b) follow from the characterizations

$$g(\vec{\lambda}, \vec{\nu}) = \min_{\vec{x}' \in \Omega} L(\vec{x}', \vec{\lambda}, \vec{\nu}) \leq L(\vec{x}, \vec{\lambda}, \vec{\nu}) \tag{7.37}$$

$$f_0(\vec{x}) = \max_{\substack{\vec{\lambda}' \in \mathbb{R}^m_+ \\ \vec{\nu}' \in \mathbb{R}^p}} L(\vec{x}, \vec{\lambda}', \vec{\nu}') \geq L(\vec{x}, \vec{\lambda}, \vec{\nu}). \tag{7.38}$$

The inequalities in (c) follow from (b), in the sense that

$$f_0(\vec{x}) \geq g(\vec{\lambda}, \vec{\nu}) \qquad \forall \vec{\lambda} \in \mathbb{R}^m_+ \qquad \forall \vec{\nu} \in \mathbb{R}^p \tag{7.39}$$

$$\implies f_0(\vec{x}) \geq \max_{\substack{\vec{\lambda}' \in \mathbb{R}^m_+ \\ \vec{\nu}' \in \mathbb{R}^p}} g(\vec{\lambda}', \vec{\nu}') = d^\star, \tag{7.40}$$

and additionally

$$f_0(\vec{x}) \geq g(\vec{\lambda}, \vec{\nu}) \qquad \forall \vec{x} \in \Omega \tag{7.41}$$

$$\implies p^\star = \min_{\vec{x}' \in \Omega} f_0(\vec{x}') \geq g(\vec{\lambda}, \vec{\nu}). \tag{7.42}$$

□

From all these inequalities, the relationship between $p^\star$ and $d^\star$ is totally unclear. Actually, their relationship is very simple; it is captured in a condition called "weak duality".

Definition 138 (Types of Duality)
Let $\mathcal{P}$ be a primal problem with optimum $p^\star$. Let $\mathcal{D}$ be the corresponding dual problem with optimum $d^\star$.

(a) If $p^\star \geq d^\star$ then we say that *weak duality* holds.

(b) If $p^\star = d^\star$ then we say that *strong duality* holds.

(c) The quantity $p^\star - d^\star$ is called the *duality gap*.

Proposition 139 (Weak Duality Always Holds)
For *any* problem, weak duality holds, i.e., the duality gap is non-negative.

The fact that weak duality always holds is actually a corollary of a slightly more generic result called the minimax inequality, which we now state and prove.

Proposition 140 (Minimax Inequality)
Let X and Y be *any* sets, and $F\colon X \times Y \to \mathbb{R}$ be any function. Then

$$\min_{x \in X} \max_{y \in Y} F(x, y) \geq \max_{y \in Y} \min_{x \in X} F(x, y). \tag{7.43}$$

Proof. For any $x \in X$ and $y \in Y$, we have

$$F(x, y) \geq \min_{x' \in X} F(x', y). \tag{7.44}$$

Taking the maximum over y on both sides, we have

$$\max_{y'\in Y} F(x,y') \geq \max_{y'\in Y}\min_{x'\in X} F(x',y'). \tag{7.45}$$

Finally, the left hand side is a function of x but the right hand side is a constant, so we might as well take the minimum in x on the left hand side to get

$$\min_{x'\in X}\max_{y'\in Y} F(x',y') \geq \max_{y'\in Y}\min_{x'\in X} F(x',y'). \tag{7.46}$$

$\square$

Here is a game theoretic interpretation of the minimax inequality.

Suppose X and Y are two players choosing actions x and y. Player X chooses action x trying to *minimize* the value of the function $F(x,y)$. Player Y chooses action y trying to *maximize* the value of the function $F(x,y)$. The two players are perfectly rational and take turns — that is, the players choose their actions one after the other, and the second player has full knowledge of the first player's action (which is locked in) as they choose their action. (In game theory this is called a two-player zero-sum sequential game.) The value of the game, given actions x and y, is $F(x,y)$.

The outer optimization corresponds to the player going first, and the inner minimization corresponds to the player going second. Why? Consider the optimization problem $\min_{x\in X}\max_{y\in Y} F(x,y)$. Then defining $G(x) = \max_{y\in Y} F(x,y)$, we see that given a particular x, $G(x)$ is the maximum value obtained by varying y of $F(x,y)$. That is, it corresponds to the best play by player Y given that player X picks action x. This means that the inner minimization corresponds to the second player Y's actions given that the first player X chooses action x. And so the outer optimization corresponds to the first player X's action. The situation is analogous on the right-hand side.

What the minimax inequality says is that *going second is better*. That is, being the second player and choosing your action with full knowledge of the first player's action leads to a better outcome for you than going first with no knowledge of the second player's action. When X is the second player, the value of the game is lower than when X is the first player; when Y is the second player, the value of the game is higher than when Y is the first player.

Given the minimax inequality, the proof that weak duality always holds is only one line.

Proof that weak duality always holds. We have

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n}\max_{\substack{\vec{\lambda}\in\mathbb{R}^m_+\\ \vec{\nu}\in\mathbb{R}^p}} L(\vec{x},\vec{\lambda},\vec{\nu}) \geq \max_{\substack{\vec{\lambda}\in\mathbb{R}^m_+\\ \vec{\nu}\in\mathbb{R}^p}}\min_{\vec{x}\in\mathbb{R}^n} L(\vec{x},\vec{\lambda},\vec{\nu}) = d^\star. \tag{7.47}$$

$\square$

Weak duality is useful because it gives a bound on how close to optimum a given decision variable is. More specifically, we have that for all $\vec{x}\in\Omega$ and all $(\vec{\lambda},\vec{\nu})\in\mathbb{R}^m_+\times\mathbb{R}^p$ that

$$f_0(\vec{x}) \geq p^\star \geq d^\star \geq g(\vec{\lambda},\vec{\nu}) \implies f_0(\vec{x}) - g(\vec{\lambda},\vec{\nu}) \geq f_0(\vec{x}) - p^\star. \tag{7.48}$$

Thus, if $f_0(\vec{x}) - g(\vec{\lambda},\vec{\nu}) \leq \epsilon$ then we know that $f_0(\vec{x}) - p^\star \leq \epsilon$. This is called a *certificate of optimality*. We can use this certificate as a stopping condition for algorithms such as gradient descent or a broad class of algorithms known as *primal-dual* algorithms.

7.3 Strong Duality

Since weak duality holds for all problems, it necessarily would not have too many critical implications. The stronger condition of *strong duality* turns out to hold the keys to efficiently computing minima for constrained optimization problems.

First, a natural question is: when does strong duality hold? It does not *always* hold, and in particular it is a stronger condition than weak duality. You will see some example problems where strong duality does not hold in discussion.

There are many conditions under which strong duality holds. In this course, we discuss one such condition which is relatively simple to evaluate and broadly applicable. This condition is called *Slater's condition*.

Theorem 141 (Slater's Condition)
Consider a *convex* problem $\mathcal{P}$:

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & f_0(\vec{x}) \\ \text{s.t.} \quad & f_i(\vec{x}) \le 0, \qquad \forall i \in \{1, \dots, m\} \\ & h_j(\vec{x}) = 0, \qquad \forall j \in \{1, \dots, p\}. \end{aligned} \tag{(7.1)}$$

If there exists any $\vec{x} \in \mathbb{R}^n$ such that all of the following hold:

- for all $i \in \{1, \dots, m\}$ such that f_i is an *affine* function, we have $f_i(\vec{x}) \le 0$;
- for all $i \in \{1, \dots, m\}$ such that f_i is *not* an affine function, we have $f_i(\vec{x}) < 0$;
- and for all $j \in \{1, \dots, p\}$, we have $h_j(\vec{x}) = 0$;

then strong duality holds for $\mathcal{P}$ and its dual $\mathcal{D}$, i.e., the duality gap is 0.

This result is sometimes called *refined Slater's condition* as it is a slight modification of another condition which is also called Slater's condition.

We will not prove this in class; a proof sketch is in [1]. It uses the separating hyperplane theorem we proved earlier. This result is actually one of the main payoffs of the separating hyperplane theorem that we will see in this course.

Again, observe that we *only* need a *single* point which fulfills the three conditions in order to declare that Slater's condition holds. Moreover, this point *need not be optimal* for the problem — any point at all which satisfies the conditions will do.

Slater's condition has a geometric interpretation: if there is a point in the *relative interior*[2] of the feasible set, then strong duality holds. In problems we are able to visualize, this makes it relatively easy to determine whether Slater's condition holds.

[2] See [6] for more details.

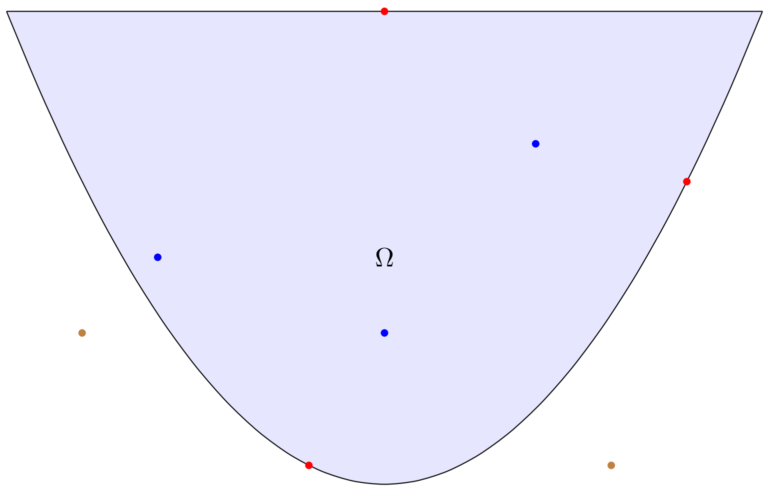


Figure 7.1: Points in the convex set $\Omega = \{(x, y) \in \mathbb{R}^2 \colon -5 \leq x \leq 5, \frac{25}{4} \geq y \geq \frac{1}{4}x^2\}$. Blue points are feasible points which fulfill the conditions of Slater's condition (thus ensuring that Slater's condition holds for the problem $\min_{\vec{x} \in \Omega} f_0(\vec{x})$ provided that f_0 is convex). Red points are feasible points which do not fulfill the conditions of Slater's condition. Brown points are infeasible points.

Example 142. In this example, we consider the equality-constrained minimum-norm problem

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & \|\vec{x}\|_2^2 && (7.49)\\ \text{s.t.} \quad & A\vec{x} = \vec{y}. \end{aligned}$$

This problem only has equality constraints, and so it only has the Lagrange multiplier $\vec{\nu}$. The Lagrangian of this problem is

$$L(\vec{x}, \vec{\nu}) = \|\vec{x}\|_2^2 + \sum_{j=1}^{p} \nu_j (A\vec{x} - \vec{y})_j \tag{7.50}$$

$$= \|\vec{x}\|_2^2 + \vec{\nu}^\top (A\vec{x} - \vec{y}). \tag{7.51}$$

The dual function is

$$g(\vec{\nu}) = \min_{\vec{x} \in \mathbb{R}^n} L(\vec{x}, \vec{\nu}) \tag{7.52}$$

$$= \min_{\vec{x} \in \mathbb{R}^n} \left\{ \|\vec{x}\|_2^2 + \vec{\nu}^\top (A\vec{x} - \vec{y}) \right\}. \tag{7.53}$$

The Lagrangian is convex in $\vec{x}$ and bounded below, and so it is minimized wherever its gradient in $\vec{x}$ is $\vec{0}$. The optimal $\vec{x}^\star$ is a function of $\vec{\nu}$, so we write it as $\vec{x}^\star(\vec{\nu})$. We have

$$\vec{0} = \nabla_{\vec{x}} L(\vec{x}^\star(\vec{\nu}), \vec{\nu}) \tag{7.54}$$

$$= 2\vec{x}^\star(\vec{\nu}) + A^\top \vec{\nu} \tag{7.55}$$

$$\implies \vec{x}^\star(\vec{\nu}) = -\frac{1}{2} A^\top \vec{\nu}. \tag{7.56}$$

Thus we can plug this optimal point back in to get g:

$$g(\vec{\nu}) = L(\vec{x}^\star(\vec{\nu}), \vec{\nu}) \tag{7.57}$$

$$= \|\vec{x}^\star(\vec{\nu})\|_2^2 + \vec{\nu}^\top (A\vec{x}^\star(\vec{\nu}) - \vec{y}) \tag{7.58}$$

$$= \left\|-\frac{1}{2}A^\top\vec{\nu}\right\|_2^2 + \vec{\nu}^\top \left(A\left(-\frac{1}{2}A^\top\vec{\nu}\right) - \vec{y}\right) \tag{7.59}$$

$$= \frac{1}{4}\vec{\nu}^\top AA^\top\vec{\nu} - \frac{1}{2}\vec{\nu}^\top AA^\top\vec{\nu} - \vec{\nu}^\top\vec{y} \tag{7.60}$$

$$= -\frac{1}{4}\vec{\nu}^\top AA^\top\vec{\nu} - \vec{\nu}^\top\vec{y}. \tag{7.61}$$

Thus the dual problem is

$$d^\star = \max_{\vec{\nu}\in\mathbb{R}^p}\left\{-\frac{1}{4}\vec{\nu}^\top AA^\top\vec{\nu} - \vec{\nu}^\top\vec{y}\right\} = -\min_{\vec{\nu}\in\mathbb{R}^p}\left\{\frac{1}{4}\vec{\nu}^\top AA^\top\vec{\nu} + \vec{\nu}^\top\vec{y}\right\}. \tag{7.62}$$

There are no constraints on the dual problem, because there are no inequality constraints on the primal problem. Since this dual problem is a convex unconstrained problem whose objective value is bounded below, it can be solved by setting the gradient of the objective to $\vec{0}$ and solving for the optimal dual variable. This yields

$$\vec{0} = \nabla g(\vec{\nu}^\star) \tag{7.63}$$

$$= \frac{1}{2}AA^\top\vec{\nu}^\star + \vec{y} \tag{7.64}$$

$$\implies \vec{\nu}^\star = -2(AA^\top)^{-1}\vec{y}. \tag{7.65}$$

Our original primal problem is convex, and all constraints are affine. As long as there is a feasible point $\vec{x}$, i.e., a point $\vec{x}$ such that $A\vec{x} = \vec{y}$, then Slater's condition implies that strong duality holds. Suppose that there is such a feasible point (i.e., the primal problem is feasible). Then strong duality holds. Because the primal problem is convex and strong duality holds, we can recover the optimal primal variable $\vec{x}^\star$ from the optimal dual variable $\vec{\nu}^\star$ as

$$\vec{x}^\star = \vec{x}^\star(\vec{\nu}^\star) = -\frac{1}{2}A^\top(-2(AA^\top)^{-1}\vec{y}) = A^\top(AA^\top)^{-1}\vec{y}. \tag{7.66}$$

This corresponds to the familiar minimum-norm solution.

Example 143. Consider a so-called *linear program*:

$$\begin{aligned} p^\star = \min_{\vec{x}\in\mathbb{R}^n} \quad & \vec{c}^\top\vec{x} \\ \text{s.t.} \quad & A\vec{x} = \vec{y} \\ & \vec{x} \geq \vec{0} \end{aligned} \tag{7.67}$$

where the last constraint means $x_i \geq 0$ for all i. Slater's condition implies that strong duality holds for this problem provided there is a feasible point. The Lagrangian is

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = \vec{c}^\top\vec{x} + \sum_{i=1}^{n}\lambda_i(-x_i) + \sum_{j=1}^{m}\nu_j(A\vec{x} - \vec{y})_j \tag{7.68}$$

$$= \vec{c}^\top\vec{x} - \vec{\lambda}^\top\vec{x} + \vec{\nu}^\top(A\vec{x} - \vec{y}) \tag{7.69}$$

$$= \left(\vec{c} + A^\top\vec{\nu} - \vec{\lambda}\right)^\top\vec{x} - \vec{\nu}^\top\vec{y}. \tag{7.70}$$

This function is affine in $\vec{x}$, hence convex in $\vec{x}$, so its dual function is

$$g(\vec{\lambda}, \vec{\nu}) = \min_{\vec{x}\in\mathbb{R}^n} L(\vec{x}, \vec{\lambda}, \vec{\nu}) \tag{7.71}$$

$$= \min_{\vec{x} \in \mathbb{R}^n} \left\{ \left(\vec{c} + A^\top \vec{\nu} - \vec{\lambda} \right)^\top \vec{x} - \vec{\nu}^\top \vec{y} \right\} \tag{7.72}$$

$$= \begin{cases} -\vec{\nu}^\top \vec{y}, & \text{if } \vec{c} + A^\top \vec{\nu} - \vec{\lambda} = \vec{0} \\ -\infty, & \text{otherwise.} \end{cases} \tag{7.73}$$

(It is a good exercise to figure out why this last equality is correct.) Thus, the dual is

$$\begin{aligned} d^\star = \max_{\substack{\vec{\lambda} \in \mathbb{R}^n \\ \vec{\nu} \in \mathbb{R}^m}} \quad & g(\vec{\lambda}, \vec{\nu}) \\ \text{s.t.} \quad & \lambda_i \geq 0, \qquad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{7.74}$$

Expanding out g, this problem is equivalent to

$$\begin{aligned} d^\star = \max_{\substack{\vec{\lambda} \in \mathbb{R}^n \\ \vec{\nu} \in \mathbb{R}^m}} \quad & -\vec{\nu}^\top \vec{y} \\ \text{s.t.} \quad & \lambda_i \geq 0, \qquad \forall i \in \{1, \ldots, n\} \\ & \vec{c} + A^\top \vec{\nu} - \vec{\lambda} = \vec{0}. \end{aligned} \tag{7.75}$$

Example 144 (Shadow Prices)**.** In this example, we determine an economic interpretation of Lagrange multipliers.

Suppose we have 200 kilos of merlot grapes and 300 kilos of shiraz grapes. Consider the following possible blends:

- 4 kilos merlot, 1 kilo shiraz for \$20 per bottle.
- 2 kilos merlot, 3 kilos shiraz for \$15 per bottle.

We want to maximize our profits. Suppose we make q_1 bottles of the first blend, and q_2 of the second. Our optimization problem is:

$$\begin{aligned} p^\star = \max_{q_1, q_2 \in \mathbb{R}} \quad & 20q_1 + 15q_2 \\ \text{s.t.} \quad & 4q_1 + 2q_2 \leq 200 \\ & q_1 + 3q_2 \leq 300 \\ & q_1 \geq 0 \\ & q_2 \geq 0. \end{aligned} \tag{7.76}$$

In reality, we can't make fractional bottles of wine, so we can round q_1 and q_2 to the nearest integer if needed. Now consider the following modification. Suppose that we actually want to sell off some of the grapes instead of turning them into wine. We can earn λ_1 dollars per kilo of merlot and λ_2 per kilo of shiraz. This yields a new optimization problem

$$\max_{q_1, q_2 \in \mathbb{R}_+} \{20q_1 + 15q_2 + \lambda_1(200 - 4q_1 - 2q_2) + \lambda_2(300 - q_1 - 3q_2)\} \tag{7.77}$$

$$= \max_{q_1, q_2 \in \mathbb{R}_+} \{(20 - 4\lambda_1 - \lambda_2)q_1 + (15 - 2\lambda_1 - 3\lambda_2)q_2 + 200\lambda_1 + 300\lambda_2\}. \tag{7.78}$$

If the coefficient of q_1 is negative, we shouldn't make any of the first blend. If the coefficient of q_2 is negative, we shouldn't make any of the second blend. If both are positive, we should make as many of each as possible. What happens when $20 - 4\lambda_1 - \lambda_2 = 0$ and $15 - 2\lambda_1 - 3\lambda_2 = 0$? This is an indifference point, i.e. the point at which our profit is the same no matter how many bottles of either wine we make. Under this condition, the minimum profit we could possibly make is

$$\begin{aligned} \min_{\lambda_1, \lambda_2 \in \mathbb{R}_+} \quad & 200\lambda_1 + 300\lambda_2 \\ \text{s.t.} \quad & 20 - 4\lambda_1 - \lambda_2 = 0 \\ & 15 - 2\lambda_1 - 3\lambda_2 = 0. \end{aligned} \tag{7.79}$$

This problem is the *dual* to our original (primal) problem. The λ_i's are called *shadow prices*, and capture how much we're willing to pay to violate our constraints.

7.4 Karush-Kuhn-Tucker (KKT) Conditions

We now go into some more involved theory which connects strong duality to optimality conditions.

A broadly applicable set of conditions which are sometimes necessary and sometimes sufficient for optimality is called the *Karush-Kuhn-Tucker* (KKT) conditions.

Definition 145 (KKT Conditions)
Let $(\vec{\widetilde{x}}, \vec{\widetilde{\lambda}}, \vec{\widetilde{\nu}}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ be a decision variable and Lagrange multipliers. Suppose that the objective function f_0 and constraint functions $f_1, \dots, f_m, h_1, \dots, h_p$ are differentiable. We say that $(\vec{\widetilde{x}}, \vec{\widetilde{\lambda}}, \vec{\widetilde{\nu}})$ fulfills the *KKT conditions* if

- (Primal feasibility.) $\vec{\widetilde{x}}$ is feasible for $\mathcal{P}$, i.e.,

$$f_i(\vec{\widetilde{x}}) \leq 0, \qquad \forall i \in \{1, \dots, m\} \tag{7.80}$$

$$\text{and} \quad h_j(\vec{\widetilde{x}}) = 0, \qquad \forall j \in \{1, \dots, p\}. \tag{7.81}$$

- (Dual feasibility.) $(\vec{\widetilde{\lambda}}, \vec{\widetilde{\nu}})$ is feasible for $\mathcal{D}$, i.e.,

$$\widetilde{\lambda}_i \geq 0, \qquad \forall i \in \{1, \dots, m\}. \tag{7.82}$$

- (Complementary slackness.)

$$\widetilde{\lambda}_i f_i(\vec{\widetilde{x}}) = 0, \qquad \forall i \in \{1, \dots, m\}. \tag{7.83}$$

- (Stationarity, or first-order condition.)

$$\vec{0} = \nabla_{\vec{x}} L(\vec{\widetilde{x}}, \vec{\widetilde{\lambda}}, \vec{\widetilde{\nu}}) \tag{7.84}$$

$$= \nabla f(\vec{\widetilde{x}}) + \sum_{i=1}^{m} \widetilde{\lambda}_i \nabla f_i(\vec{\widetilde{x}}) + \sum_{j=1}^{p} \widetilde{\nu}_j \nabla h_j(\vec{\widetilde{x}}). \tag{7.85}$$

Given an arbitrary optimization problem, the KKT conditions do *not* have to be related to the optimality conditions. But it turns out that in many cases, they are related.

Theorem 146 (If Strong Duality Holds, then KKT Conditions are Necessary for Optimality)
Suppose $\mathcal{P}$ is a primal problem with dual $\mathcal{D}$. Suppose that the objective function f_0 and constraint functions

$f_1, \dots, f_m, h_1, \dots, h_p$ are differentiable, strong duality holds, and $(\vec{x}^\star, \vec{\lambda}^\star, \vec{\nu}^\star)$ are optimal primal and dual variables. Then $(\vec{x}^\star, \vec{\lambda}^\star, \vec{\nu}^\star)$ fulfill the KKT conditions.

Proof. By assumption, $\vec{x}^\star$ is feasible for $\mathcal{P}$ and $(\vec{\lambda}^\star, \vec{\nu}^\star)$ is feasible for $\mathcal{D}$. This implies that $\lambda_i^\star f_i(\vec{x}^\star) \leq 0$ for all i, and $\nu_j^\star h_j(\vec{x}^\star) = 0$ for all j. Thus, we have

$$\begin{aligned} d^\star &= g(\vec{\lambda}^\star, \vec{\nu}^\star) && (7.86)\\ &= \min_{\vec{x} \in \mathbb{R}^n} L(\vec{x}, \vec{\lambda}^\star, \vec{\nu}^\star) && (7.87)\\ &\leq L(\vec{x}^\star, \vec{\lambda}^\star, \vec{\nu}^\star) && (7.88)\\ &= f_0(\vec{x}^\star) + \sum_{i=1}^{m} \underbrace{\lambda_i^\star f_i(\vec{x}^\star)}_{\leq 0} + \sum_{j=1}^{p} \underbrace{\nu_j^\star h_j(\vec{x}^\star)}_{=0} && (7.89)\\ &\leq f_0(\vec{x}^\star) && (7.90)\\ &= p^\star. && (7.91) \end{aligned}$$

Because $p^\star = d^\star$, all inequalities in the above chain are actually equalities. This means that $\vec{x}^\star$ minimizes $L(\vec{x}, \vec{\lambda}^\star, \vec{\nu}^\star)$ over $\vec{x} \in \mathbb{R}^n$. By the main theorem of vector calculus, this implies that $\nabla_{\vec{x}} L(\vec{x}^\star, \vec{\lambda}^\star, \vec{\nu}^\star) = \vec{0}$, which is the stationarity condition. It also implies that $\sum_{i=1}^{m} \lambda_i^\star f_i(\vec{x}^\star) = 0$. But each term in the sum is ≤ 0, so they must all be 0. This means that $\lambda_i^\star f_i(\vec{x}^\star) = 0$ for each i. This gives the complementary slackness condition. Thus the KKT conditions hold for $(\vec{x}^\star, \vec{\lambda}^\star, \vec{\nu}^\star)$. □

Theorem 147 (If Convexity Holds, then KKT Conditions are Sufficient for Optimality)
Suppose $\mathcal{P}$ is a *convex* primal problem with dual $\mathcal{D}$. Suppose that the objective function f_0 and constraint functions $f_1, \dots, f_m, h_1, \dots, h_p$ of $\mathcal{P}$ are differentiable. Suppose that $(\tilde{\vec{x}}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}})$ fulfill the KKT conditions. Then strong duality holds and $(\tilde{\vec{x}}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}})$ are optimal primal and dual variables.

Proof. By assumption, $\tilde{\vec{x}}$ is feasible for $\mathcal{P}$ and $(\tilde{\vec{\lambda}}, \tilde{\vec{\nu}})$ is feasible for $\mathcal{D}$. Since $\mathcal{P}$ is convex, the f_i are convex and h_i are affine. Thus

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = f_0(\vec{x}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{x}) + \sum_{j=1}^{p} \nu_j h_j(\vec{x}) \tag{7.92}$$

is convex in $\vec{x}$. Since stationarity holds, we have

$$\nabla_{\vec{x}} L(\tilde{\vec{x}}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}}) = \vec{0}, \tag{7.93}$$

so because the primal problem is convex, we have that $\tilde{\vec{x}}$ minimizes $L(\vec{x}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}})$ over all $\vec{x} \in \mathbb{R}^n$. Thus

$$\begin{aligned} g(\tilde{\vec{\lambda}}, \tilde{\vec{\nu}}) &= \min_{\vec{x} \in \mathbb{R}^n} L(\vec{x}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}}) && (7.94)\\ &= L(\tilde{\vec{x}}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}}) && (7.95)\\ &= f_0(\tilde{\vec{x}}) + \sum_{i=1}^{m} \underbrace{\tilde{\lambda}_i f_i(\tilde{\vec{x}})}_{=0} + \sum_{j=1}^{p} \underbrace{\tilde{\nu}_i h_i(\tilde{\vec{x}})}_{=0} && (7.96)\\ &= f_0(\tilde{\vec{x}}). && (7.97) \end{aligned}$$

Thus $f_0(\tilde{\vec{x}}) = g(\tilde{\vec{\lambda}}, \tilde{\vec{\nu}})$, so the duality gap is 0 and strong duality holds, with $(\tilde{\vec{x}}, \tilde{\vec{\lambda}}, \tilde{\vec{\nu}})$ being optimal primal and dual variables. □

In this course, most problems will be convex and strong duality will hold; in this case the above two theorems can apply and we get that the KKT conditions are *equivalent* to optimality conditions. This is the source of the intuition that convex problems are easier to optimize.

Corollary 148 (If Convexity and Strong Duality Hold, then KKT Conditions are Necessary and Sufficient for Optimality)**.** *Suppose $\mathcal{P}$ is a convex primal problem with dual $\mathcal{D}$. Suppose strong duality holds for $\mathcal{P}$ and $\mathcal{D}$. Suppose that the objective function f_0 and constraint functions $f_1, \ldots, f_m, h_1, \ldots, h_p$ for $\mathcal{P}$ are differentiable. Let $(\vec{x}, \vec{\lambda}, \vec{\nu}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ be primal and dual variables. Then $(\vec{x}, \vec{\lambda}, \vec{\nu})$ are optimal primal and dual variables if and only if they fulfill the KKT conditions.*

The following is a generic sequence of steps that you can try for any convex optimization problem.

Problem Solving Strategy 149 (Solving Convex Optimization Problems Using KKT Conditions)**.** *Suppose you have a problem $\mathcal{P}$ with dual $\mathcal{D}$.*

(i) Show that $\mathcal{P}$ is convex and the objective and constraint functions are differentiable.

(ii) Show Slater's condition holds and/or that strong duality holds for $\mathcal{P}$ and $\mathcal{D}$.

(iii) Compute the KKT conditions for $\mathcal{P}$ and $\mathcal{D}$.

(iv) Solve for the optimal primal and dual variables using the KKT conditions.

Even for more complicated problems where it is not possible to solve the KKT conditions analytically, many algorithms can be interpreted as solving the KKT conditions under various conditions.

Chapter 8

Types of Optimization Problems

Relevant sections of the textbooks:

- [1] Chapter 4.
- [2] Chapters 9, 10, 12.

8.1 Linear Programs

In this chapter we introduce a *taxonomy* of common optimization problems that can be efficiently solved through a variety of ways. We use the notation $\vec{u} \geq \vec{v}$ to denote $u_i \geq v_i$ for all i.

A linear program is one with a linear objective and linear constraints. It is the most restrictive (e.g. smallest) class in the taxonomy we present. It has a *standard form* denoted below.

Definition 150 (Linear Program)
A *linear program* (LP) is an optimization problem with a linear objective and affine constraints. A *standard form linear program* is an optimization problem of the following form:

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & \vec{c}^\top \vec{x} \\ \text{s.t.} \quad & A\vec{x} = \vec{y} \\ & \vec{x} \geq \vec{0}. \end{aligned} \tag{8.1}$$

There are many equivalent forms of a linear program; in particular, the following proposition (whose proof we leave as an exercise) can be shown using slack variables.

Proposition 151
Any linear program is equivalent to a standard form linear program.

Putting linear programs in standard form is important because the standard form is commonly accepted by optimization algorithms and implementations. Usually if you provide a linear program that isn't in standard form to a solver, they will convert it to standard form first before running their algorithms. Conversions to-and-from standard form may increase the number of variables in the problem and the eventual algorithmic complexity of solving it. One example of this conversion is done below.

Example 152. Suppose we have the linear program

$$\min_{\vec{x}\in\mathbb{R}^2} \quad 2x_1 + 4x_2 \tag{8.2}$$
$$\text{s.t.} \quad x_1 + x_2 \geq 3 \tag{8.3}$$
$$3x_1 + 2x_2 = 14 \tag{8.4}$$
$$x_1 \geq 0. \tag{8.5}$$

To convert this to standard form, first we note that

$$x_1 + x_2 \geq 3 \iff -x_1 - x_2 \leq -3 \tag{8.6}$$

which obtains the following system:

$$\min_{\vec{x}\in\mathbb{R}^2} \quad 2x_1 + 4x_2 \tag{8.7}$$
$$\text{s.t.} \quad -x_1 - x_2 \leq -3 \tag{8.8}$$
$$3x_1 + 2x_2 = 14 \tag{8.9}$$
$$x_1 \geq 0. \tag{8.10}$$

Adding a slack variable $x_3 \geq 0$ to the first constraint yields equality:

$$\min_{\vec{x}\in\mathbb{R}^3} \quad 2x_1 + 4x_2 \tag{8.11}$$
$$\text{s.t.} \quad -x_1 - x_2 + x_3 = -3 \tag{8.12}$$
$$3x_1 + 2x_2 = 14 \tag{8.13}$$
$$x_1 \geq 0 \tag{8.14}$$
$$x_3 \geq 0. \tag{8.15}$$

The only reason this is not in standard form is that x_2 is unconstrained. We can represent any real number as the difference of non-negative numbers; one such construction is $x_2 = x_2^+ - x_2^-$ where $x_2^+ = \max\{x_2, 0\}$ and $x_2^- = -\min\{x_2, 0\}$, but there are many others. Thus we can replace x_2 by $x_4 - x_5$ and add the constraints that $x_4 \geq 0$ and $x_5 \geq 0$, obtaining the problem

$$\min_{\vec{x}\in\mathbb{R}^5} \quad 2x_1 + 4x_4 - 4x_5 \tag{8.16}$$
$$\text{s.t.} \quad -x_1 + x_3 - x_4 + x_5 = -3 \tag{8.17}$$
$$3x_1 + 2x_4 - 2x_5 = 14 \tag{8.18}$$
$$x_1 \geq 0 \tag{8.19}$$
$$x_3 \geq 0 \tag{8.20}$$
$$x_4 \geq 0 \tag{8.21}$$
$$x_5 \geq 0. \tag{8.22}$$

Finally, we notice that x_2 is neither in the objective nor the constraints and so can be eliminated.

$$\min_{\vec{x}\in\mathbb{R}^4} \quad 2x_1 + 4x_3 - 4x_4 \tag{8.23}$$
$$\text{s.t.} \quad -x_1 + x_2 - x_3 + x_4 = -3 \tag{8.24}$$

$$3x_1 + 2x_3 - 2x_4 = 14 \tag{8.25}$$
$$x_1 \geq 0 \tag{8.26}$$
$$x_2 \geq 0 \tag{8.27}$$
$$x_3 \geq 0 \tag{8.28}$$
$$x_4 \geq 0. \tag{8.29}$$

Linear programs are convex problems, and in particular the feasible set of a linear program is a convex set.

Proposition 153

Any linear program is a convex optimization problem.

Proposition 154

The dual of the standard form linear program

$$\begin{aligned} p^\star = \min_{\vec{x}\in\mathbb{R}^n} \quad & \vec{c}^\top \vec{x} \\ \text{s.t.} \quad & A\vec{x} = \vec{y} \\ & \vec{x} \geq \vec{0} \end{aligned} \tag{8.30}$$

is

$$\begin{aligned} d^\star = \max_{\substack{\vec{\lambda}\in\mathbb{R}^n \\ \vec{\nu}\in\mathbb{R}^m}} \quad & -\vec{y}^\top \vec{\nu} \\ \text{s.t.} \quad & \vec{c} - \vec{\lambda} + A^\top \vec{\nu} = \vec{0} \\ & \vec{\lambda} \geq \vec{0}. \end{aligned} \tag{8.31}$$

Proof. The Lagrangian is

$$L(\vec{x}, \vec{\lambda}, \vec{\nu}) = \vec{c}^\top \vec{x} - \vec{\lambda}^\top \vec{x} + \vec{\nu}^\top (A\vec{x} - \vec{y}) \tag{8.32}$$
$$= (\vec{c} - \vec{\lambda} + A^\top \vec{\nu})^\top \vec{x} - \vec{\nu}^\top \vec{y}. \tag{8.33}$$

The dual function is

$$g(\vec{\lambda}, \vec{\nu}) = \min_{\vec{x}\in\mathbb{R}^n} L(\vec{x}, \vec{\lambda}, \vec{\nu}) \tag{8.34}$$
$$= \min_{\vec{x}\in\mathbb{R}^n} \left\{ (\vec{c} - \vec{\lambda} + A^\top \vec{\nu})^\top \vec{x} - \vec{\nu}^\top \vec{y} \right\} \tag{8.35}$$
$$= \min_{\vec{x}\in\mathbb{R}^n} \left\{ (\vec{c} - \vec{\lambda} + A^\top \vec{\nu})^\top \vec{x} \right\} - \vec{\nu}^\top \vec{y} \tag{8.36}$$
$$= \begin{cases} -\vec{\nu}^\top \vec{y}, & \text{if } \vec{c} - \vec{\lambda} + A^\top \vec{\nu} = \vec{0} \\ -\infty, & \text{otherwise.} \end{cases} \tag{8.37}$$

The dual problem is

$$d^\star = \max_{\substack{\vec{\lambda}\in\mathbb{R}^n \\ \vec{\nu}\in\mathbb{R}^m}} \quad g(\vec{\lambda}, \vec{\nu}) \tag{8.38}$$

$$\text{s.t.} \quad \vec{\lambda} \geq \vec{0}. \tag{8.39}$$

Expanding this out and adding the domain constraint, we get

$$\begin{aligned} d^\star = \max_{\substack{\vec{\lambda} \in \mathbb{R}^n \\ \vec{\nu} \in \mathbb{R}^m}} \quad & -\vec{\nu}^\top \vec{y} && (8.40) \\ \text{s.t.} \quad & \vec{c} - \vec{\lambda} + A^\top \vec{\nu} = \vec{0} && (8.41) \\ & \vec{\lambda} \geq \vec{0} && (8.42) \end{aligned}$$

as desired. □

One very relevant question is how to solve linear programs efficiently. There are several methods available to us — for example, any constrained convex optimization solver, such as projected gradient descent, will solve the problem, given an appropriate learning rate and efficient projection method. The algorithm most used in practice is the interior point method; we will learn the basics of interior point methods later in this course.[1] But there is one algorithm which was used for many years in practice, that truly exploits the structure of linear programs to efficiently solve them. It is called the *simplex algorithm*, which was invented by George Dantzig in 1947.

Definition 155 (Polyhedron)

A *polyhedron* is a set of the form $\{\vec{x} \in \mathbb{R}^n \mid A\vec{x} = \vec{y}, \vec{x} \geq \vec{0}\}$. In other words, it is the feasible set of a linear program.

Definition 156 (Extreme Point)

Let $K \subseteq \mathbb{R}^n$ be a set. We say that $\vec{x} \in K$ is an *extreme point* of K if there *does not* exist $\vec{y}, \vec{z} \in K \setminus \{\vec{x}\}$ and $\theta \in [0, 1]$ such that $\vec{x} = \theta\vec{y} + (1 - \theta)\vec{z}$.

An extreme point of a polyhedron is called a *vertex* of the polyhedron. We now try to figure out when polyhedra have vertices.

Proposition 157

A polyhedron has a vertex if and only if it does not contain a line.

Intuitively one considers a vertex to be the intersection of two or more constraint regions. If there are no intersection points then the constraints never intersect and thus there is space to fit a line.

Theorem 158 (Main Theorem of Linear Programming)

Consider the standard form linear program:

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & \vec{c}^\top \vec{x} \\ \text{s.t.} \quad & A\vec{x} = \vec{y} \\ & \vec{x} \geq \vec{0}. \end{aligned} \tag{8.43}$$

Assume that:

[1] Talk to your TA Tarun if you want to learn more about interior point methods!

(i) the polyhedron $\Omega \doteq \left\{\vec{x} \in \mathbb{R}^n \mid A\vec{x} = \vec{y}, \vec{x} \geq \vec{0}\right\}$ has a vertex;

(ii) $p^\star > -\infty$ is finite; and

(iii) there exists some $\vec{x}^\star \in \Omega$ such that $\vec{c}^\top \vec{x}^\star = p^\star$.

Then there exists a *vertex* $\vec{v} \in \Omega$ such that $\vec{c}^\top \vec{v} = p^\star$.

Intuitively, what this says is that to solve a linear program, we only need to check the vertices of the constraint polyhedron. This reduces an optimization problem over $\mathbb{R}^n$ to an optimization problem over the finite set of vertices.

Proof. Let Γ be the set of all optimal solutions, i.e., $\Gamma = \{\vec{x} \in \Omega \mid \vec{c}^\top \vec{x} = p^\star\}$. Since Ω is a polyhedron and we have added one more affine constraint, Γ is a polyhedron. Since Ω does not contain a line and $\Gamma \subseteq \Omega$, we have that Γ does not contain a line. Thus Γ has a vertex.

Let $\vec{v}$ be a vertex of Γ. We want to show that it is a vertex of Ω, which will show that there is an optimal solution which is a vertex of Ω.

Suppose for the sake of contradiction that $\vec{v}$ is a vertex of Γ but not a vertex of Ω. Then there exist $\vec{y}, \vec{z} \in \Omega$ with $\vec{y} \neq \vec{z}$ and $\theta \in (0, 1)$ such that $\vec{v} = \theta\vec{y} + (1-\theta)\vec{z}$. We know that $\vec{c}^\top \vec{v} = p^\star$ since $\vec{v} \in \Gamma$, but we also know that $\vec{c}^\top \vec{y} \geq p^\star$ and $\vec{c}^\top \vec{z} \geq p^\star$ since $\vec{y}, \vec{z} \in \Omega$. Thus we have that

$$p^\star = \vec{c}^\top \vec{v} = \vec{c}^\top(\theta\vec{y} + (1-\theta)\vec{z}) = \theta \underbrace{\vec{c}^\top \vec{y}}_{\geq p^\star} + (1-\theta) \underbrace{\vec{c}^\top \vec{z}}_{\geq p^\star} \geq p^\star. \tag{8.44}$$

Since $p^\star = p^\star$, all inequalities in the above chain are actually equalities. In particular, we have $\vec{c}^\top \vec{y} = \vec{c}^\top \vec{z} = p^\star$. Thus $\vec{y} \in \Gamma$ and $\vec{z} \in \Gamma$. Thus $\vec{v} = \theta\vec{y} + (1-\theta)\vec{z}$ is not a vertex of Γ, a contradiction. $\square$

This theorem proposes a "greedy-like heuristic" to implement a solver for linear programs with bounded feasible sets Ω, which is called the *simplex method*. The simplex method is the following procedure:

- Start at a vertex $\vec{v}$ of Ω.
- While there is a neighboring vertex $\vec{w}$ with $\vec{c}^\top \vec{v} > \vec{c}^\top \vec{w}$, move to it, i.e., set $\vec{v} \leftarrow \vec{w}$.
- When there are no neighboring vertices with better optima, stop and return $\vec{v}$.

There are (rather more technical) modifications one can make to this algorithm to solve linear programs with unbounded feasible sets. But the main idea is just the same as gradient descent: iteratively search locally for another point with better objective value, and move to it.

8.2 Quadratic Programs

Definition 159 (Quadratic Program)

A *quadratic program* (QP) is an optimization problem with a quadratic objective and affine constraints. A *standard form quadratic program* is an optimization problem of the following form:

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & \frac{1}{2}\vec{x}^\top H \vec{x} + \vec{c}^\top \vec{x} \\ \text{s.t.} \quad & A\vec{x} \leq \vec{y} \\ & C\vec{x} = \vec{z}, \end{aligned} \tag{8.45}$$

where $H \in \mathbb{S}^n$.

In the standard form, we do not lose any generality by enforcing $H \in \mathbb{S}^n$. In particular, for any H we have

$$\frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x} = \frac{1}{2}\vec{x}^\top\left(\frac{H + H^\top}{2}\right)\vec{x} + \vec{c}^\top\vec{x} \tag{8.46}$$

whence the matrix $\frac{H+H^\top}{2}$ (i.e., the *symmetric part* of H) is always symmetric. So if we have a non-symmetric H we can just replace it with its symmetric part, and thus obtain a standard form quadratic program.

Quadratic programs may or may not be convex.

Proposition 160

Consider the following standard form quadratic program:

$$\begin{aligned} p^\star = \min_{\vec{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x} \\ \text{s.t.} \quad & A\vec{x} \leq \vec{y} \\ & C\vec{x} = \vec{z}, \end{aligned} \tag{8.47}$$

where $H \in \mathbb{S}^n$. The following are equivalent:

(i) The problem is convex.

(ii) $H \in \mathbb{S}^n_+$.

Proof. The gradient and Hessian of the objective are

$$\nabla\left(\frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x}\right) = \frac{1}{2}(H + H^\top)\vec{x} + \vec{c} = H\vec{x} + \vec{c} \tag{8.48}$$

$$\nabla^2\left(\frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x}\right) = H \tag{8.49}$$

so the objective function is convex if and only if $H \in \mathbb{S}^n_+$. Thus the problem is a convex problem if and only if $H \in \mathbb{S}^n_+$ and the constraint set is convex. But the constraint set is always convex because it is defined by a set of linear equations and inequalities. Thus the problem is convex if and only if $H \in \mathbb{S}^n_+$. □

Example 161. Let us consider the unconstrained quadratic program

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n}\left(\frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x}\right) \tag{8.50}$$

where $H \in \mathbb{S}^n$, and solve it in a variety of cases, namely that where $H \notin \mathbb{S}^n_+$, $H \in \mathbb{S}^n_+$ with $\vec{c} \in \mathcal{N}(H) \setminus \{\vec{0}\}$, and $H \in \mathbb{S}^n_+$ with $\vec{c} \in \mathcal{N}(H)^\perp = \mathcal{R}(H^\top) = \mathcal{R}(H)$.

Case 1. Suppose that $H \notin \mathbb{S}^n_+$. Then H has a negative eigenvalue λ; let $\vec{v}$ be any corresponding unit eigenvector. Then, if we choose $\vec{x}_t = t \cdot \vec{v}$, we get

$$\begin{aligned} p^\star &\leq \lim_{t\to\infty}\left(\frac{1}{2}\vec{x}_t^\top H\vec{x}_t + \vec{c}^\top\vec{x}_t\right) && (8.51) \\ &= \lim_{t\to\infty}\left(\frac{1}{2}t^2\vec{v}^\top H\vec{v} + t\vec{c}^\top\vec{v}\right). && (8.52) \end{aligned}$$

To bound the terms inside the limit, we compute

$$\vec{v}^\top H\vec{v} = \vec{v}^\top(\lambda\vec{v}) = \lambda\vec{v}^\top\vec{v} = \lambda \tag{8.53}$$

$$\vec{c}^\top\vec{v} \leq \|\vec{c}\|_2\,\|\vec{v}\|_2 = \|\vec{c}\|_2\,. \tag{8.54}$$

Thus we have

$$p^\star \leq \lim_{t\to\infty}\left(\frac{1}{2}\lambda t^2 + t\,\|\vec{v}\|_2\right). \tag{8.55}$$

Since $\lambda < 0$, the term inside the limit is a concave (i.e., downward facing) quadratic function of t, and so its limit as $t \to \pm\infty$ is $-\infty$. Thus $p^\star \leq -\infty$ so $p^\star = -\infty$.

Case 2. Suppose that $H \in \mathbb{S}^n_+$, and suppose that $\vec{c} \in \mathcal{N}(H) \setminus \{0\}$. Then H has (at least) an eigenvalue λ equal to 0, and in particular, by the spectral theorem $\vec{c}$ can be written as a linear combination of eigenvectors of H with eigenvalue 0. Let $\vec{v}$ be any unit eigenvector with eigenvalue 0 such that $\vec{c}^\top\vec{v} \neq 0$. Let $\vec{x}_t = -t\cdot\operatorname{sgn}\left(\vec{c}^\top\vec{v}\right)\cdot\vec{v}$. Then

$$p^\star \leq \lim_{t\to\infty}\left(\frac{1}{2}\vec{x}_t^\top H\vec{x}_t + \vec{c}^\top\vec{x}_t\right) \tag{8.56}$$

$$= \lim_{t\to\infty}\left(\frac{1}{2}t^2\vec{v}^\top H\vec{v} - t\cdot\operatorname{sgn}\left(\vec{c}^\top\vec{v}\right)\cdot\vec{c}^\top\vec{v}\right) \tag{8.57}$$

$$= \lim_{t\to\infty}\left(\frac{1}{2}t^2\vec{v}^\top\vec{0} - t\cdot\left|\vec{c}^\top\vec{v}\right|\right) \tag{8.58}$$

$$= \lim_{t\to\infty}\left(-t\cdot\left|\vec{c}^\top\vec{v}\right|\right) \tag{8.59}$$

$$= \lim_{t\to\infty} t\cdot\underbrace{\left(-\left|\vec{c}^\top\vec{v}\right|\right)}_{<0} \tag{8.60}$$

$$= -\infty. \tag{8.61}$$

Thus $p^\star \leq -\infty$ so $p^\star = -\infty$.

Case 3. Suppose that $H \in \mathbb{S}^n_+$ with $\vec{c} \in \mathcal{R}(H)$. Then there is nonzero $\vec{x}_0$ such that $\vec{c} = -H\vec{x}_0$. Rewriting the objective, we obtain

$$\frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x} = \frac{1}{2}\vec{x}^\top H\vec{x} - \vec{x}_0^\top H\vec{x} \tag{8.62}$$

$$= \frac{1}{2}\vec{x}^\top H\vec{x} - \vec{x}_0^\top H\vec{x} + \frac{1}{2}\vec{x}_0^\top H\vec{x}_0 - \frac{1}{2}\vec{x}_0^\top H\vec{x}_0 \tag{8.63}$$

$$= \frac{1}{2}\left(\vec{x}^\top H\vec{x} - 2\vec{x}_0^\top H\vec{x} + \vec{x}_0^\top H\vec{x}_0\right) - \frac{1}{2}\vec{x}_0^\top H\vec{x}_0 \tag{8.64}$$

$$= \frac{1}{2}(\vec{x} - \vec{x}_0)^\top H(\vec{x} - \vec{x}_0) - \frac{1}{2}\vec{x}_0^\top H\vec{x}_0. \tag{8.65}$$

Since $H \in \mathbb{S}^n_+$, the minimizer is at any $\vec{x}$ such that $\vec{x} - \vec{x}_0 \in \mathcal{N}(H)$. One can write this as $\vec{x} \in \vec{x}_0 + \mathcal{N}(H)$. A particular solution in terms of problem parameters is $\vec{x} = -H^\dagger\vec{c}$ where $H^\dagger$ is the Moore-Penrose pseudoinverse of H. Recall that we discussed the Moore-Penrose pseudoinverse in more generality in homework where we derived the solution to the least-norm least-squares problem, but one can show that if $H = U\Lambda U^\top$ then $H^\dagger = U\Lambda^\dagger U^\top$ where $\Lambda^\dagger$ is the diagonal matrix whose entries are $\Lambda^\dagger_{ii} = \begin{cases} 1/\Lambda_{ii}, & \text{if } \Lambda_{ii} \neq 0 \\ 0, & \text{if } \Lambda_{ii} = 0 \end{cases}$.

The previous example shows that we can solve unconstrained quadratic programs directly and read off the solutions. It turns out that one can transform any quadratic program with equality constraints into an unconstrained quadratic program. So really, this analysis encapsulates a huge class of quadratic programs.

Computing the dual of a quadratic program has a similar number of cases; it is an exercise which is left to homework.

Example 162 (Linear-Quadratic Regulator)**.** Suppose we have a discrete-time dynamical system, of the form

$$\begin{aligned} \vec{x}_{t+1} &= A\vec{x}_t + B\vec{u}_t, \qquad \forall t \geq 0 && (8.66) \\ \vec{x}_0 &= \vec{\xi}. && (8.67) \end{aligned}$$

Then one can show that

$$\vec{x}_t = A^t\vec{\xi} + \sum_{k=0}^{t-1} A^{t-k-1}B\vec{u}_k. \tag{8.68}$$

For a fixed terminal time T, we want to reach goal state $\vec{g}$. Namely, we want to solve the problem

$$\begin{aligned} \min_{\substack{\vec{x}_0,\dots,\vec{x}_T \\ \vec{u}_0,\dots,\vec{u}_{T-1}}} \quad & \|\vec{x}_T - \vec{g}\|_2^2 + \sum_{k=0}^{T-1} \|\vec{u}_k\|_2^2 && (8.69) \\ \text{s.t.} \quad & \vec{x}_{t+1} = A\vec{x}_t + B\vec{u}_t, \qquad \forall t \in \{0, 1, \dots, T-1\} && (8.70) \\ & \vec{x}_0 = \vec{\xi}. && (8.71) \end{aligned}$$

This is a quadratic program since the objective function is a quadratic function of each $\vec{x}_t$, and the constraints are affine equations relating the $\vec{x}_t$ and the $\vec{u}_t$.

As a last note, problems with quadratic objectives and quadratic inequality constraints are called *quadratically constrained quadratic programs* (QCQPs). Like quadratic programs, QCQPs can be convex or non-convex.

8.3 Quadratically-Constrained Quadratic Programs

Definition 163 (Quadratically-Constrained Quadratic Program)

A *quadratically-constrained quadratic program* (QCQP) is an optimization problem with a quadratic objective and quadratic constraints. A *standard form quadratically constrained quadratic program* is an optimization problem of the following form:

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & \frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x} && (8.72) \\ \text{s.t.} \quad & \frac{1}{2}\vec{x}^\top P_i\vec{x} + \vec{b}_i^\top\vec{x} + c_i \leq 0, \qquad \forall i \in \{1, \dots, m\} \\ & \frac{1}{2}\vec{x}^\top Q_i\vec{x} + \vec{d}_i^\top\vec{x} + f_i = 0, \qquad \forall i \in \{1, \dots, p\}, \end{aligned}$$

where $H, P_1, \dots, P_m, Q_1, \dots, Q_p \in \mathbb{S}^n$.

Proposition 164

Consider the following standard form quadratically-constrained quadratic program:

$$\begin{aligned} p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad & \frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x} && (8.73) \\ \text{s.t.} \quad & \frac{1}{2}\vec{x}^\top P_i\vec{x} + \vec{b}_i^\top\vec{x} + c_i \leq 0, \qquad \forall i \in \{1, \dots, m\} \\ & \frac{1}{2}\vec{x}^\top Q_i\vec{x} + \vec{d}_i^\top\vec{x} + f_i = 0, \qquad \forall i \in \{1, \dots, p\}, \end{aligned}$$

where $H, P_1, \ldots, P_m, Q_1, \ldots, Q_p \in \mathbb{S}^n$. The following are equivalent:

(i) The problem is convex.

(ii) $H, P_1, \ldots, P_m \in \mathbb{S}^n_+$ and $Q_1 = \cdots = Q_p = 0$.

Proof. Left as exercise. □

8.4 Second-Order Cone Programs

There is one more broad class of problems that we consider in this course, called *second-order cone programs* (SOCPs). They are among the broadest class of problems that we can efficiently solve using algorithms such as the interior point method, which we may discuss later in the course.

Definition 165 (Cone)
Let $C \subseteq \mathbb{R}^n$ be a set.

(a) We say that C is a *cone* if for all $\vec{x} \in C$ and $\alpha \geq 0$ we have $\alpha\vec{x} \in C$.

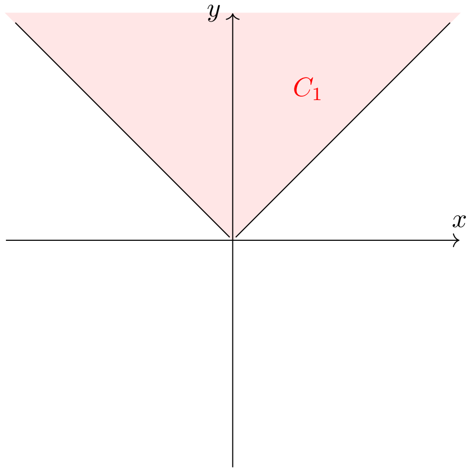
(b) We say that C is a *convex cone* if for all $\vec{x}, \vec{y} \in C$ and $\alpha, \beta \geq 0$ we have $\alpha\vec{x} + \beta\vec{y} \in C$.

By setting $\beta = 0$, we see that a convex cone is a cone, and a convex cone is convex. (This is good — otherwise, our definitions might not make too much sense!)

Example 166. The sets

$$C_1 \doteq \{(\vec{x}, y) \in \mathbb{R}^{n+1} \mid \|\vec{x}\|_2 \leq y\} \tag{8.74}$$
$$C_2 \doteq \{(x, y) \in \mathbb{R}^2 \mid y \geq 0\} \tag{8.75}$$

are cones. If $n = 1$ then C_1 looks like:



Now, we discuss some important classes of cones.

Definition 167

(a) A set of the form
$$C \doteq \{(\vec{x}, t) \in \mathbb{R}^{n+1} \mid A\vec{x} \leq t \cdot \vec{y}, t \geq 0\} \tag{8.76}$$
is called a *polyhedral cone*, and in particular corresponds to the polyhedron $\{\vec{x} \in \mathbb{R}^n \mid A\vec{x} \leq \vec{y}\}$.

(b) A set of the form
$$C \doteq \{(\vec{x}, t) \in \mathbb{R}^{n+1} \mid \|A\vec{x} - t\vec{y}\|_2 \leq tz, t \geq 0\} \tag{8.77}$$
is called a *ellipsoidal cone*, and in particular corresponds to the ellipse $\{\vec{x} \in \mathbb{R}^n \mid \|A\vec{x} - \vec{y}\|_2 \leq z\}$.

(c) A very special type of ellipsoidal cone is the *second order cone*:
$$C \doteq \{(\vec{x}, t) \in \mathbb{R}^{n+1} \mid \|\vec{x}\|_2 \leq t\}. \tag{8.78}$$
This corresponds to the special ellipse which is the unit circle.

Proposition 168
Polyhedral, ellipsoidal, and second-order cones are convex cones.

Proof. Left as an exercise. □

This allows us to define the notion of the second-order cone program.

Definition 169 (Second-Order Cone Program)
A *second-order cone program* is an optimization problem with a linear objective and affine and "second-order cone constraints", i.e., constraints which say that an affine function of $\vec{x}$ is contained in the second-order cone. A *standard form second-order cone program* is an optimization problem of the following form:

$$p^\star = \min_{\vec{x} \in \mathbb{R}^n} \quad \vec{c}^\top \vec{x} \tag{8.79}$$
$$\text{s.t.} \quad \|A_i\vec{x} - \vec{y}_i\|_2 \leq \vec{b}_i^\top \vec{x} + z_i, \qquad \forall i \in \{1, \ldots, m\}. \tag{8.80}$$

Second order cone constraints are strictly more broad than affine constraints; to encode an affine constraint $A_i\vec{x} = \vec{y}_i$ as a second-order cone constraint, pick the corresponding $\vec{b}_i = \vec{0}$ and $z_i = 0$. This makes the constraint $\|A_i\vec{x} - \vec{y}_i\|_2 \leq 0$ or equivalently $A_i\vec{x} = \vec{y}_i$.

Proposition 170
Second-order cone problems are convex.

Proof. Each second-order cone constraint $\|A_i\vec{x} - \vec{y}_i\|_2 \leq \vec{b}_i^\top \vec{x} + z_i$ can be alternatively formulated as constraining the tuple $(A_i\vec{x} - \vec{y}_i, \vec{b}_i^\top \vec{x} + z_i) \in \mathbb{R}^{n+1}$ to lie within the second-order cone in $\mathbb{R}^{n+1}$. But this tuple is an affine transformation of $\vec{x}$, in particular
$$\begin{bmatrix} A_i\vec{x} - \vec{y}_i \\ \vec{b}_i^\top \vec{x} + z_i \end{bmatrix} = \begin{bmatrix} A_i \\ \vec{b}_i^\top \end{bmatrix} \vec{x} + \begin{bmatrix} -\vec{y}_i \\ z_i \end{bmatrix}. \tag{8.81}$$

Since the second order cone is convex and the tuple is an affine transformation of $\vec{x}$, it follows that $\{\vec{x} \in \mathbb{R}^n \mid \|A_i\vec{x} - \vec{y}_i\|_2 \leq \vec{b}_i^\top \vec{x} + z_i\}$ is a convex set. Thus the feasible set is convex (as the intersection of convex sets). The objective function is linear in $\vec{x}$, so the second-order cone problem is convex. □

Example 171. Consider the following problem:

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} \sum_{i=1}^{m} \|A_i\vec{x} - \vec{y}_i\|_2 . \tag{8.82}$$

One can formulate this as a second-order cone program by using slack variables:

$$\begin{aligned} p^\star = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ \vec{s}\in\mathbb{R}^m}} \quad & \sum_{i=1}^{m} s_i && (8.83) \\ \text{s.t.} \quad & \|A_i\vec{x} - \vec{y}_i\|_2 \leq s_i, \qquad \forall i \in \{1,\ldots,m\}. && (8.84) \end{aligned}$$

The following problem is also very related:

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} \max_{i\in\{1,\ldots,m\}} \|A_i\vec{x} - \vec{y}_i\|_2 . \tag{8.85}$$

We can use a similar slack variable reformulation to formulate this problem as a second order cone program.

$$\begin{aligned} p^\star = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ s\in\mathbb{R}}} \quad & s && (8.86) \\ \text{s.t.} \quad & \|A_i\vec{x} - \vec{y}_i\|_2 \leq s, \qquad \forall i \in \{1,\ldots,m\}. && (8.87) \end{aligned}$$

These problems can be formulated in terms of route planning – more specifically, finding the route which minimizes the total length between waypoints (in the first problem), or the route which minimizes the maximum length between waypoints (in the second problem).

Example 172 (LPs, Convex QPs, and Convex QCQPs as SOCPs)**.** One can see how LPs are QPs and how QPs are QCQPs, because in each transition the set of properties becomes more permissive — first a linear objective can become a quadratic objective, then linear constraints can become quadratic constraints. It is less clear how LPs, convex QPs, and convex QCQPs are SOCPs. In this example we derive a way to write convex QCQPs as SOCPs, which is also applicable to LPs and convex QPs (since LPs and convex QPs are convex QCQPs).

Consider a convex QCQP of the form

$$\begin{aligned} p^\star = \min_{\vec{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\vec{x}^\top H\vec{x} + \vec{c}^\top\vec{x} && (8.88) \\ \text{s.t.} \quad & \frac{1}{2}\vec{x}^\top P_i\vec{x} + \vec{b}_i^\top\vec{x} + c_i \leq 0, \qquad \forall i \in \{1,\ldots,m\} \\ & C\vec{x} = \vec{z}, \end{aligned}$$

where $H, P_1, \ldots, P_m \in \mathbb{S}^n_+$. We use the epigraph reformulation to obtain

$$\begin{aligned} p^\star = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ t\in\mathbb{R}}} \quad & t + \vec{c}^\top\vec{x} && (8.89) \\ \text{s.t.} \quad & \frac{1}{2}\vec{x}^\top P_i\vec{x} + \vec{b}_i^\top\vec{x} + c_i \leq 0, \qquad \forall i \in \{1,\ldots,m\} \\ & \frac{1}{2}\vec{x}^\top H\vec{x} \leq t \end{aligned}$$

$$C\vec{x} = \vec{z}.$$

Let us try to convert a general quadratic constraint

$$\frac{1}{2}\vec{x}^\top P_i \vec{x} + \vec{b}_i^\top \vec{x} + c_i \leq 0 \tag{8.90}$$

to a second-order cone constraint. Notice that this constraint is equivalent to

$$\vec{x}^\top P_i \vec{x} + 2(\vec{b}_i^\top \vec{x} + c_i) \leq 0. \tag{8.91}$$

In order to write each term as a square, we first write $\vec{x}^\top P_i \vec{x} = \left\| P_i^{1/2}\vec{x} \right\|_2^2$. We now use a difference of squares identity:

$$(u+v)^2 - (u-v)^2 = 4uv. \tag{8.92}$$

To apply this to the above formula, we plug in $u = \frac{1}{2}$ and $v = \vec{b}_i^\top \vec{x} + c_i$, to get

$$2(\vec{b}_i^\top \vec{x} + c_i) = \left(\frac{1}{2} + \vec{b}_i^\top \vec{x} + c_i\right)^2 - \left(\frac{1}{2} - \vec{b}_i^\top \vec{x} + c_i\right)^2. \tag{8.93}$$

This gives us the constraint

$$\left\| P_i^{1/2}\vec{x} \right\|_2^2 + \left(\frac{1}{2} + \vec{b}_i^\top \vec{x} + c_i\right)^2 - \left(\frac{1}{2} - \vec{b}_i^\top \vec{x} + c_i\right)^2 \leq 0, \tag{8.94}$$

which is equivalent to

$$\left\| P_i^{1/2}\vec{x} \right\|_2^2 + \left(\frac{1}{2} + \vec{b}_i^\top \vec{x} + c_i\right)^2 \leq \left(\frac{1}{2} - \vec{b}_i^\top \vec{x} + c_i\right)^2. \tag{8.95}$$

Taking square roots and writing things in terms of the ℓ^2-norm, we have

$$\left\| \begin{bmatrix} P_i^{1/2}\vec{x} \\ \frac{1}{2} + \vec{b}_i^\top \vec{x} + c_i \end{bmatrix} \right\|_2 \leq \frac{1}{2} - \vec{b}_i^\top \vec{x} + c_i. \tag{8.96}$$

Thus we can write the QCQP as

$$\begin{aligned} p^\star = \min_{\substack{\vec{x}\in\mathbb{R}^n \\ t\in\mathbb{R}}} \quad & t + \vec{c}^\top \vec{x} \qquad\qquad (8.97) \\ \text{s.t.} \quad & \left\| \begin{bmatrix} P_i^{1/2}\vec{x} \\ \frac{1}{2} + \vec{b}_i^\top \vec{x} + c_i \end{bmatrix} \right\|_2 \leq \frac{1}{2} - \vec{b}_i^\top \vec{x} + c_i, \qquad \forall i \in \{1, \ldots, m\} \\ & \left\| \begin{bmatrix} H^{1/2}\vec{x} \\ \frac{1}{2} - t \end{bmatrix} \right\|_2 \leq \frac{1}{2} - t \\ & \left\| C\vec{x} - \vec{z} \right\|_2 \leq 0. \end{aligned}$$

As a last note, non-convex QPs and non-convex QCQPs cannot be SOCPs because SOCPs are convex. Mechanically, the thing that goes wrong in the above proof is that we cannot take square roots of non-PSD matrices.

8.5 General Taxonomy

We conclude this chapter with a taxonomy of problems that we have discussed until now:

$$\texttt{LPs} \subset \texttt{Convex QPs} \subset \texttt{Convex QCQPs} \subset \texttt{SOCPs} \subset \texttt{Convex Problems} \tag{8.98}$$

All inclusions are strict, i.e., none of the classes is equivalent to any of the others. Also, general QPs are a super-set of QPs, but non-convex QPs are not SOCPs (and same with general QCQPs).

For extra optional reading, you may also look into *geometric programs* (GPs), which are nonconvex programs that can be turned into convex programs with a change of variables; *semidefinite programs* (SDPs), which are generally the broadest type of convex optimization problem used in practice; or mixed-integer programs (MIPs), which are useful in practice to incorporate integer constraints, but difficult to solve exactly. All such material is out of scope of the course.

Chapter 9

Regularization and Sparsity

Relevant sections of the textbooks:

- [2] Chapters 9, 12, 13.

9.1 Recapping Ridge Regression and Defining LASSO

The first example of regularization we saw was ridge regression. In this section, we'll review ridge regression. The most basic perspective of ridge regression focuses on the additional term we add to the objective function. In ridge regression, we solve the following problem:

$$\min_{\vec{x} \in \mathbb{R}^n} \left\{ \|A\vec{x} - \vec{y}\|_2^2 + \lambda \|\vec{x}\|_2^2 \right\}. \tag{9.1}$$

This is different from the OLS problem due to the additional $\lambda \|\vec{x}\|_2^2$ term, which can be thought of as a *regularizer* (i.e., a penalty) for having large $\vec{x}$ values. The λ parameter controls the strength of the penalty and is usually called a *regularization parameter*. In this sense, ridge regression is *regularized least squares*. More generally, we may define regularization as follows.

Definition 173 (Regularization)
Consider the optimization problem

$$p^\star = \min_{\vec{x} \in \Omega} f_0(\vec{x}). \tag{9.2}$$

For a given function $R\colon \Omega \to \mathbb{R}_+$ (the *regularizer*) and a *regularization parameter* $\lambda > 0$, the *regularized version* of the above problem is the problem

$$p_\lambda^\star = \min_{\vec{x} \in \Omega} \{f_0(\vec{x}) + \lambda R(\vec{x})\}. \tag{9.3}$$

Here λ controls the strength of the regularization.

In general, the original problem and the regularized problem *do not have the same solutions*, nor do versions of the regularized problem with different λ parameter. One need only consider ridge regression to keep this in mind; for a fixed A and $\vec{y}$, increasing λ will decrease the norm of the solution to the ridge regression problem, and sending it to 0 (i.e., recovering unregularized least squares) will increase the norm of the solution.

One example of regularization is the ℓ^2-norm penalty $R(\vec{x}) = \|\vec{x}\|_2^2$, which (when combined with $f_0(\vec{x}) = \|A\vec{x} - \vec{y}\|_2^2$) yields ridge regression. Another example is the elastic-net regression, which we covered briefly as an example when discussing convexity. But the main objective of this chapter is to look at the so-called LASSO regression problem, which uses an ℓ^1-norm regularizer. Recall that for a vector $\vec{x} \in \mathbb{R}^n$, its ℓ^1-norm is defined as $\|\vec{x}\|_1 = \sum_{i=1}^n |x_i|$.

Definition 174 (LASSO Regression)
Lett $A \in \mathbb{R}^{m\times n}$, $\vec{y} \in \mathbb{R}^m$, and $\lambda > 0$. The *LASSO regression* problem is:

$$\min_{\vec{x}\in\mathbb{R}^n} \left\{ \|A\vec{x} - \vec{y}\|_2^2 + \lambda \|\vec{x}\|_1 \right\}. \tag{9.4}$$

Here are some key properties of the LASSO regression problem.

Proposition 175
Consider the LASSO regression problem

$$\min_{\vec{x}\in\mathbb{R}^n} f_0(\vec{x}) \qquad \text{where} \qquad f_0(\vec{x}) \doteq \|A\vec{x} - \vec{y}\|_2^2 + \lambda \|\vec{x}\|_1 . \tag{9.5}$$

(a) The function $f_0 \colon \mathbb{R}^n \to \mathbb{R}$ is convex.

(b) If A has full column rank then f_0 is μ-strongly convex with $\mu = 2\sigma_n\{A\}^2$.

(c) A solution $\vec{x}^\star \in \operatorname{argmin}_{\vec{x}\in\mathbb{R}^n} f_0(\vec{x})$ always exists.

(d) If A has full column rank then the above solution is unique.

This picture is *very* different from ridge regression, where we are guaranteed that a solution always exists, is unique, and solvable in closed form. The question then becomes: why do we even care about the LASSO problem at all? The basic answer is that *it induces sparsity in the solution*, i.e., solutions to LASSO usually tend to have few nonzero entries. This sparsity is useful for applications in high-dimensional statistics and machine learning, as it reveals a certain structure — in words, it points out which "features" are the most relevant to the regression. In the following sections, we will observe how this sparsity emerges, both geometrically and algebraically.

9.2 Understanding the Difference Between the ℓ^2-Norm and the ℓ^1-Norm

In this section, we attempt to build more intuition about the difference between the ℓ^2-norm and the ℓ^1-norm. We do this by solving some problems which use the ℓ^2 norm, then replace it with the ℓ^1 norm and solve this new problem. Besides giving us intuition, it will help us learn how to analyze the LASSO problem.

Here is a diagram of the norm balls of the ℓ^1 (blue) and ℓ^2 (red) norms in $n = 1$ dimensions:

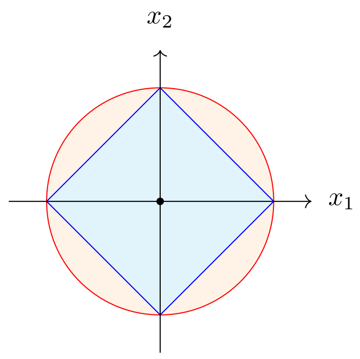


Figure 9.1: The ℓ^1 and ℓ^2 norm balls in $n = 2$ dimensions. Recall that the ℓ^p-norm ball is defined as the set of vectors $\vec{v}$ such that $\|\vec{v}\|_p \leq 1$.

The border of the norm balls are the points where each norm is equal to 1. Notice the difference in the geometry of these norm balls. The ℓ^2 norm ball is circular, while the ℓ^1 norm ball has distinctive corners.

In fact, these corners hint at a key difference between these norms: the ℓ^2 norm is differentiable everywhere, but $\|\vec{x}\|_1$ is *not differentiable* when *any* $x_i = 0$. These corners will help us understand how the ℓ^1 norm regularizer induces sparsity in the solution, and also inform our analysis of problems involving the ℓ^1-norm, including LASSO.

Example 176 (Least ℓ^1-Norm)**.** Recall that we solved the problem

$$\min_{\vec{x}\in\mathbb{R}^n} \quad \|\vec{x}\|_2^2 \tag{9.6}$$
$$\text{s.t.} \quad A\vec{x} = \vec{y}. \tag{9.7}$$

Using the KKT conditions, namely stationarity, we found an explicit solution to this problem: $\vec{x}^\star = A^\top(AA^\top)^{-1}\vec{y}$. Now let us replace the ℓ^2 norm with an ℓ^1 norm; we obtain the problem

$$\min_{\vec{x}\in\mathbb{R}^n} \quad \|\vec{x}\|_1 \tag{9.8}$$
$$\text{s.t.} \quad A\vec{x} = \vec{y}. \tag{9.9}$$

We cannot apply stationarity to this problem because the objective is non-differentiable. Thus, this problem seems intractable to solve by hand, at least for the moment. Instead, let us formulate it as a linear program. As before, we represent each x_i as the difference of non-negative numbers which sum to $|x_i|$. More formally, we introduce slack variables $\vec{x}^+, \vec{x}^- \in \mathbb{R}^n$ such that for each $i \in \{1, \ldots, n\}$ we have $x_i^+ \geq 0, x_i^- \geq 0, x_i^+ - x_i^- = x_i$, and $x_i^+ + x_i^- = |x_i|$. Thus we can rewrite the problem using the following linear program:

$$\min_{\vec{x}^+,\vec{x}^-\in\mathbb{R}^n} \quad \sum_{i=1}^n (x_i^+ + x_i^-) \tag{9.10}$$
$$\text{s.t.} \quad A(\vec{x}^+ - \vec{x}^-) = \vec{y} \tag{9.11}$$
$$\vec{x}^+ \geq \vec{0} \tag{9.12}$$
$$\vec{x}^- \geq \vec{0}. \tag{9.13}$$

This is a linear program which is efficiently solvable.

As a corollary, we can consider the ℓ^1-norm regression problem:

$$\min_{\vec{x}\in\mathbb{R}^n} \|A\vec{x} - \vec{y}\|_1 . \tag{9.14}$$

We can introduce the slack variable $\vec{e} = A\vec{x} - \vec{y}$ and obtain the problem:

$$\min_{\substack{\vec{x}\in\mathbb{R}^n \\ \vec{e}\in\mathbb{R}^m}} \quad \|\vec{e}\|_1 \tag{9.15}$$

$$\text{s.t.} \quad A\vec{x} - \vec{y} = \vec{e} \tag{9.16}$$

which is an equality-constrained ℓ^1 minimization problem, and thus a linear program as demonstrated above.

Example 177 (Mean Versus Median)**.** Let k be a positive integer. Suppose we have points $\vec{x}_1, \dots, \vec{x}_k \in \mathbb{R}^n$. Consider the problem

$$\min_{\vec{x}\in\mathbb{R}^n} \sum_{i=1}^{k} \|\vec{x} - \vec{x}_i\|_2^2 . \tag{9.17}$$

This is an unconstrained strongly convex differentiable problem, so it has a unique solution $\vec{x}_1^\star$ which we may find by setting the derivative of the objective to $\vec{0}$. We obtain

$$\vec{0} = 2\sum_{i=1}^{k}(\vec{x}_1^\star - \vec{x}_i) \tag{9.18}$$

$$\implies \vec{0} = \sum_{i=1}^{k}(\vec{x}_1^\star - \vec{x}_i) = k\cdot\vec{x}_1^\star - \sum_{i=1}^{k}\vec{x}_i \tag{9.19}$$

$$\implies \vec{x}_1^\star = \frac{1}{k}\sum_{i=1}^{k}\vec{x}_i. \tag{9.20}$$

This computation implies that the sample mean is the point which minimizes the total squared distance to all points in the dataset.

Now suppose that we instead consider the problem

$$\min_{\vec{x}\in\mathbb{R}^n} \sum_{i=1}^{k} \|\vec{x} - \vec{x}_i\|_2 . \tag{9.21}$$

The solution to this problem is the *sample median* of the points. To see this, suppose that $n = 1$, i.e., all our data x_i are scalar-valued. Then we obtain the problem

$$\min_{x\in\mathbb{R}} \sum_{i=1}^{k} |x - x_i| . \tag{9.22}$$

This is an unconstrained, convex, non-differentiable problem. Let us examine all critical points – that is, points where the derivative is 0 or undefined. The derivative of the objective is

$$\frac{\mathrm{d}}{\mathrm{d}x}\sum_{i=1}^{k}|x - x_i| = \sum_{i=1}^{k}\frac{\mathrm{d}}{\mathrm{d}x}|x - x_i| \tag{9.23}$$

$$= \sum_{i=1}^{k}\begin{cases} 1, & \text{if } x > x_i \\ -1, & \text{if } x < x_i \\ \text{undefined}, & \text{if } x = x_i \end{cases} \tag{9.24}$$

$$= \begin{cases} \sum_{i:\ x>x_i} 1 + \sum_{i:\ x<x_i} -1, & \text{if } x \notin \{x_1,\dots,x_k\} \\ \text{undefined}, & \text{if } x \in \{x_1,\dots,x_k\} \end{cases} \tag{9.25}$$

$$= \begin{cases} \left|\{i \in \{1, \dots, k\} \colon x > x_i\}\right| - \left|\{i \in \{1, \dots, k\} \colon x < x_i\}\right|, & \text{if } x \notin \{x_1, \dots, x_k\} \\ \text{undefined}, & \text{if } x \in \{x_1, \dots, x_k\}. \end{cases} \tag{9.26}$$

Thus if x is such that $|\{i \in \{1, \dots, k\} \colon x > x_i\}| = |\{i \in \{1, \dots, k\} \colon x < x_i\}|$, then the derivative is 0, so this x is a candidate solution. To put this convoluted-looking condition in words, notice that the first term in hte equality is just the number of x_i which are larger than x, and the second term is the number of x_i which are smaller than x. Thus the condition says that there are the same number of points in the set which are larger than x as there are points which are smaller than x. This x would fulfill the traditional definition of "median" as the middle of the sorted list of points.

To formally solve this problem, one must also check all the values $x = x_i$ and compare the objective values. But eventually after doing all this, one recovers that the optimal solutions are all possible medians of the dataset.

Because the median is defined using the $|\cdot|$ instead of $(\cdot)^2$ function, it inherits several different properties. The most striking is its robustness; the median is much more robust than the mean. The mean is very sensitive to outliers, while the median is less sensitive (i.e. if we blow up an outlier point, the mean will change a lot, while the median will be unaffected).

9.3 Analysis of LASSO Regression

In this section we will solve the one-dimensional LASSO problem. The ideas generalize to the vector case directly, through a reduction of the vector LASSO problems to several one-dimensional LASSO problems. The details of this reduction are left as a homework exercise.

First consider the scalar ridge regression problem, with $\vec{a} \neq \vec{0}$:

$$\min_{x \in \mathbb{R}} f_{\mathrm{RR}}(x) \qquad \text{where} \qquad f_{\mathrm{RR}}(x) \doteq \frac{1}{2}\left\|\vec{a}x - \vec{y}\right\|_2^2 + \frac{1}{2}\lambda x^2. \tag{9.27}$$

By taking the derivative, we get

$$\begin{aligned} \frac{\mathrm{d}f_{\mathrm{RR}}}{\mathrm{d}x}(x) &= \vec{a}^\top(\vec{a}x - \vec{y}) + \lambda x & (9.28)\\ &= (\vec{a}^\top\vec{a} + \lambda)x - \vec{a}^\top\vec{y} & (9.29)\\ &= (\|\vec{a}\|_2^2 + \lambda)x - \vec{a}^\top\vec{y} & (9.30)\\ \implies x_{\mathrm{RR}}^\star &= \frac{\vec{a}^\top\vec{y}}{\|\vec{a}\|_2^2 + \lambda}. & (9.31) \end{aligned}$$

By setting $\lambda = 0$ we obtain the least squares solution:

$$x_{\mathrm{LS}}^\star = \frac{\vec{a}^\top\vec{y}}{\|\vec{a}\|_2^2} \tag{9.32}$$

Now consider the scalar LASSO problem

$$\min_{x \in \mathbb{R}} f_{\mathrm{LASSO}}(x) \qquad \text{where} \qquad f_{\mathrm{LASSO}}(x) \doteq \frac{1}{2}\left\|\vec{a}x - \vec{y}\right\|_2^2 + \lambda\left|x\right|. \tag{9.33}$$

This has a derivative everywhere except $x = 0$. We obtain

$$\frac{\mathrm{d}f_{\mathrm{LASSO}}}{\mathrm{d}x}(x) = \vec{a}^\top(\vec{a}x - \vec{y}) + \lambda\begin{cases} 1, & \text{if } x > 0 \\ -1, & \text{if } x < 0 \\ \text{undefined}, & \text{if } x = 0. \end{cases} \tag{9.34}$$

Let $x^\star$ be a critical point of this problem. We solve what $x^\star$ should be using casework.

Case 1. If $x^\star > 0$, then the derivative is well-defined, so it must be equal to 0. Thus we have

$$\begin{aligned} 0 &= \frac{\mathrm{d}f_{\text{LASSO}}}{\mathrm{d}x}(x^\star) && (9.35)\\ &= \vec{a}^\top(\vec{a}x^\star - \vec{y}) + \lambda && (9.36)\\ \implies x^\star &= \frac{\vec{a}^\top\vec{y} - \lambda}{\|\vec{a}\|_2^2}. && (9.37) \end{aligned}$$

Thus if $x^\star > 0$ then $\vec{x}^\star = \frac{\vec{a}^\top\vec{y}-\lambda}{\|\vec{a}\|_2^2}$. Thus $\vec{x}^\star > 0$ if and only if $\vec{a}^\top\vec{y} > \lambda$.

Case 2. If $x^\star < 0$, then the derivative is well-defined, so it must be equal to 0. Thus we have

$$\begin{aligned} 0 &= \frac{\mathrm{d}f_{\text{LASSO}}}{\mathrm{d}x}(x^\star) && (9.38)\\ &= \vec{a}^\top(\vec{a}x^\star - \vec{y}) - \lambda && (9.39)\\ \implies x^\star &= \frac{\vec{a}^\top\vec{y} + \lambda}{\|\vec{a}\|_2^2}. && (9.40) \end{aligned}$$

Thus if $x^\star < 0$ then $\vec{x}^\star = \frac{\vec{a}^\top\vec{y}+\lambda}{\|\vec{a}\|_2^2}$. Thus $\vec{x}^\star < 0$ if and only if $\vec{a}^\top\vec{y} < -\lambda$.

Case 3. If $x^\star = 0$ then it is neither > 0 nor < 0. Thus we must have $-\lambda \leq \vec{a}^\top\vec{y} \leq \lambda$.

Thus we have three cases:

- $x^\star > 0 \iff \vec{a}^\top\vec{y} > \lambda$, in which case $x^\star = (\vec{a}^\top\vec{y} - \lambda)/\|\vec{a}\|_2^2$;
- $x^\star < 0 \iff \vec{a}^\top\vec{y} < -\lambda$, in which case $x^\star = (\vec{a}^\top\vec{y} + \lambda)/\|\vec{a}\|_2^2$; and
- $x^\star = 0 \iff -\lambda \leq \vec{a}^\top\vec{y} \leq \lambda$,

or in other words,

$$x^\star = \begin{cases} (\vec{a}^\top\vec{y} - \lambda)/\|\vec{a}\|_2^2, & \text{if } \vec{a}^\top\vec{y} > \lambda \\ (\vec{a}^\top\vec{y} + \lambda)/\|\vec{a}\|_2^2, & \text{if } \vec{a}^\top\vec{y} < -\lambda \\ 0, & \text{if } -\lambda \leq \vec{a}^\top\vec{y} \leq \lambda. \end{cases} \tag{9.41}$$

As a function of $\vec{a}^\top\vec{y}$, the solution $x^\star = x^\star_{\text{LASSO}}$ looks like:

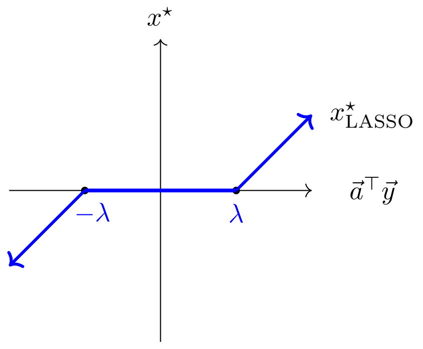


Figure 9.2: The plot of the function which maps $\vec{a}^\top\vec{y} \mapsto x^\star_{\text{LASSO}}$, where the latter term is the solution to our scalar LASSO problem. When $\vec{a}^\top\vec{y} \in [-\lambda, \lambda]$, we have $x^\star = 0$. The function $\vec{a}^\top\vec{y} \mapsto \vec{x}^\star$ is continuous, yet not differentiable at $\vec{a}^\top\vec{y} = \pm\lambda$.

If we plot the least squares solution in red on the same graph, it has the same nonzero slope as the LASSO solution, and looks like this:

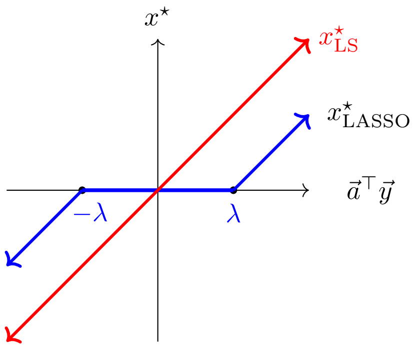


Figure 9.3: In red, we add the plot of the function which maps $\vec{a}^\top \vec{y} \mapsto x^\star_{\mathrm{LS}}$, where the latter term is the solution to to our scalar least squares problem. Note that the LASSO solution (in blue) is always closer to zero than the least squares solution, and is set directly to zero when $\vec{a}^\top \vec{y} \in [-\lambda, \lambda]$.

This illustrates a concept called *soft thresholding*: in the regime where the least squares solution $x^\star_{\mathrm{LS}}$ is already close to zero, $x^\star_{\mathrm{LASSO}}$ becomes exactly zero. Meanwhile, ridge regression does not do this: $x^\star_{\mathrm{RR}} = 0$ if and only if $\vec{a}^\top \vec{y} = 0$, which is exactly when the unregularized least squares solution itself is zero. This fundamental difference is why the solutions to LASSO regression tend to be sparse, i.e., have many entries set to 0.

9.4 Geometry of LASSO Regression

In this section we introduce a geometric description of LASSO. The geometry relies on a crucial theorem which unveils a deep connection between regularization and constrained optimization for convex problems. We state the result here; the proof uses duality theory and is left to homework.

Theorem 178

Let $f_0 \colon \mathbb{R}^n \to \mathbb{R}$ be strictly convex and such that $\lim_{\alpha \to \infty} f_0(\alpha \vec{x}) = \infty$ for all $\vec{x} \neq \vec{0}$,[a] and $R \colon \mathbb{R}^n \to \mathbb{R}_+$ be convex and take non-negative values. Further suppose that there exists $\vec{x}_0 \in \mathbb{R}^n$ such that $R(\vec{x}_0) = 0$.

For $\lambda \geq 0$ and $t \geq 0$, let $\mathcal{R}(\lambda)$ and $\mathcal{C}(t)$ be sets of solutions to the "regularized" and "constraint" programs:

$$\mathcal{R}(\lambda) \doteq \operatorname*{argmin}_{\vec{x} \in \mathbb{R}^n} \{f_0(\vec{x}) + \lambda R(\vec{x})\} \tag{9.42}$$

$$\mathcal{C}(t) \doteq \operatorname*{argmin}_{\substack{\vec{x} \in \mathbb{R}^n \\ R(\vec{x}) \leq t}} f_0(\vec{x}). \tag{9.43}$$

Then:

(a) for every $\lambda \geq 0$ there exists $t \geq 0$ such that $\mathcal{R}(\lambda) = \mathcal{C}(t)$; and

(b) for every $t > 0$ there exists $\lambda \geq 0$ such that $\mathcal{R}(\lambda) = \mathcal{C}(t)$.

[a]This assumption is called "coercivity".

This shows that in some sense, regularized convex problems are equivalent to constrained convex problems; and in this equivalence, the regularizer for the regularized problem shapes the constraint set of the constrained problem. In particular, regularized least squares ($f_0(\vec{x}) = \|A\vec{x} - \vec{y}\|_2^2$) with full column rank is equivalent to constrained least squares (with the same f_0).

Now, we sketch the feasible sets and level sets of the objective function for the constrained problems corresponding to both ridge regression and LASSO regression.

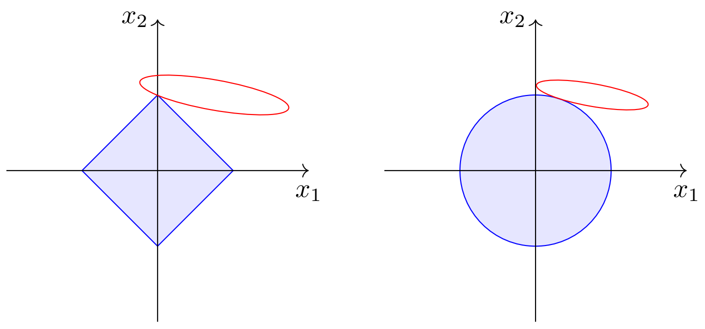


Figure 9.4: Geometric differences between LASSO and ridge regression. On the left side, the blue diamond depicts the feasible region for an ℓ^1-norm constraint such as $\|\vec{x}\|_1 \leq t$, while the circle on the right side is the feasible region for an ℓ^2-norm constraint such as $\|\vec{x}\|_2^2 \leq t$. On both graphs, the red line is a level set of our objective function; specifically, the minimal level set that still intersects the feasible region. The intersection of this level set with the feasible region is the solution to our constrained problem and thus to an equivalent regularized problem.

Note how with the ℓ^1-norm constraint, the intersection of the feasible region with the minimal level set is more likely to be at a corner of the feasible region, which is a point where some coordinates are set exactly to zero. Meanwhile, with the ℓ^2-norm constraint, the intersection can be at an arbitrary point on the circle (or sphere in higher dimensions), and likely isn't at a corner. This is why LASSO induces sparsity in $\vec{x}$, due to the distinctive corners we saw earlier in its norm ball. Meanwhile, although ridge regression compresses $\vec{x}^{\star}_{\mathrm{RR}}$ to be smaller, it doesn't necessarily induce sparsity in $\vec{x}^{\star}_{\mathrm{RR}}$.

Chapter 10

Advanced Descent Methods

Relevant sections of the textbooks:

- [1] Chapter 11.
- [2] Section 12.5.

10.1 Coordinate Descent

Recall from Chapter 6 the general idea of descent-based methods for solving optimization problems. These methods are based on the process of starting with some initial guess $\vec{x}^{(0)} \in \mathbb{R}^n$, then generating a sequence of refined guesses $\vec{x}^{(1)}, \vec{x}^{(2)}, \vec{x}^{(3)}, \ldots$ using the general update rule

$$\vec{x}^{(t+1)} = \vec{x}^{(t)} + \eta \vec{v}^{(t)} \tag{10.1}$$

for some search direction $\vec{v}^{(t)}$ and step size η. In Chapter 6 we covered the gradient descent method, which uses the gradient of the function as the search direction. In this chapter we will revisit descent-based optimization methods and introduce alternative update rules.

In this section we will introduce *coordinate descent*, a class of descent-based algorithms that finds a minimizer of multivariate functions by iteratively minimizing it along one direction at a time. Consider the unconstrained convex optimization problem

$$p^\star = \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}), \tag{10.2}$$

with the optimization variable

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{10.3}$$

Before we introduce the algorithm, we introduce some notation. For indices i and j, we introduce the notation $\vec{x}_{i:j} = (x_i, x_{i+1}, \ldots, x_{j-1}, x_j) \in \mathbb{R}^{j-i+1}$ to be the entries of $\vec{x}$ between indices i and j (inclusive on both ends).

Given an initial guess $\vec{x}^{(0)}$, for $t \geq 0$ the coordinate descent algorithm updates the iterate $\vec{x}^{(t)}$ by *sequentially* minimizing the function $f(\vec{x})$ with respect to each coordinate, namely using the update rule

$$x_i^{(t+1)} \in \operatorname*{argmin}_{x_i \in \mathbb{R}} f(\vec{x}_{1:i-1}^{(t+1)}, x_i, \vec{x}_{i+1:n}^{(t)}). \tag{10.4}$$

Namely, we perform the update

$$x_1^{(t+1)} \in \operatorname*{argmin}_{x_1 \in \mathbb{R}} f(x_1, \vec{x}_{2:n}^{(t)}) \tag{10.5}$$

$$x_2^{(t+1)} \in \operatorname*{argmin}_{x_2 \in \mathbb{R}} f(x_1^{(t+1)}, x_2, \vec{x}_{3:n}^{(t)}) \tag{10.6}$$

$$\vdots \qquad \vdots \tag{10.7}$$

$$x_n^{(t+1)} \in \operatorname*{argmin}_{x_n \in \mathbb{R}} f(\vec{x}_{1:n-1}^{(t+1)}, x_n). \tag{10.8}$$

This is a sequential process since after finding the minimizer along the i^{th} coordinate (i.e. $x_i^{(t+1)}$) we use its values for minimizing subsequent coordinates. Also we note that the order of the coordinates is arbitrary. We formalize this update in the following algorithm.

Algorithm 6 CoordinateDescent

```
1: function CoordinateDescent($f, \vec{x}^{(0)}, T$)
2:     for $t = 0, 1, \ldots, T_1$ do
3:         for $i = 1, \ldots, N$ do
4:             $x_i^{(t+1)} \leftarrow \operatorname{argmin}_{x_i \in \mathbb{R}} f(x_{1:i-1}^{(t+1)}, x_i, x_{i+1:n}^{(t)})$.
5:         end for
6:     end for
7:     return $\vec{x}^T$
8: end function
```

The algorithm breaks down the difficult multivariate optimization problem into a sequence of simpler univariate optimization problems.

We first want to discuss the issue of well-posedness of the algorithm. We know that any of the argmins used may not exist, in which case the algorithm is not well-defined, and so we cannot even think about its behavior or convergence. Nevertheless, in a large class of problems which have many different characterizations, the argmins are well-defined. We say in this case that the coordinate descent algorithm is well-posed.

We now want to address the question of convergence. It is not obvious that minimizing the function $f(\vec{x})$ can be achieved by minimizing along each direction separately. In fact, the algorithm is not guaranteed to converge to an optimal solution for general convex functions. However, under some additional assumptions on the function, we can guarantee convergence. To build an intuition for what additional assumptions are needed we consider the following question. Let $f(\vec{x})$ be a convex differentiable function. Suppose that $x_i^\star \in \operatorname{argmin}_{x_i \in \mathbb{R}} f(x_{1:i-1}^\star, x_i, x_{i+1:n}^\star)$ for all i. Can we conclude that $\vec{x}^\star$ is a global minimizer of $f(\vec{x})$? The answer to this question is yes. We can prove this by recalling the first order optimality condition for unconstrained convex functions and the definition of partial derivatives. If $\vec{x}^\star$ is a minimizer of $f(\vec{x})$ along the direction $\vec{e}_i$ then we have

$$\frac{\partial f}{\partial x_i}(\vec{x}^\star) = 0. \tag{10.9}$$

If this is true for all i then $\nabla f(\vec{x}^\star) = \vec{0}$, implying that $\vec{x}^\star$ is a global minimizer for f. This discussion forms a proof of the following theorem, which is Theorem 12.4 in [2].

Theorem 179 (Convergence of Coordinate Descent for Differentiable Convex Functions)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function which is *separately strictly convex* in each argument. That is, suppose that for each i, and each fixed $\vec{x}_{1:i-1}$ and $\vec{x}_{i+1:n}$, the function $x_i \mapsto f(\vec{x}_{1:i-1}, x_i, \vec{x}_{i+1:n})$ is strictly convex. If the coordinate descent algorithm is well-posed, and the unconstrained minimization problem

$$\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \tag{10.10}$$

has a solution, then the sequence of iterates $\vec{x}^{(0)}, \vec{x}^{(1)}, \ldots$ generated by the coordinate descent algorithm converges to an optimal solution to (10.10).

The coordinate descent algorithm may not converge to an optimal solution for general non-differentiable functions, even if they are convex. However, we can still prove that coordinate descent converges for a special class of functions of the form

$$f(\vec{x}) = g(\vec{x}) + \sum_{i=1}^{n} h_i(x_i) \tag{10.11}$$

where $g\colon \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and each $h_i\colon \mathbb{R} \to \mathbb{R}$ is convex (but not necessarily differentiable). This form includes various ℓ^1 regularization problems (such as LASSO regression) which have a separable non-differentiable component. The provable convergence of coordinate descent algorithm makes it an attractive choice for this class of problems.

Example 180. In this example we will consider the LASSO regression problem and examine how coordinate descent algorithm can be applied to solve it. Note that the LASSO objective follows the form described in (10.11). For $A \in \mathbb{R}^{m \times n}$ which has columns $\vec{a}_1, \ldots, \vec{a}_n \in \mathbb{R}^m$, and $\vec{y} \in \mathbb{R}^m$, we consider the LASSO objective

$$f(\vec{x}) = \frac{1}{2} \left\| A\vec{x} - \vec{y} \right\|_2^2 + \lambda \left\| x \right\|_1 . \tag{10.12}$$

We aim to use coordinate descent to minimize this function. Let $\vec{x}^{(0)}$ be the initial guess. Then we perform the coordinate descent update by solving the following optimization problem:

$$x_i^{(t+1)} = \operatorname*{argmin}_{x_i \in \mathbb{R}} f(\vec{x}^{(t+1)}_{1:i-1}, x_i, \vec{x}^{(t)}_{i+1:n}). \tag{10.13}$$

Each of these optimization problems will be solved similarly to what we did in Section 9.3. For notational clarity, let us instead solve the more generic problem

$$x_i^{\star} \mathrel{\dot{\in}} \operatorname*{argmin}_{x_i \in \mathbb{R}} f(\vec{x}_{1:i-1}, x_i, \vec{x}_{i+1:n}). \tag{10.14}$$

Then we have that $\frac{\partial f}{\partial x_i}(\vec{x}_{1:i-1}, x_i^{\star}, \vec{x}_{i+1:n})$ is either 0 or undefined. Thus we compute each partial derivative of f. Indeed, we have

$$\begin{aligned}
\frac{\partial f}{\partial x_i}(\vec{x}) &= \frac{1}{2} \frac{\partial}{\partial x_i} \left\| A\vec{x} - \vec{y} \right\|_2^2 + \lambda \frac{\partial}{\partial x_i} \left\| \vec{x} \right\|_1 && (10.15) \\
&= \vec{e}_i^{\top} \nabla \left(\frac{1}{2} \left\| A\vec{x} - \vec{y} \right\|_2^2 \right) + \lambda \frac{\partial}{\partial x_i} \sum_{j=1}^{n} |x_j| && (10.16) \\
&= \vec{e}_i^{\top} A^{\top} (A\vec{x} - \vec{y}) + \lambda \frac{\partial}{\partial x_i} |x_i| && (10.17) \\
&= \vec{a}_i^{\top} (A\vec{x} - \vec{y}) + \lambda \begin{cases} \operatorname{sgn}(x_i), & \text{if } x_i \neq 0 \\ \text{undefined}, & \text{if } x_i = 0. \end{cases} && (10.18)
\end{aligned}$$

We now introduce the additional notation $A_{i:j} \in \mathbb{R}^{m\times(j-i+1)}$ as the sub-matrix of A whose columns are the i^{th} through j^{th} columns of A (inclusive). Using this notation, we can simplify the first term as

$$\vec{a}_i^\top(A\vec{x}-\vec{y}) = \vec{a}_i^\top A\vec{x} - \vec{a}_i^\top \vec{y} \tag{10.19}$$

$$= \vec{a}_i^\top \left(\sum_{j=1}^{n} \vec{a}_j x_j - \vec{y}\right) \tag{10.20}$$

$$= \|\vec{a}_i\|_2^2 x_i + \vec{a}_i^\top \left(\sum_{\substack{j=1\\ j\neq i}}^{n} \vec{a}_j x_j - \vec{y}\right) \tag{10.21}$$

$$= \|\vec{a}_i\|_2^2 + \vec{a}_i^\top (A_{1:i-1}\vec{x}_{1:i-1} + A_{i+1:n}\vec{x}_{i+1:n} - \vec{y}) \tag{10.22}$$

Now there are two cases depending on the value of $x_i^\star$.

Case 1. Suppose $x_i^\star \neq 0$. Then we have by the optimality condition that

$$0 = \frac{\partial f}{\partial x_i}(\vec{x}_{1:i-1}, x_i^\star, \vec{x}_{i+1:n}) \tag{10.23}$$

$$= \|\vec{a}_i\|_2^2 x_i^\star + \vec{a}_i^\top (A_{1:i-1}\vec{x}_{1:i-1} + A_{i+1:n}\vec{x}_{i+1:n} - \vec{y}) + \lambda \operatorname{sgn}(x_i^\star) \tag{10.24}$$

$$\implies x_i^\star = \frac{1}{\|\vec{a}_i\|_2^2}\left(\vec{a}_i^\top [\vec{y} - A_{1:i-1}\vec{x}_{1:i-1} - A_{i+1:n}\vec{x}_{i+1:n}] - \lambda \operatorname{sgn}(x_i^\star)\right). \tag{10.25}$$

Thus in particular we have

$$x_i^\star > 0 \iff \vec{a}_i^\top (\vec{y} - A_{1:i-1}\vec{x}_{1:i-1} - A_{i+1:n}\vec{x}_{i+1:n}) > \lambda, \tag{10.26}$$

and the corresponding relation

$$x_i^\star < 0 \iff \vec{a}_i^\top (\vec{y} - A_{1:i-1}\vec{x}_{1:i-1} - A_{i+1:n}\vec{x}_{i+1:n}) < -\lambda. \tag{10.27}$$

Case 2. By contrapositive, we have

$$-\lambda \leq \vec{a}_i^\top (\vec{y} - A_{1:i-1}\vec{x}_{1:i-1} - A_{i+1:n}\vec{x}_{i+1:n}) \leq \lambda \iff x_i^\star = 0. \tag{10.28}$$

Therefore we have derived a closed-form coordinate descent update:

$$\vec{a}_i^\top \left(\vec{y} - A_{1:i-1}\vec{x}_{1:i-1}^{(t+1)} - A_{i+1:n}\vec{x}_{i+1:n}^{(t)}\right) > \lambda \implies x_i^{(t+1)} = \frac{1}{\|\vec{a}_i\|_2^2}\left(\vec{a}_i^\top \left[\vec{y} - A_{1:i-1}\vec{x}_{1:i-1}^{(t+1)} - A_{i+1:n}\vec{x}_{i+1:n}^{(t)}\right] - \lambda\right) \tag{10.29}$$

$$\vec{a}_i^\top \left(\vec{y} - A_{1:i-1}\vec{x}_{1:i-1}^{(t+1)} - A_{i+1:n}\vec{x}_{i+1:n}^{(t)}\right) < -\lambda \implies x_i^{(t+1)} = \frac{1}{\|\vec{a}_i\|_2^2}\left(\vec{a}_i^\top \left[\vec{y} - A_{1:i-1}\vec{x}_{1:i-1}^{(t+1)} - A_{i+1:n}\vec{x}_{i+1:n}^{(t)}\right] + \lambda\right) \tag{10.30}$$

$$\left|\vec{a}_i^\top \left(\vec{y} - A_{1:i-1}\vec{x}_{1:i-1}^{(t+1)} - A_{i+1:n}\vec{x}_{i+1:n}^{(t)}\right)\right| \leq \lambda \implies x_i^{(t+1)} = 0. \tag{10.31}$$

10.2 Newton's Method

Let us consider the unconstrained optimization problem

$$p^\star = \min_{\vec{x}\in\mathbb{R}^n} f(\vec{x}). \tag{10.32}$$

Recall that in the gradient descent algorithm, we assumed that the objective function $f(\vec{x})$ is differentiable. Furthermore, we assumed that at every point $\vec{x} \in \mathbb{R}^n$ we can compute $f(\vec{x})$ as well as $\nabla f(\vec{x})$. Here, we make the additional assumption that $f(\vec{x})$ is twice differentiable and that we can compute the Hessian $\nabla^2 f(\vec{x})$. We wish to use the Hessian to choose a good search direction and accelerate convergence. Optimization algorithms that utilize second derivatives (e.g. the Hessian) are called *second-order methods*.

One of the most famous second-order methods is *Newton's method*. Newton's method is based on the following idea for minimizing strictly-convex functions with positive definite Hessians: first, start with an initial guess $\vec{x}^{(0)}$. Then in each iteration $t = 1, 2, 3, \ldots$, approximate the objective function with its second-order Taylor approximation around the point $\vec{x}^{(t)}$. The minimizer of this quadratic approximation is then chosen as the next iterate $\vec{x}^{(t+1)}$.

More formally, let us assume that f is strictly convex and twice-differentiable with positive definite Hessian at $\vec{x}^{(t)}$, and let us write the second-order Taylor approximation of the function $f(\vec{x})$ around the point $\vec{x}^{(t)}$. We obtain

$$\widehat{f}_2(\vec{x}; \vec{x}^{(t)}) = f(\vec{x}^{(t)}) + [\nabla f(\vec{x}^{(t)})]^\top (\vec{x} - \vec{x}^{(t)}) + \frac{1}{2}(\vec{x} - \vec{x}^{(t)})^\top [\nabla^2 f(\vec{x}^{(t)})](\vec{x} - \vec{x}^{(t)}). \tag{10.33}$$

Since the Hessian $\nabla^2 f(\vec{x}^{(t)}) \succ 0$, we can solve the problem

$$\min_{\vec{x} \in \mathbb{R}^n} \widehat{f}_2(\vec{x}; \vec{x}^{(t)}), \tag{10.34}$$

which is a convex quadratic program, using our (by now) standard techniques. Setting the gradient (in $\vec{x}$) to $\vec{0}$, we obtain

$$\begin{aligned} \vec{0} &= \nabla \widehat{f}_2(\vec{x}^\star; \vec{x}^{(t)}) && (10.35)\\ &= \nabla f(\vec{x}^{(t)}) + [\nabla^2 f(\vec{x}^{(t)})](\vec{x}^\star - \vec{x}^{(t)}) && (10.36)\\ \implies \vec{x}^\star &= \vec{x}^{(t)} - [\nabla^2 f(\vec{x}^{(t)})]^{-1}[\nabla f(\vec{x}^{(t)})]. && (10.37) \end{aligned}$$

This gives the Newton's method update rule:

$$\vec{x}^{(t+1)} = \vec{x}^{(t)} - [\nabla^2 f(\vec{x}^{(t)})]^{-1}[\nabla f(\vec{x}^{(t)})]. \tag{10.38}$$

The formal algorithm, detailed in Algorithm 7, just repeats this iteration.

Algorithm 7 Newton's Method

```
1: function NewtonMethod($f, \vec{x}^{(0)}, T$)
2:     for $t = 0, 1, \ldots, T-1$ do
3:         $\vec{x}^{(t+1)} \leftarrow \vec{x}^{(t)} - \left[\nabla^2 f(\vec{x}^{(t)})\right]^{-1} [\nabla f(\vec{x}^{(t)})]$
4:     end for
5:     return $\vec{x}^{(T)}$
6: end function
```

We call the vector $\left[\nabla^2 f(\vec{x}^{(t)})\right]^{-1} [\nabla f(\vec{x}^{(t)})]$ the Newton direction. Here, we do not choose a step-size η. Instead, we take a *full* step in the Newton direction towards the minimizer of the quadratic approximation of the objective function. This is the basic version of Newton's method; it is *not* guaranteed to converge in general. To achieve convergence, we can introduce a step-size $\eta > 0$ to the Newton update, obtaining the so-called *damped* Newton's method, which has the iteration

$$\vec{x}^{(t+1)} = \vec{x}^{(t)} - \eta \left[\nabla^2 f(\vec{x}^{(t)})\right]^{-1} [\nabla f(\vec{x}^{(t)})]. \tag{10.39}$$

A full algorithm is very similar to Algorithm 7 and is provided in Algorithm 8.

Algorithm 8 Damped Newton's Method

```
1: function DampedNewtonMethod($f, \vec{x}^{(0)}, \eta, T$)
2:     for $t = 0, 1, \ldots, T-1$ do
3:         $\vec{x}^{(t+1)} \leftarrow \vec{x}^{(t)} - \eta \left[\nabla^2 f(\vec{x}^{(t)})\right]^{-1} [\nabla f(\vec{x}^{(t)})]$
4:     end for
5:     return $\vec{x}^{(T)}$
6: end function
```

We will not discuss convergence proofs of Newton's method in this course; you may use [7] for further reading.

All of our discussion thus far has been under the assumption $\nabla^2 f(\vec{x}^{(t)}) \succ 0$. If the Hessian is not positive definite, one may adapt the algorithm accordingly, forming a new class of methods called *quasi-Newton's methods*. Discussion of quasi-Newton's methods are out of scope of the course.

Finally, we discuss the algorithmic complexity of Newton's method. In every iteration, we need to compute and invert the Hessian $\nabla^2 f(\vec{x}^{(t)})$ to obtain the search direction. This is much more expensive than computing the gradient $\nabla f(\vec{x}^{(t)})$, which is used both in the gradient descent method and in Newton's method. However, this expensive step is not without justification; in many convex optimization problems, Newton's method can be shown to converge to the optimal solution much faster (i.e., in fewer iterations) than gradient descent.

10.3 Newton's Method with Linear Equality Constraints

Our derivation of Newton's method can be used to handle equality-constrained optimization problems. Let $A \in \mathbb{R}^{m \times n}$ and $\vec{y} \in \mathbb{R}^m$, and let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable strictly convex function with positive definite Hessian. Consider the following convex optimization problem:

$$p^\star = \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}). \tag{10.40}$$

$$\text{s.t.} \quad A\vec{x} = \vec{y}. \tag{10.41}$$

We will use the same approach as with the unconstrained Newton's method, that is, we will take the second-order Taylor approximation around $\vec{x}^{(t)}$ and minimize it over the constraint set $\Omega \doteq \{\vec{x} \in \mathbb{R}^n : A\vec{x} = \vec{y}\}$. This method gives the following constrained convex quadratic program:

$$\min_{\vec{x} \in \mathbb{R}^n} \quad f(\vec{x}^{(t)}) + [\nabla f(\vec{x}^{(t)})]^\top (\vec{x} - \vec{x}^{(t)}) + \frac{1}{2}(\vec{x} - \vec{x}^{(t)})^\top [\nabla^2 f(\vec{x}^{(t)})](\vec{x} - \vec{x}^{(t)}) \tag{10.42}$$

$$\text{s.t.} \quad A\vec{x} = \vec{b}. \tag{10.43}$$

Note that the quadratic program is convex and, if the original problem is feasible (i.e., Ω is nonempty) that strong duality holds by Slater's condition. Thus, we can solve this QP by solving the KKT conditions, as they are necessary and sufficient for global optimality. We begin by writing the Lagrangian $L\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ associated with this quadratic program, which is defined as

$$L(\vec{x}, \vec{\nu}) = f(\vec{x}^{(t)}) + [\nabla f(\vec{x}^{(t)})]^\top (\vec{x} - \vec{x}^{(t)}) + \frac{1}{2}(\vec{x} - \vec{x}^{(t)})^\top [\nabla^2 f(\vec{x}^{(t)})](\vec{x} - \vec{x}^{(t)}) + \vec{\nu}^\top (A\vec{x} - \vec{b}). \tag{10.44}$$

Suppose that $(\vec{x}^\star, \vec{\nu}^\star)$ are globally optimal for the constrained quadratic program. Then they must satisfy the KKT conditions, which are:

- Primal feasibility:

$$A\vec{x}^{\star} = \vec{y}. \tag{10.45}$$

- Stationarity/first-order condition:

$$\begin{aligned}\vec{0} &= \nabla_{\vec{x}} L(\vec{x}^{\star}, \vec{\nu}^{\star}) && (10.46)\\ &= \nabla f(\vec{x}^{(t)}) + [\nabla^2 f(\vec{x}^{(t)})](\vec{x}^{\star} - \vec{x}^{(t)}) + A^{\top}\vec{\nu}. && (10.47)\end{aligned}$$

Let us define a vector $\vec{v}^{(t)} = \vec{x}^{\star} - \vec{x}^{(t)}$. Since $\vec{x}^{(t)}$ is feasible, we have $A\vec{x}^{(t)} = \vec{y}$. Thus we have

$$\begin{aligned}A\vec{v}^{(t)} &= A(\vec{x}^{\star} - \vec{x}^{(t)}) && (10.48)\\ &= A\vec{x}^{\star} - A\vec{x}^{(t)} && (10.49)\\ &= \vec{y} - \vec{y} && (10.50)\\ &= \vec{0}. && (10.51)\end{aligned}$$

Thus, if we write the system in terms of $\vec{v}^{(t)}$ instead of $\vec{x}^{\star}$, we have the system of equations

$$\begin{aligned}\vec{0} &= \nabla f(\vec{x}^{(t)}) + [\nabla^2 f(\vec{x}^{(t)})]\vec{v}^{(t)} + A^{\top}\vec{\nu}^{\star} && (10.52)\\ \vec{0} &= A\vec{v}^{(t)} && (10.53)\end{aligned}$$

which can be expressed in matrix form as

$$\begin{bmatrix}\nabla^2 f(\vec{x}^{(t)}) & A^{\top}\\ A & 0\end{bmatrix}\begin{bmatrix}\vec{v}^{(t)}\\ \vec{\nu}\end{bmatrix} = \begin{bmatrix}\nabla f(\vec{x}^{(t)})\\ 0\end{bmatrix}. \tag{10.54}$$

After solving this system of equations for $\vec{v}^{(t)}$, our update rule becomes

$$\vec{x}^{(t+1)} = \vec{x}^{\star} = \vec{x}^{(t)} + \vec{v}^{(t)}, \tag{10.55}$$

Which is equivalent to setting the new iterate as the minimizer of the constrained QP. The formal iteration is given in Algorithm 9.

Algorithm 9 Newton's Method with Linear Equality Constraints

```
1: function EqualityConstrainedNewtonMethod($f, \vec{x}^{(0)}, T, A, \vec{y}$)
2:     for $t = 0, 1, \ldots, T-1$ do
3:         Solve the system $\begin{bmatrix}\nabla^2 f(\vec{x}^{(t)}) & A^{\top}\\ A & 0\end{bmatrix}\begin{bmatrix}\vec{v}^{(t)}\\ \vec{\nu}\end{bmatrix} = \begin{bmatrix}\nabla f(\vec{x}^{(t)})\\ 0\end{bmatrix}$ for $\vec{v}^{(t)}$
4:         $\vec{x}^{(t+1)} \leftarrow \vec{x}^{(t)} + \vec{v}^{(t)}$
5:     end for
6:     return $\vec{x}^{(T)}$
7: end function
```

There also exist damped versions of this algorithm, but their analysis is out of scope of the course.

10.4 (OPTIONAL) Interior Point Method

In the previous section we introduced Newton's method, which allows us to solve convex optimization problems with simple linear equality constraints. In this section, we will build on top of Newton's method, introducing a new class of algorithms to handle convex optimization problems with inequality constraints. Precisely, we will introduce the *interior point method*, which allows us to solve convex optimization problems of the following form

$$p^{\star} = \min_{\vec{x} \in \mathbb{R}^n} \quad f_0(\vec{x}) \tag{10.56}$$

$$\text{s.t.} \quad f_i(\vec{x}) \leq 0, \qquad \forall i = 1, 2, \ldots, m \tag{10.57}$$

$$A\vec{x} = \vec{y}, \tag{10.58}$$

where f_0, f_1, …, f_m are all convex twice-differentiable functions. Interior point methods (IPM) are a class of algorithms which solves the problem (10.56) by solving a sequence of convex optimization problems with linear constraints using Newton's method. The key idea used in IPM is the *barrier function*, which we introduce next.

10.4.1 Barrier Functions

Consider the problem (10.56). Our goal is to eliminate the inequality constraints and convert the problem to an equality constrained problem to which Newton's method can be applied. One way to do so is to augment the inequality constraints to the objective using indicator functions, as we did when developing our theory of duality. More precisely, consider the following unconstrained problem, which we denote by $\mathcal{P}$:

$$\text{problem } \mathcal{P}: \qquad p^{\star} = \min_{\vec{x} \in \mathbb{R}^n} \quad f_0(\vec{x}) + \sum_{i=1}^{m} I(f_i(\vec{x})) \tag{10.59}$$

$$A\vec{x} = \vec{b}. \tag{10.60}$$

where $I \colon \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is given by

$$I(z) = \mathbb{1}[z \leq 0] = \begin{cases} 0, & \text{if } z \leq 0 \\ +\infty, & \text{if } z > 0 \end{cases} \tag{10.61}$$

This gives us an optimization problem with only linear equality constraints that is *equivalent* to the original optimization problem (10.56) (i.e., they have the same solution). However, introducing the indicator function now makes the objective function non-differentiable so we can no longer apply Newton's method to solve this problem. To overcome this problem, we will instead approximate the indicator function with a differentiable function ϕ, which we call a *barrier function*.

There are several choices for ϕ, i.e., good approximations for the indicator function, but they must all have something in common. Namely, ϕ should be a convex increasing function on $\mathbb{R}_{--}$, such that $\lim_{z \nearrow 0} \phi(z) = +\infty$, just like the indicator function I. There are many candidate functions that satisfy these criteria. One of the most used barrier functions that we will introduce here is the logarithmic barrier function, which, for some $\alpha > 0$, takes the form

$$\phi_\alpha(z) = -\frac{1}{\alpha} \log(-z), \tag{10.62}$$

The parameter α controls the accuracy of the approximation — as α grows larger, the logarithmic barrier function becomes a better and better approximation to the indicator function.

Using this logarithmic barrier, we can define an approximate optimization problem $\widehat{\mathcal{P}}(\alpha)$ to $\mathcal{P}$ by the following:

$$\text{problem } \widehat{\mathcal{P}}(\alpha): \qquad \widehat{p}^{\star}_{\alpha} = \min_{\vec{x} \in \mathbb{R}^n} \quad f_0(\vec{x}) + \sum_{i=1}^{m} \phi_\alpha(f_i(\vec{x})) \tag{10.63}$$

$$A\vec{x} = \vec{y}. \tag{10.64}$$

This optimization problem has a convex twice-differentiable objective function and linear equality constraints, so Newton's method can be applied.

10.4.2 Barrier Method

The problem $\widehat{\mathcal{P}}(\alpha)$ is just an approximation of the original problem $\mathcal{P}$. In particular, they do not necessarily have the same solution. Thus, the natural question to ask is how close the solutions of $\mathcal{P}$ and $\widehat{\mathcal{P}}(\alpha)$ are. The choice of the parameter α will be crucial, since small α mean that ϕ_α does not behave like an indicator function and so $\widehat{\mathcal{P}}(\alpha)$ and $\mathcal{P}$ are totally different in general, whereas large α offer much better approximations and thus allows us to expect much better solutions. Indeed, it is possible to show, with some additional assumptions [7], some more complicated bounds relating the distance of the solutions to $\mathcal{P}$ to the solutions to $\widehat{\mathcal{P}}(\alpha)$, as a function of α. We do not go into this analysis here, but its main takeaway is that large values of α give us good approximate solutions.

One natural question is: why don't we just take the largest possible α and get a near-perfect approximation to the indicator functions, and thus the original program $\mathcal{P}$? This actually may not be optimal from an algorithmic standpoint, as as solving problem $\widehat{\mathcal{P}}(\alpha)$ using Newton's method becomes difficult for large values of α. This is because Newton's method relies on computing and inverting the Hessian of the objective, and with large values of α the Hessian changes rapidly for points near the boundary of the feasible set for the original problem, even points which may be the solution.

The barrier method overcomes this problem by solving the approximate problem for a relatively small value of α and obtain the approximate solution $\vec{x}^\star(\alpha)$. The algorithm then refines this approximate solution by using it as an initial guess to solve the approximate problem with a larger value of α. This way, when the algorithm attempts to solve the difficult approximate problems $\widehat{\mathcal{P}}(\alpha)$ (as the value of α increases), it does so with a good initial guess. Newton's method converges extremely fast near the optimal solution, so this procedure greatly improves the convergence of Newton's method even when α is very large.[1]

When solving constrained optimization problems, it is necessary for most algorithms to start with a feasible initialization $\vec{x}^{(0)}$. This is particularly important for the barrier method; in fact, it is necessary to start at a strictly feasible initial guess, so that the value of the approximate problem is finite and the derivative of the objective function is defined. In Algorithm 10 we present one instance of the barrier method in which the value of α is updated by scaling it with some constant $\mu > 1$.

Algorithm 10 Barrier Method.

```
1: function BarrierMethod($f_0, f_1, \dots, f_m, \vec{x}^{(0)}, \alpha^{(0)}, A, \vec{y}, \mu, T$)
2:     for $t = 1, \dots, T$ do
3:         $\vec{x}^{(t)} \leftarrow$ Solution of $\widehat{\mathcal{P}}(\alpha^{(t-1)})$ using Newton's method (Algorithm 9) starting at $\vec{x}^{(t-1)}$
4:         $\alpha^{(t)} \leftarrow \mu\alpha^{(t-1)}$
5:     end for
6:     return $\vec{x}^{(T)}$
7: end function
```

[1]This easy-to-hard solution process is one instance of a more general algorithmic paradigm called *homotopy continuation* or *homotopy analysis*, which is used to precisely simulate very unstable dynamical systems.

Chapter 11

Applications

WARNING: This chapter is a draft and has not been polished or proofread. As such, it may contain (potentially serious) errors. Please note that you are responsible for learning the correct content from the lectures and textbooks. Nevertheless, we hope that this is a useful study aid.

In this chapter, we will discuss some applications of the theory we have developed so far in this class. Our exploration will include deterministic control and the linear-quadratic regulator, stochastic control and the policy gradient algorithm, and support vector machines.

11.1 Deterministic Control and Linear-Quadratic Regulator

The first application we'll discuss is in the area of deterministic control.

Although control is usually thought of as related to motion, it can apply to any dynamical system where we can understand the state and its dependence on time based on some function such as $x(t+1) = f(x(t), u(t))$ which takes in a state $x(t)$ and a control input $u(t)$ and outputs the next state $x(t+1)$. The goal of control is to choose the control inputs $u(t)$ to achieve some desired behavior of the system.

Example 181 (Vertical Rocket System)**.** For example, we can consider a vertical rocket. Our goal is to maximize its height by time T. We can let $x_1(t)$ denote its height, $x_2(t)$ denote its vertical velocity, and $x_3(t)$ denote the weight of the rocket (which we will approximate as the weight of the fuel). As you can infer, the weight of the fuel will go down over time and that can affect the rocket's velocity. The forces pushing the rocket down are drag and gravity, and the upward force comes from the rocket's thrust.

Define the following equations for the forces:

- Inertial force: $x_3 \ddot{x_1} = x_3 \dot{x_2}$.
- Drag: $c_D \rho(x_1) x_2^2$ where c_D is a constant and $\rho(x_1)$ is the density of the air at height x_1.
- Gravity: $g x_3$
- Thrust: $c_T \dot{x_3}$ (proportional to fuel ejection)

Suppose we hvae some initial state $(x_1(0), x_2(0), x_3(0)) = (0, 0, M)$ and we want to maximize $x_1(T)$. We want to write this in a dynamical form, where $\vec{x}(t) = (x_1(t), x_2(t), x_3(t))^\top$ and $u(t) = -x_3(t)$. We can write the following equations:

$$\begin{aligned}
\dot{x_1}(t) &= x_2(t) \\
x_3(t)\dot{x}_2(t) &= -c_D\rho(x_1(t))x_2^2(t) + c_T\dot{u}(t) - gx_3(t) \\
\implies \dot{x_2}(t) &= -\frac{c_D\rho(x_1(t))x_2^2(t)}{x_3(t)} + \frac{c_T\dot{u}(t)}{x_3(t)} - g \\
\dot{x}_3(t) &= -u(t)
\end{aligned}$$

Now we have a dynamical system of the form $\dot{\vec{x}}(t) = f(\vec{x}(t), u(t))$. We can now apply our optimization skills to solve this problem. Let's write this as an optimization problem:

$$\begin{aligned}
\max_{u(t)} \quad & x_1(T) \\
\text{s.t.} \quad & \dot{\vec{x}}(t) = f(\vec{x}(t), u(t)) \forall t \in [0, T] \\
& \vec{x}(0) = \vec{x}_0
\end{aligned}$$

Then, we can use a numerical solver to solve this optimization problem.

Even though our dynamics were complex here, we can pretend we can write this as a **linear, time-invariant** system. For example, we could have: $\vec{x}(t+1) = A\vec{x}(t) + Bu(t)$ where A and B are matrices. This is a linear system because it is linear in $\vec{x}(t)$ and $u(t)$. It is time-invariant because the matrices A and B do not depend on t. Indeed, this is the form for the linear-quadratic regulator (LQR) problem.

If we look at our rocket system, can we write this as the above? Directly, we can't due to the nonlinearities, but we can use Taylor's theorem to linearize our system about an operating point, and then use the controls generated by a linear system to control the nonlinear system. In this section, we'll understand this linear system fully, which will help us understand the nonlinear system better.

Let's formally define our linear quadratic regulator system, along with a cost function to minimize:

Definition 182 (Linear Quadratic Regulator)
A linear quadratic regulator is a system of the form:

$$\begin{aligned}
\min_{\vec{x}_1,\ldots,\vec{x}_N,\vec{u}_1,\ldots,\vec{u}_N} \quad & \sum_{t=0}^{N-1} \frac{1}{2}\left(\vec{x}(t)^\top Q\vec{x}(t) + u(t)^\top Ru(t)\right) + \frac{1}{2}\vec{x}(N)^\top Q_f\vec{x}(N) \\
\vec{x}(t+1) &= A\vec{x}(t) + B\vec{u}(t) \\
\vec{x}(0) &= \vec{x}_0
\end{aligned}$$

where we can assume that $Q, Q_f, R \succeq 0$.

You can see that this is actually a quadratic program, since the objective is quadratic in the variables $\vec{x}(t)$ and $\vec{u}(t)$ and the constraints are linear. So, you should expect to be able to solve this with the methods you already know. However, solving it in the traditional QP sense is not the most efficient way to solve this problem especially with a lot of variables. The way it is traditionally solved is using the dynamic programming approach and Bellman's equation. However, in this section, we will solve it using the KKT conditions and the Riccati equation.

The Lagrangian is:

$$L(\vec{x}_0, \dots, \vec{x}_N, \vec{u}_0, \dots, \vec{u}_N, \vec{\lambda}_1, \dots, \vec{\lambda}_{N-1}) =$$
$$\sum_{t=0}^{N-1} \frac{1}{2}\left(\vec{x}(t)^\top Q\vec{x}(t) + u(t)^\top Ru(t)\right) + \frac{1}{2}\vec{x}(N)^\top Q_f \vec{x}(N) + \sum_{t=0}^{N-1} \vec{\lambda}_{t+1}^\top (A\vec{x}(t) + B\vec{u}(t) - \vec{x}(t+1))$$

Note that refined Slater's conditions holds by default so strong duality holds. Also, our problem is convex and differentiable, so the KKT conditions are necessary and sufficient. Most of the KKT conditions aren't useful or applicable here except for the stationarity condition. We can write the stationarity condition as:

$$\nabla_{\vec{x}(t)} L = Q\vec{x}(t) + A^\top \vec{\lambda}(t+1) - \vec{\lambda}(t) = 0 \forall t = 1, \dots, N-1$$
$$\nabla_{\vec{u}(t)} L = R\vec{u}(t) + B^\top \vec{\lambda}(t) = 0 \forall t = 0, \dots, N-1$$
$$\nabla_{\vec{x}(N)} L = Q_f \vec{x}(N) - \vec{\lambda}(N) = 0$$

The first condition implies that $\vec{\lambda}(t) = A^\top \vec{\lambda}(t+1) + Q\vec{x}(t)$. The last condition implies that $\vec{\lambda}(N) = Q_f \vec{x}(N)$. This looks very much like the system we started out with. We can call these the "dynamics" of the "adjoint system," with $\vec{\lambda}$ being called the "co-state." We can also write the implication of the second condition: $\vec{u}(t) = -R^{-1}B^\top \vec{\lambda}(t)$.

Now, we have something that is fit to solve with reverse induction: we can start with the last timestep and work backwards. It turns out that the optimal control is linear in the state, and we can prove this as well.

Theorem 183 (Optimal Control in LQR is Linear)
The optimal control for the LQR problem is linear in the state.

Proof. Our approach will be using backwards induction, with the goal of finding and proving the optimal $\vec{u}_t$. Our induction hypothesis will be that $\vec{\lambda}_{t+1} = P_{t+1}\vec{x}_{t+1}$.

Our base case is $t = N$, where we have $\vec{\lambda}(N) = Q_f \vec{x}(N)$. This implies that $P_N = Q_f$.

Now, we will prove the inductive step, which will just take some algebraic manipulation. To show that $\vec{\lambda}_t = P_t \vec{x}_t$, we have:

$$\vec{\lambda}_{t+1} = P_{t+1}\vec{x}_{t+1} = P_{t+1}(A\vec{x}_t + B\vec{u}_t) = P_{t+1}(A\vec{x}_t + B(-R^{-1}B^\top \vec{\lambda}_t))$$

Now, some simplification yields $\vec{\lambda}_{t+1} = (I + P_{t+1}BR^{-1}B^\top)^{-1} P_{t+1} A\vec{x}_t$. Now we have $\vec{\lambda}_t = A\vec{\lambda}_{t+1} + Q\vec{x}_t$. Substituting in our expression for $\vec{\lambda}_{t+1}$, we have:

$$\vec{\lambda}_t = A(I + P_{t+1}BR^{-1}B^\top)^{-1} P_{t+1} A\vec{x}_t + Q\vec{x}_t$$

Now, we have written $\vec{\lambda}_t$ as a matrix times $\vec{x}_t$. Formally, we would have to prove invertibility as well inductively, but otherwise, we have proven what we wanted to prove, which is that the optimal control is linear in the state. □

Now, note that we have written P_t as some function of P_{t+1}. Starting from P_N, we can compute the P_t's in a backward function. Once we have the P's, we can compute the λ's, and once we have the λ's, we can compute the u's. Therefore, we have a way to solve the LQR problem just by using matrix multiplication. We can even compute the P_t's offline, since they have no dependence on the state $\vec{x}$.

11.2 Support Vector Machines

Support vector machines are an application of optimization to the classical machine learning problem of classification. We will see that using optimization and a specific perspective induced by the KKT conditions will allow us to elegantly solve this classification problem.

But first, let's review what classification is. Suppose you have some image, and you want to figure out what kind of animal it is. For example, you could be classifying it as a cat or a dog. Or, it could be X-ray images, and you want to figure out if there is a fracture present or not. Nowadays, we use a wide range of techniques such as deep neural networks to solve these tasks. However, these build up from much more fundamental ideas such as support vector machines.

For the purpose of this class, we will focus on binary classification. In binary classification, we are given a bunch of points, and we want to separate them into two classes labeled $+1$ and -1. Our points can be called $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. We could try solving this using least squares, but we saw previously in the context of logistic regression that this doesn't work super well. Here, we'll derive a different technique.

Hard-Margin SVM

In particular, let's say we have some two-dimensional data. We want to find a line (or *separating hyperplane*) that separates the data into two classes. We can write this as $f(\vec{x}) = \vec{w}^\top \vec{x} + b = \vec{w}^\top(\vec{x} - \vec{x}_0)$ where $b = -\vec{w}^\top \vec{x}_0$. We can think of $\vec{x}_0$ as the offset of the line from the origin. We can also think of $\vec{w}$ as the normal vector to the line.

If I want to use this to separate my points, then if $y = +1$, I would like to have $f(\vec{x}) > 0$, and if $y = -1$, I would like to have $f(\vec{x}) < 0$. We can write this as $y_i f(\vec{x}_i) > 0$ for all i, combining these two cases into one inequality.

For now, we'll also have an additional assumption on our data: we want it to be *linearly separable*. This means that there exists some $\vec{w}$ and b such that our condition $y_i f(\vec{x}_i) > 0$ holds for all i. Of course, there are some problems with having such an assumption, and it might seem impractical right now (we'll deal with this later). But even if this is the case, there could be many different lines that separate the data. Which one should we choose?

It makes the most sense to choose the one with the largest *margin*, where the margin is defined as the distance of the hyperplane to the closest point to it. We want to have as large a buffer region around our separator, and the margin defines the width of this buffer region. So, we will look for the *max margin separator*. If we consider some point $\vec{x}$, with the hyperplane defined by $\vec{w}^\top \vec{x} + b = 0$, then the distance from $\vec{x}$ to the hyperplane is given by $\frac{|\vec{w}^\top \vec{x} + b|}{\|\vec{w}\|}$. We can see this by noting that the distance from $\vec{x}$ to the hyperplane is the projection of $\vec{x}$ onto the normal vector $\vec{w}$.

Now I have an equation in terms of the hyperplane in question, so I can write my maximum margin problem as an optimization problem. This looks like the following:

$$\begin{aligned} \max_{\vec{w}, b, m} \quad & m \\ \text{s.t.} \quad & y_i(\vec{w}^\top \vec{x}_i + b) > 0 \quad \forall i \\ & \frac{|\vec{w}^\top \vec{x}_i + b|}{\|\vec{w}\|} \geq m \quad \forall i \end{aligned}$$

This defines the *hard-margin SVM problem*. It's usually written in a slightly different form though:

$$\begin{aligned} \max_{\vec{w}, b, m} \quad & m \\ \text{s.t.} \quad & y_i(\vec{w}^\top \vec{x}_i + b) \geq m \quad \forall i \end{aligned}$$

$$m \geq 0$$

We can see that this is equivalent to the previous problem by noting that if $y_i(\vec{w}^\top \vec{x}_i + b) \geq m$, then $y_i(\vec{w}^\top \vec{x}_i + b) > 0$ as well. We can also see that the constraint $m \geq 0$ is redundant, since we are maximizing m, but we write it anyway to make the problem make more sense.

We can also make another simplification: we can choose the norm of $\vec{w}$: specifically, we can choose $\|\vec{w}\| = \frac{1}{m}$. This is because if we scale $\vec{w}$ by some constant, then the margin m will also scale by that constant. So, we can just choose $\|\vec{w}\| = \frac{1}{m}$. This gives us the following problem:

$$\begin{aligned} \max_{\vec{w}, b} \quad & \frac{1}{\|\vec{w}\|} \\ \text{s.t.} \quad & y_i(\vec{w}^\top \vec{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

This looks a little ugly, but we can make it look better by writing it as a minimization problem in this way:

$$\begin{aligned} \min_{\vec{w}, b} \quad & \frac{1}{2}\|\vec{w}\|^2 \\ \text{s.t.} \quad & y_i(\vec{w}^\top \vec{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

Now, we have a quadratic objective and linear constraints. This is a *quadratic program*, which we know how to solve. But what happens if we don't actually have linear separability in our data?

Soft Margin SVM

Most of our real world data isn't actually linearly separable. In that case, our hard margin SVM problem would just be infeasible. But maybe we still want to try to separate the points, even if our hyperplane won't perfectly divide the points into two classes.

Essentially, we can try allowing small violations of the margin. This changes our problem to the following:

$$\begin{aligned} \min_{\vec{w}, b, \vec{\xi}} \quad & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(\vec{w}^\top \vec{x}_i + b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

Here, we have a new variable $\vec{\xi}$, which is a vector of slack variables. We also have a new parameter C, which is a regularization parameter. If we didn't have this hyperparameter, we could just choose giant $\vec{\xi}$'s, so we need to do some regularization to prevent this. If C is large, then we are penalizing the slack variables a lot, and we are trying to minimize the number of violations of the margin. If C is small, then we are not penalizing the slack variables very much, and we are allowing more violations of the margin.

We can also think of the SVM problem as minimizing a loss function, e.g. the 0-1 loss function (step function). We get a loss of 1 if we classify a point incorrectly, and a loss of 0 if we classify a point correctly. The problem with this is that the step function loss is not convex. It turns out that the *hinge loss* function is a convex approximation of the step function loss. The hinge loss function is defined as follows:

$$\ell_{\text{hinge}}(z) = \max(0, 1 - z)\ell_{\text{hinge}}(1 - y_i(\vec{w}^\top \vec{x}_i + b)) = \max(0, y_i(\vec{w}^\top \vec{x}_i + b) - 1)$$

Now, how does this connect to our soft-margin SVM? We can rewrite the soft-margin SVM problem as the following:

$$\begin{aligned} \min_{\vec{w},b} \quad & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & \xi_i \geq \max(0, 1 - y_i(\vec{w}^\top \vec{x}_i + b)) \quad \forall i \end{aligned}$$

which we can rewrite as:

$$\min_{\vec{w},b} \quad \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\ell_{\text{hinge}}(1 - y_i(\vec{w}^\top \vec{x}_i + b))$$

Now, as opposed to our previous perspective, the first term is the one that looks like a regularizer. This gives us a hinge loss perspective of the soft-margin SVM problem.

KKT Conditions

Let's look at the insight that the KKT conditions can provide us here. Let's write out the Lagrangian for the earlier problem with the ξ's:

$$\begin{aligned} \mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\beta}) &= \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\vec{w}^\top \vec{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \\ &= \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i y_i(\vec{w}^\top \vec{x}_i + b) + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i + \sum_{i=1}^{n}\alpha_i \\ &= \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n}\alpha_i y_i(\vec{w}^\top \vec{x}_i + b) + \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}(C - \alpha_i - \beta_i)\xi_i \end{aligned}$$

Now, note that our problem is a QP, with a quadratic objective with a PSD Hessian. So we know this is convex, and our constraints are only linear so strong duality is going to hold. Thus, the KKT conditions are necessary and sufficient for optimality.

Then, we can write out the first order (stationarity) conditions from the Lagrangian:

$$\begin{aligned} \nabla_{\vec{w}}\mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\beta}) &= \vec{w} - \sum_{i=1}^{n}\alpha_i y_i \vec{x}_i = 0 \\ \frac{\partial}{\partial b}\mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\beta}) &= -\sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial}{\partial \xi_i}\mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\beta}) &= C - \alpha_i - \beta_i = 0 \end{aligned}$$

Recall the interpretation of dual variables as the price we're willing to pay to violate a constraint. Whether these α_i's are zero or not will play a big role for us in this way.

Complementary slackness gives us the following:

$$\begin{aligned} \alpha_i((1-\xi_i) - y_i(\vec{w}^\top \vec{x}_i + b)) = 0 \quad \forall i \\ \beta_i \xi_i = 0 \quad \forall i \end{aligned}$$

It turns out here that the nonzero α_i's here are going to determine what the *support vectors* of our problem will be. The support vectors are the points that are closest to the decision boundary.

Bibliography

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[2] G. Calafiore and L. El Ghaoui, *Optimization Models*. Cambridge University Press, 2014.

[3] C. C. Pugh, *Real Mathematical Analysis*. Springer, 2002, vol. 2011.

[4] P. Varaiya *et al.*, *Lecture notes on optimization*. Unpublished manuscript, University of California, Department of Electrical Engineering and Computer Science, 1998.

[5] G. Garrigos and R. M. Gower, “Handbook of convergence theorems for (stochastic) gradient methods,” *arXiv preprint arXiv:2301.11235*, 2023.

[6] D. P. Bertsekas, “Nonlinear programming,” *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[7] Y. Nesterov *et al.*, *Lectures on convex optimization*. Springer, 2018, vol. 137.