Introduction to Mathematics and Optimization

Niels Lauritzen

August 1, 2026

CONTENTS

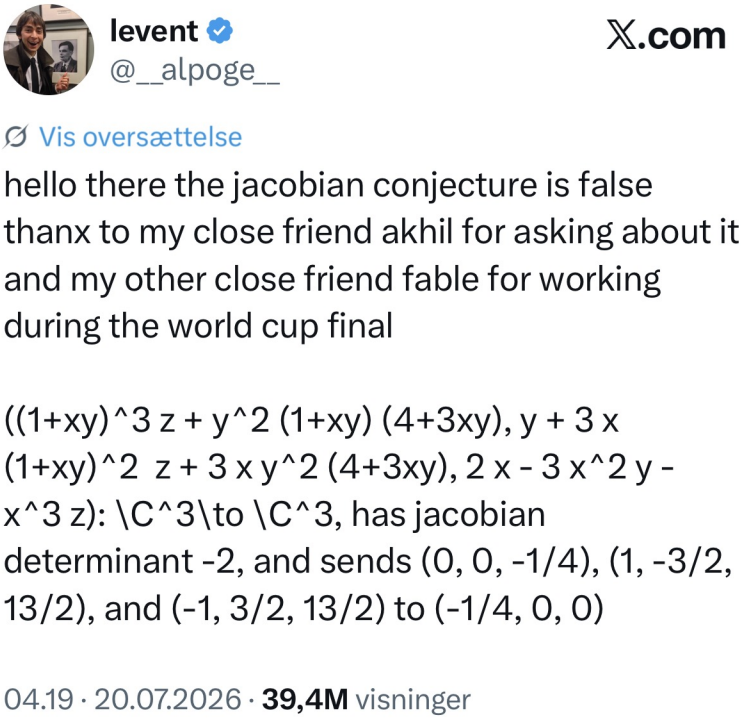
1 THE LANGUAGE OF MATHEMATICS AND PROMPTING

1.1 Prompting and learning

Generative AI has come a very long way since the introduction of ChatGPT on November 30, 2022, and it keeps evolving with explosive speed. Frontier large language models (LLM) do college level mathematics (and computer science) superbly. So much so, that the use of generative AI, in any form, is in **no way** allowed during the written exam in this course.

(1.1) REMARK (The LLM landscape, updated August, 2026).

The past year has seen generative AI once more developing at breakneck speed. Agents started entering the workflow in November 2025 and were matured in Claude Code, Codex, OpenClaw and the like in the beginning of 2026. Frontier models like Mythos was held back because of cybersecurity risks. Over the summer incredibly powerful models like Fable and GPT 5.6 helped solve hard open research problems in mathematics. In fact, Fable helped solve one of the most difficult math problems during the FIFA World Cup final. This was casually announced on X by Levent Alpöge in the tweet below



Make no mistake, this was a genuine really hard mathematical problem dating back to 1939, that captured the attention of very capable mathematicians.

When I write helped solve, it means that there was a human in the loop guiding the LLM. Oftentimes the LLM does a lot of the heavy lifting and also contributes central ideas.

The natural interface to LLMs is the chatbot in the browser. Through your AU university account you have access to Microsoft Copilot.

You communicate with chatbots through natural language. This process is called prompting. The more precise

your prompt is, the better the response. When learning new material, you can work from prompts instructing the chatbot not to give away the answers but emphasize guidance. The exercise below is a good example of this.

(1.2) EXERCISE.

LLM prompt — not displayed in the pdf version.

A click on a chatbot link copies the prompt to the clipboard and takes you to the chatbot, where you can paste the prompt. Click on a chatbot of your choice. Then attach the pdf version imo26.pdf of the interactive notes. Submit and interact.
Try different scenarios and chatbots. Browse through the beginning of the notes and change the prompt to suit you.

♠

Gemini has a mode called guided learning and ChatGPT has something called study mode that you may also use. These are modes that support learning in the conversation with the chatbot.

In any case, precise prompting is a very valuable skill.

In this chapter several examples of prompts will be given. In the following chapters less so. Here you are expected to prompt the chatbots on your own. Sometimes a prompt related to the context pops up as below.

LLM prompt — not displayed in the pdf version.

(1.3) EXERCISE.

Come up with prompts that make a chatbot act like a mathematics tutor for you. Here is a small example that you may extend.

LLM prompt — not displayed in the pdf version.

Try out the features guided learning in Gemini and study mode in ChatGPT on the example above. ♠

(1.4) EXERCISE.

Start your LaTeX journey using the prompt below.

LLM prompt — not displayed in the pdf version.

Come up with your own prompt for a question related to software. ♠

(1.5) EXERCISE.

Below I ask for feedback from the chatbot on some dubious chunk of mathematics.

LLM prompt — not displayed in the pdf version.

Insert your own mathematics in LaTeX notation and ask for feedback in a prompt. ♠

1.1.1 AI atrophy

AI atrophy is the gradual weakening of human thinking by outsourcing cognitive efforts to AI. A typical example of this phenomenon is given in the prompt below.

LLM prompt — not displayed in the pdf version.

Here every inch of the cognitive effort is outsourced. Once in a while we all crawl down this rabbit hole. Personally, I get depressed using such mindless interaction. More importantly, it is arguably the worst way of preparing for the exam in this course.

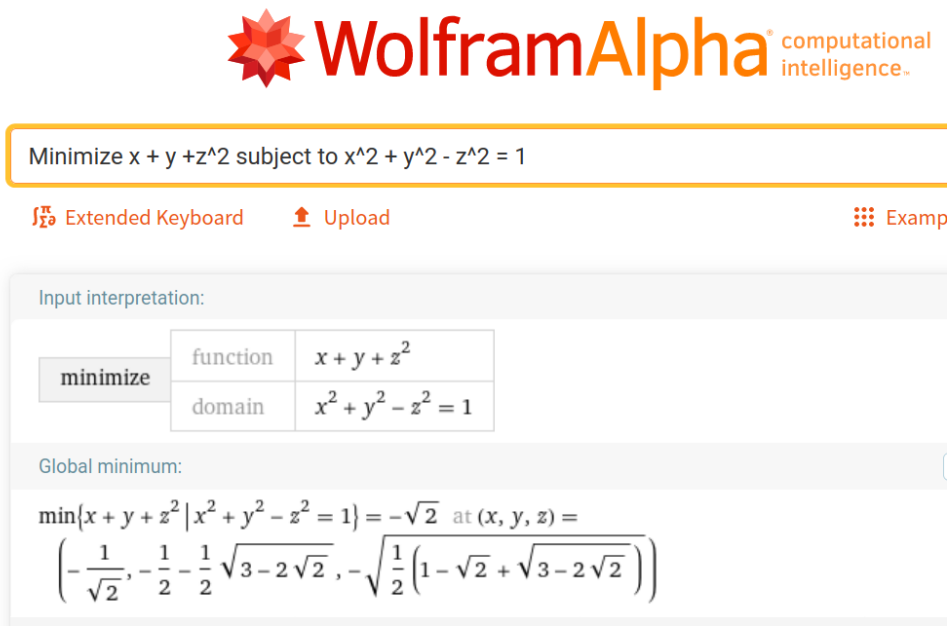
I cannot emphasize enough that approaching learning in this way is extremely ineffective. The Zen state of learning in this day and age is embodied in the quote below due to Andrej Karpathy.

> You can outsource your thinking, but you cannot outsoruce your understanding.

But make no mistake. To gain understanding you also need to do some thinking. Only later when your understanding is on firm ground this outsourcing to AI makes sense.

1.2 Python as your mathematical laboratory

Computers are exceptionally fun, but be careful! Nothing really beats a clear thinking human mind. To wit, I asked WolframAlpha to solve a certain optimization problem and it came up with the answer



(1.6) EXERCISE.

Prompt a chatbot with

LLM prompt — not displayed in the pdf version.

and explain why the output from WolframAlpha is weird. Use prompting to make it explain the mathematics input notation.

Finally use your own mental powers (and feedback from the chatbot) to explain what the proper output should have been.

Hint:

$$3-2\sqrt{2}=(\sqrt{2}-1)^2.$$

♠

We will use the programming language Python in exploring and experimenting with mathematics. The interactive notes contain live Python cells: you edit small snippets of code directly on the page, press Run and the code executes in your browser. No installation is needed. Along with Python come powerful libraries: numpy for numerics, matplotlib for plotting and sympy for computer algebra i.e., exact symbolic computation with numbers, symbols and equations.

First adjust the prompt below according to your needs and get feedback from a chatbot.

LLM prompt — not displayed in the pdf version.

Below is an example of a basic graphics command in Python. Push the Run button to evaluate.

python cell — not displayed in the pdf version.

You can also run Python on your own computer, for example through Jupyter notebooks. For heavier computer algebra there is the full-blown system Sage, built on top of Python, which earlier editions of these notes used.

(1.7) EXERCISE.

Did you notice that you can edit and enter new commands in the Python cell? Do the following problems using Python — the documentation for numpy, matplotlib and sympy is helpful, and so is asking a chatbot.

(i) Consider $f(x)=x\sin(1/x)$. Plot the graph of f from 0 to 0.1. Computing $f(0)$ does not make sense. Do you see a way of assigning a natural value to $f(0)$ using the graph?

(ii) Find an approximate solution with four decimals to the equation $\cos(x)=x$.

Hint: This is an example of an equation, that can only be solved numerically. Try first plotting the graph of $f(x)=x-\cos(x)$ from 0 to 1. Then use a suitable function from scipy or sympy.

(iii) Compute π with 100 decimals.

LLM prompt — not displayed in the pdf version.

♠

(1.8) EXERCISE.

Compute the sum

$$\frac{1}{\sqrt{1}+\sqrt{2}}+\frac{1}{\sqrt{2}+\sqrt{3}}+\frac{1}{\sqrt{3}+\sqrt{4}}.$$

What is the elegant answer? Explain!

Bonus question. Generalize your answer/method to computing the sum

$$\frac{1}{\sqrt{1}+\sqrt{2}}+\frac{1}{\sqrt{2}+\sqrt{3}}+\frac{1}{\sqrt{3}+\sqrt{4}}+\cdots+\frac{1}{\sqrt{N-1}+\sqrt{N}},$$

for $N=5,6,7,\ldots$.

♠

We need a precise setup for communicating mathematics. This involves the introduction of propositional logic, predicates and sets. Let me emphasize, that this is an introductory course and not a rigorous introduction to mathematics. As such it is an organic approach, where I hope that you will return and fill out the gaps instead of getting overwhelmed by formal details already from the beginning.

However, the underlying goal is to show that mathematical precision and proofs are similar to constructing correct computer programs.

In fact, this whole first chapter may be viewed as the beginning of a computer program, where we state the exact definitions for use in the following chapters. The plan is this: first the language of mathematics (propositional logic and sets), then its fundamental objects (numbers and their ordering), then the rules of reasoning (mathematical proofs, including induction and proof by contradiction) and finally the concept of a function — ending with a first look at neural networks, the machinery driving modern AI.

1.3 Propositional logic and predicates

A proposition is a (mathematical) statement or sentence that is true (t) or false (f). This could be a boolean expression in a computer program, like $1 < 2$ or an everyday outburst like *I am tired*. Both of these can be assigned true or false in a meaningful way.

(1.9) REMARK.

In examples we will freely use numbers you know from school, like $1 < 2$, and occasionally a *set* S — simply a collection of objects, where $x \in S$ means that x belongs to S. Sets and numbers get their full formal treatment in the two sections following this one.

Python.

python cell — not displayed in the pdf version.

Later we will see propositions with variables in them like $x < 2$. These are called predicates.

Propositions can be combined into new (compound) propositions. Take for example the propositions p: *it rains* and q: *it is cloudy*.

Then (p and q) is a perfectly good new proposition reading *it rains and it is cloudy*. The same goes for (if p then q), which reads *if it rains then it is cloudy*. The proposition (if q then p) reads *if it is cloudy then it rains*. This proposition is (clearly) false.

We need some notation to describe these compound propositions:

$p \wedge q$	p and q
$p \vee q$	p or q
$p \implies q$	if p then q
$\neg p$	not p

The compound propositions are either true(t) or false (f) depending on p and q. The dependencies are displayed in the *truth tables* below.

(1.10) DEFINITION.

p	q	$p \wedge q$
t	t	t
t	f	f
f	t	f
f	f	f

p	q	$p \vee q$
t	t	t
t	f	t
f	t	t
f	f	f

p	q	$p \implies q$
t	t	t
t	f	f
f	t	t
f	f	t

p	$\neg p$
t	f
f	t

The tables for the compound propositions $p \wedge q, p \vee q$ and also $\neg p$ are not too hard to grasp. The table for $p \implies q$ raises a few more questions. Why is $f \implies t$ true? I will not go into this at this point (see Example 1.23), but just point out that there are many explanations available online and, perhaps more importantly, refer you to Exercise 1.11.

(1.11) EXERCISE.

Suppose that we are presented with four cards

(1.1)

with a (natural) number on the front and the color blue or red on the back. In (1.1), the first and third cards are shown with their fronts facing up and the second and fourth cards are shown with their backs facing up.

A claim (proposition) is made that if a card has an even number on the front, then it must have the color blue on the back.

Your task is to verify this for the cards above. Of course you can do this by turning all four cards, but is there a way of checking this by turning less than four cards?

What if we add the claim, that if a card has the color blue on the back, then it must have an even number on the front?

Hint: Find two propositions p and q so that the claim reads $p \implies q$.

♠

(1.12) EXERCISE.

A prosecutor says to the defendant: "If you committed this crime you did not act alone". Explain why the defendant should not answer "no, that is not true" here. ♠

(1.13) EXERCISE.

Explain why Python thinks that the value[1] of

python cell — not displayed in the pdf version.

is False! Notice that you are dividing one by zero in the last "integer" above. ♠

[1] Thanks to Gerth Brodal for pointing this out to me

1.3.1 Propositional logic as a formal language

The entities p and q above may have real world interpretations like *it rains* or *it is cloudy*, but we will view them as variables that can be assigned the values true or false. Independent of this assignment we define a proposition as follows.

(1.14) DEFINITION.

A proposition in the variables $x_1,\ldots,x_r$ is an expression involving the symbols $x_1,\ldots,x_r,(,),\neg,\wedge,\vee,\Longrightarrow$ that can be generated using the rules below

(i) The variables $x_1,\ldots,x_r$ are (atomic) propositions.

(ii) If P is a proposition, then $(\neg P)$ is a proposition.

(iii) If P and Q are propositions, then $(P\wedge Q)$, $(P\vee Q)$ and $(P\Longrightarrow Q)$ are propositions.

(1.15) EXAMPLE.

The expression $(x_1\Longrightarrow((\neg x_2)\vee x_3))$ is a proposition. Let us see how it is generated using the rules in Definition 1.14.

(1) First, x_2 is a proposition using (i).

(2) Then $(\neg x_2)$ is a proposition by using (ii) with $P=x_2$, since we know by (1) that P is a proposition.

(3) Since x_3 is a proposition by (i), it follows that $((\neg x_2)\vee x_3)$ is a proposition using (iii) with $P=(\neg x_2)$ and $Q=x_3$, since we know by (2) that P is a proposition.

(4) Finally, since x_1 is a proposition by (i) it follows by (iii) with $P=x_1$ and $Q=((\neg x_2)\vee x_3)$ that

$$(x_1\Longrightarrow((\neg x_2)\vee x_3))$$

is a proposition, since we know by (3) that Q is a proposition.

♠

(1.16) QUIZ.

quiz — not displayed in the pdf version.

♠

1.3.2 Truth tables and equivalent propositions

Given a proposition, it makes sense to substitute values (t or f) for the variables and evaluate it using the rules in Definition 1.10, since the parentheses leave no ambiguity as to how the evaluation must take place. For a given proposition in the variables $x_1,\ldots,x_r$, there are 2^r ways of assignments to the set of variables. Each of these assignments results in the value true or false after evaluation. This is conveniently recorded in the *truth table* of the proposition as illustrated in the example below, where $r=3$ so that the truth table has $2^3=8$ rows.

(1.17) EXAMPLE.

The truth tables corresponding to the propositions $(x_1 \wedge (x_2 \vee x_3))$ and $((x_1 \wedge x_2) \vee (x_1 \wedge x_3))$ are given below.

x_1	x_2	x_3	$(x_1 \wedge (x_2 \vee x_3))$
f	f	f	f
f	f	t	f
f	t	f	f
f	t	t	f
t	f	f	f
t	f	t	t
t	t	f	t
t	t	t	t

x_1	x_2	x_3	$((x_1 \wedge x_2) \vee (x_1 \wedge x_3))$
f	f	f	f
f	f	t	f
f	t	f	f
f	t	t	f
t	f	f	f
t	f	t	t
t	t	f	t
t	t	t	t

For example, if $x_1 = t, x_2 = f$ and $x_3 = t$, then

$$(x_1 \wedge (x_2 \vee x_3)) = (t \wedge (f \vee t)) = (t \wedge t) = t$$

and

$$((x_1 \wedge x_2) \vee (x_1 \wedge x_3)) = ((t \wedge f) \vee (t \wedge t)) = (f \vee t) = t.$$

♠

The two propositions in Example 1.17 have identical truth tables. In general, if two propositions P and Q have identical truth tables we call them *equivalent* and write

$$P \equiv Q.$$

In Example 1.17 we saw that

$$(x_1 \wedge (x_2 \vee x_3)) \equiv ((x_1 \wedge x_2) \vee (x_1 \wedge x_3)).$$

The definition below is very important to keep in mind.

(1.18) DEFINITION.

The notation $p \iff q$ is used frequently. It means that both $p \implies q$ and $q \implies p$ are true i.e.,

$$p \iff q \equiv (p \implies q) \wedge (q \implies p).$$

1.3.3 Computing truth tables in Python

Python may be used to compute truth tables for propositions using the logic module in sympy. Below is an example. Be sure to check how to enter $\wedge, \vee, \implies$ and $\neg$.

python cell — not displayed in the pdf version.

(1.19) EXERCISE.

Construct by hand the truth table for the proposition $(p \wedge q) \vee (\neg r)$. ♠

(1.20) EXERCISE.

Convince yourself either using Python or by writing out truth tables that

(i) $(x_1 \implies x_2) \equiv ((\neg x_2) \implies (\neg x_1))$

(ii) $(\neg(x_1 \vee x_2)) \equiv ((\neg x_1) \wedge (\neg x_2))$

(iii) $\neg(x_1 \wedge x_2) \equiv (\neg x_1) \vee (\neg x_2)$

(iv) $x_1 \implies x_2 \equiv (\neg x_1) \vee x_2$

♠

1.3.4 Variables, predicates and quantification

(1.21) DEFINITION (predicate).

A predicate is a proposition depending on one or more variables.

Variables are fundamental in computer programs. In the predicate $x = 1$, x appears as a variable. Depending on what you substitute for x, the resulting proposition could be true or false or even meaningless. As an example, the latter case appears if x is replaced by the character 'a'. This is what computer scientists call a type error. You cannot compare a character with a digit.

(1.22) EXAMPLE.

If $p(n) = n$ is a prime number, then $p(3)$ is true, whereas $p(6)$ is false.

$$q(m,n) = p(m) \wedge (\neg p(n))$$

is a predicate in two variables m and n. Here $q(3,4)$ is true, whereas $q(5,7)$ is false. ♠

For every $\forall$ and there exists $\exists$

Suppose that we have a predicate $p(x)$, such that $p(x)$ is a proposition for $x \in S$, where S is some set. Then we define the proposition

$$\exists x \in S : p(x) \tag{1.2}$$

to be true if there exists $x \in S$, such that $p(x)$ is true. We let

$$\forall x \in S : p(x) \tag{1.3}$$

be the proposition defined by

$$\neg(\exists x \in S : \neg p(x)).$$

In other words, (1.3) says that $p(x)$ is true for every $x \in S$, since there does not exist $x \in S$ making $p(x)$ false. Let me be absolutely clear. To show that $\forall x \in S : p(x)$ is false, it is enough to find just a single $x \in S$ so that $p(x)$ is false.

(1.23) EXAMPLE.

Here is a statement about real numbers

$$x^2 = 1 \qquad \implies (x-1)(x+1)(x-2) = 0 \tag{1.4}$$

This statement reads: no matter which real number x you pick, if $x^2 = 1$, then $(x-1)(x+1)(x-2) = 0$. We definitely want this to be true. Being true means that (1.4) must hold for all numbers x, also $x = 2$, which reads

$$2^2 = 4 = 1 \qquad \Longrightarrow \qquad (2-1)(2+1)(2-2) = 0 = 0$$

The above statement is an example of a false implies true statement, which we want to be true.

In general terms, in proving the statement that $p(x) \implies q(x)$ holds for every x in some set S, we are really only interested in $x \in S$ for which $p(x)$ is true, since $p(x)$ is our assumption. We still need $p(x) \implies q(x)$ to be true for $x \in S$ for which $p(x)$ is false. This is assured by the truth table for $\implies$, since $f \implies t$ and $f \implies f$ are both true. ♠

The following is an excerpt from the infamous *Beredskabsprøve Datalogi*.

(1.24) QUIZ.

quiz — not displayed in the pdf version.

♠

1.4 Sets

Propositions are important, but are confined by the binary values of true and false. We would like to work mathematically with objects like integers, floating point numbers, neural networks, computer programs and so on.

1.4.1 Objects and equality

One of the cornerstones of modern mathematics is deciding when two objects are the same i.e., given two objects A and B, deciding whether the proposition $A = B$ is true or false. Oftentimes an algorithm for evaluating $A = B$ is needed.

You may laugh here, but this is not always that easy. Even though objects appear different they are the same as in, for example the propositions

$$\frac{105}{189} = \frac{35}{63} \qquad \text{and} \qquad \sin\left(\frac{\pi}{2}\right) = 1.$$

The first proposition above is an identity of fractions (rational numbers). The second is an identity, which calls for knowledge of the sine function and real numbers. Each of these identities calls for some rather advanced mathematics. The first proposition is true in a very precise way, since $105 \cdot 63 = 189 \cdot 35$.

(1.25) EXERCISE.

python cell — not displayed in the pdf version.

Use the Python cell above to reason about equality in the quiz below. In each case describe the objects i.e., are they numbers, symbols, etc.? Also, please check your computations by hand with the old fashioned paper and pencil, especially $(a+b)(a-b)$.

quiz — not displayed in the pdf version.

♠

(1.26) EXERCISE.

You know that $(a+b)^2 = a^2+2ab+b^2$. Use Python to find similar identities for $(a+b)^3$ and $(a+b)^4$.

Hint: Go back and look at (the beginning of) Exercise 1.25. ♠

For two objects A and B we will use the notation $A \neq B$ for the proposition $\neg(A = B)$.

So far we have used sets informally as collections of distinct objects or *elements*. We introduce set theory properly here. A set is also an object as described in section 1.4.1 and it makes sense to ask when two sets are equal.

(1.27) DEFINITION.

Two sets A and B are equal i.e., $A = B$ if they contain the same elements.

An example of a set could be the set $\{1,2,3\}$ of natural numbers between 0 and 4. Notice again that we use the symbol "{" to start the listing of elements in a set and the symbol "}" to denote the end of the listing. Notice also that (by our definition of equality between sets), the order of the elements in the listing does not matter i.e.,

$$\{1,2,3\} = \{2,3,1\}.$$

We are also not allowing duplicates like for example in the listing $\{1,2,2,3,3,3\}$ (such a thing is called a multiset).

An example of a set not involving numbers could be the set of letters

$$S = \{A,n,e,x,a,m,p,l,c,o,u,d,b,t,h,s,r,i\}$$

used in this sentence. The number of elements in a set S is called the *cardinality* of the set. We will denote it by $|S|$.

To convince someone beyond a doubt (we will talk about this formally later in this chapter) that two sets A and B are equal, one needs to argue that if x is an element of A, then x is an element of B and the other way round, if y is an element of B, then y is an element of A. If this is true, then A and B must contain the same elements.

(1.28) EXERCISE.

Give a precise reason as to why the two sets $\{1,2,3\}$ and $\{1,2,4\}$ are not equal. Is it possible for a set with 5 elements to be equal to a set with 7 elements? ♠

Sets may be explored using python. This is illustrated in the snippet below.

python cell — not displayed in the pdf version.

(1.29) EXERCISE.

Come up with three lines of python code that verifies $\{1,2,3\} \neq \{1,2,4\}$. Try it out. ♠

The empty set

There is a unique set containing no or zero elements. This set is called the empty set and is denoted $\emptyset$ i.e.,

$$\emptyset = \{\} \qquad \text{and} \qquad |\emptyset| = 0.$$

python cell — not displayed in the pdf version.

(1.30) EXERCISE.

For some reason (perhaps a good one) python does not accept $\{\}$ as input for the empty set. Why is this? Evaluate the python snippet below and explain.

python cell — not displayed in the pdf version.

♠

1.4.2 The symbols $\in$ and $\notin$

The symbol $\in$ is ubiquitous in set theory (and mathematics). If A is a set, then

$$x \in A \tag{1.5}$$

is a proposition. It is true if x *is an element of* or *belongs to* A. The notation

$$x \notin A$$

is defined by the proposition $\neg(x \in A)$. Also, as a bit of short hand notation, we will write

$$a_1, a_2, \ldots, a_n \in A \qquad \text{for the proposition} \qquad (a_1 \in A) \wedge (a_2 \in A) \wedge \cdots \wedge (a_n \in A).$$

Belongs to ($\in$) is straightforward in python.

python cell — not displayed in the pdf version.

(1.31) QUIZ.

paragraphquiz — not displayed in the pdf version.

♠

1.4.3 Subsets

If A and B are sets, then $A \subseteq B$[2] means that every element of A is an element of B. So $A \subseteq B$ is a placeholder for the proposition

$$\forall x \in A : x \in B$$

In this case we say that A *is a subset of* B. We also use the notation $A \subsetneq B$ to indicate that $A \subseteq B$ and $A \neq B$. In this case we say that A is a *strict* subset of B.

[2] At times, the symbol $\subset$ is used instead of $\subseteq$. In our context these two symbols mean the same. However, the notation $A \subsetneq B$ means that $A \subseteq B$ and $A \neq B$. For example, $\{1,2,3\} \subseteq \{1,2,3\}$ and $\{1,2,3\} \subset \{1,2,3\}$.

(1.32) EXERCISE.

List the subsets of $\{1,2\}$. How many are there? ♠

(1.33) EXERCISE.

It turns out that the empty set $\emptyset$ is a subset of any set.

python cell — not displayed in the pdf version.

Explain why this is so using the definition of $\subseteq$.

LLM prompt — not displayed in the pdf version.

♠

(1.34) EXERCISE.

Below Python will list all subsets of the set $\{1,2,3\}$. Before pressing the Run button, try to write them down on your own.

python cell — not displayed in the pdf version.

List all the subsets of a set with five elements. In general, how many subsets does a set with n elements have? ♠

(1.35) QUIZ.

paragraphquiz — not displayed in the pdf version.

♠

(1.36) QUIZ.

paragraphquiz — not displayed in the pdf version.

♠

1.4.4 Set-builder notation

If S is a set and $p(x)$ a predicate for $x \in S$, then we build the subset

$$\{x \in S \mid p(x)\} \subseteq S \tag{1.6}$$

of $x \in S$ such that $p(x)$ is true.

(1.37) EXAMPLE.

Suppose that $S = \{-2,-1,2,3\}$ and

$$p(x) = x \text{ is positive.}$$

Then

$$\{x \in S \mid p(x)\} = \{2,3\} \subseteq S.$$

♠

Python. This notation has found its way to several programming languages like list comprehension in python.

python cell — not displayed in the pdf version.

Suppose that $p_1(x), \ldots, p_n(x)$ are predicates with a variable x taking values in S, then we often use the notation (using , instead of $\wedge$)

$$\{x \in S \mid p_1(x), \ldots, p_n(x)\} \quad \text{for} \quad \{x \in S \mid p_1(x) \wedge \cdots \wedge p_n(x)\}.$$

(1.38) EXERCISE.

List the elements in the following subsets.

(i)
$$\{x \in \mathbb{Z} \mid x^2 - 5x + 6 = 0\}.$$

(ii)
$$\{(x,y) \mid x \in \mathbb{Z}, y \in \mathbb{Z}, x^2 + y^2 < 5\}.$$

♠

(1.39) EXERCISE.

Consider the predicate $q(n)$

$$n \text{ and } n+2 \text{ are both prime numbers.}$$

Write down the elements in

$$\{n \in \mathbb{N} \mid q(n) \wedge n \leq 50\}.$$

Is

$$\{n \in \mathbb{N} \mid q(n)\}$$

an infinite set?

Hint: Explore the fascinating world of prime numbers and learn about twin primes. ♠

(1.40) EXERCISE.

You have previously encountered systems of linear equations like

$$x + y = 3 \tag{1.7}$$
$$3x - y = 5. \tag{1.8}$$

The solutions to (1.7) can be identified with a subset of $\mathbb{R}^2$. Define this subset precisely i.e., write the subset as

$$\{(x,y) \in \mathbb{R}^2 \mid p(x,y)\},$$

where $p(x,y)$ is a predicate in the variables $x,y \in \mathbb{R}$. ♠

(1.41) EXERCISE.

Suppose that $X = \mathbb{R}$ and

$$Y = \{x \in \mathbb{R} \mid x > 0, x < 2\}.$$

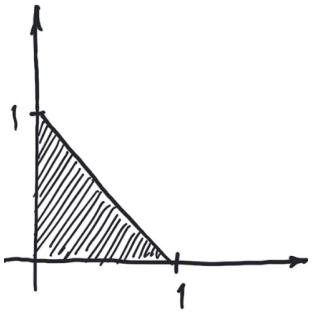
Then write down precisely what $X \setminus Y = \{x \in X \mid x \notin Y\}$ is i.e., find suitable predicates q_1, q_2 in the variable x, such that $q(x) = q_1(x) \vee q_2(x)$ and

$$X \setminus Y = \{x \in \mathbb{R} \mid q(x)\}.$$

♠

(1.42) EXERCISE.

Consider the subset S of $\mathbb{R}^2$ pictured in the drawing below



Express S as

$$S = \{(x,y) \in \mathbb{R}^2 \mid p_1(x,y), p_2(x,y), p_3(x,y)\},$$

where p_1, p_2, p_3 are predicates in the variables x,y.

Hint. A predicate in the variables x,y could be something like

$$x - y \geq 17.$$

Express $\mathbb{R}^2 \setminus S$ as

$$\{(x,y) \in \mathbb{R}^2 \mid p(x,y)\},$$

where

$$p(x,y) = q_1(x,y) \vee q_2(x,y) \vee q_3(x,y)$$

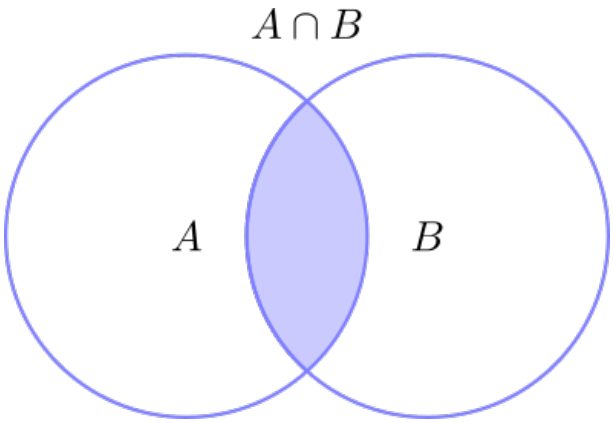
and q_1, q_2 and q_3 are suitable predicates in the variables x,y. ♠

1.4.5 Intersections, unions and the symbols $\cap$, $\cup$ and $\setminus$

Suppose that we have two sets A and B. Then the *intersection* $A \cap B$ is the set consisting of the elements in both A and B i.e.,

$$A \cap B = \{x \mid (x \in A) \wedge (x \in B)\}.$$

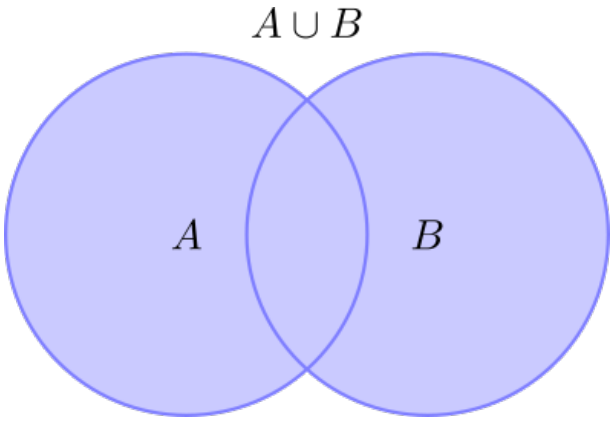
This is illustrated in the socalled Venn diagram below.



The *union* $A \cup B$ is the set consisting of the elements in A or B i.e.,

$$A \cup B = \{x \mid (x \in A) \vee (x \in B)\}.$$

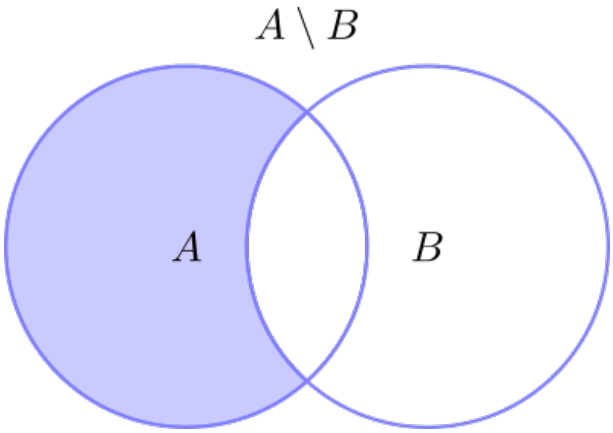
This is illustrated in the Venn diagram below.



Lastly, the difference $A \setminus B$ (between A and B) consists of the elements in A not contained in B i.e.,

$$A \setminus B = \{x \mid (x \in A) \wedge (x \notin B)\}.$$

This is illustrated in the Venn diagram below.



Python. You should experiment using the python window below to get a feeling for these three operations.

python cell — not displayed in the pdf version.

(1.43) EXERCISE.

Suppose that $A = \{1,2,3,4,5\}$, $B = \{-1,3,4,7\}$ and $C = \{2,3,8,9\}$. What is $((A \cup B) \setminus C) \setminus B$? ♠

(1.44) EXERCISE.

Let $A = \{1,2,3\}$, $B = \{3,4,5\}$ and $C = \{0,1,5\}$. Verify by hand (no computer) that

(i) $A \cup B = \{1,2,3,4,5\}$.

(ii) $A \cap B = \{3\}$.

(iii) $A \cap (B \cap C) = \emptyset$.

(iv) $B \setminus A = \{4,5\}$.

(v) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

♠

(1.45) EXERCISE.

Given two sets A and B, is it true that $A \cap B = B \cap A$ and $A \cup B = B \cup A$?

What about $A \setminus B = B \setminus A$?

Suppose that A and B are two finite sets. Is it true that

$$|A \setminus B| = |A| - |B|?$$

What about

$$|A \cup B| = |A| + |B|?$$

Seriously, both formulas are wrong. Can you come up with the correct version of the formula for $|A \cup B|$?

Use your correct formula to find a formula for

$$|A \cup B \cup C|$$

viewing $A \cup B$ as the first set and C as the second set. Here you need the formula

$$(A \cup B) \cap C = (A \cap C) \cup (B \cap C).$$

Why is this formula true? Finally, explain why

$$C \setminus (A \cap B) = (C \setminus A) \cup (C \setminus B).$$

Hint. You may find it useful to notice that two sets S_1, S_2 are equal i.e, $S_1 = S_2$ if and only if

$$x \in S_1 \iff x \in S_2.$$

Also,

$$\begin{aligned} x \in S_1 \cup S_2 &\iff x \in S_1 \vee x \in S_2 \\ x \in S_1 \cap S_2 &\iff x \in S_1 \wedge x \in S_2 \\ x \in S_1 \setminus S_2 &\iff x \in S_1 \wedge x \notin S_2 \\ x \notin S_1 &\iff \neg(x \in S_1). \end{aligned}$$

♠

(1.46) EXERCISE.

There is one more operation called the symmetric difference between two sets A and B. It is denoted $A \Delta B$. Experiment in the python window below to find out exactly what it does. Is it true that $A \Delta B = B \Delta A$?

python cell — not displayed in the pdf version.

♠

Here is another excerpt from the *Beredskabsprøve Datalogi*.

(1.47) QUIZ.

quiz — not displayed in the pdf version.

♠

1.4.6 Pairs, triples and tuples

Oftentimes we want to consider more than one variable as input to a predicate. It is convenient to group the variables into one object consisting of the variables. This is done using tuples.

Given two sets A and B, we combine two elements $a \in A$ and $b \in B$ into a pair (a,b), which is an element of the Cartesian product

$$A \times B = \{(a,b) \mid a \in A, b \in B\}.$$

of A and B. This is a new set built from A and B.

(1.48) EXAMPLE.

If $A = \{1,2\}$ and $B = \{1,2,3\}$, then

$$A \times B = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3)\}.$$

♠

(1.49) EXERCISE.

Consider two pairs (a,b) and (c,d) each from in $A \times B$. When is $(a,b) = (c,d)$? ♠

Python. The Cartesian product can be computed in python as shown below.

python cell — not displayed in the pdf version.

There is no need to restrict ourselves to pairs. We might as well consider triples $A \times B \times C$ i.e., the set of all (a,b,c), where A, B and C are sets, or for that matter general tuples

$$(a_1, a_2, \ldots, a_n) \in A_1 \times A_2 \times \cdots \times A_n \tag{1.9}$$

of any length $n \in \mathbb{N}$, where $a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$. Based on the above example with tuples we have,

$$\{0\} \times \{1,2\} \times \{1,2,3\} =$$
$$\{(0,1,1),(0,1,2),(0,1,3),(0,2,1),(0,2,2),(0,2,3)\}.$$

You may check this using the python snippet below.

python cell — not displayed in the pdf version.

(1.50) DEFINITION.

For a given set A and $n \in \mathbb{N}$ we define the n-fold cartesian product of A as

$$A^n = \underbrace{A \times A \times \cdots \times A}_{n \text{ times}}.$$

(1.51) EXERCISE.

Formally $\mathbb{R}^2$ is the set of pairs (a,b), where $a,b \in \mathbb{R}$. Is there a natural way of drawing elements in $\mathbb{R}^2$? ♠

(1.52) EXERCISE.

Let A and B be two sets. Is $A \times B = B \times A$?

Let X be any set. What is $\emptyset \times X$?

Let A, B, C and D be four sets. Is

$$(A \times B) \cap (C \times D) = (A \cap C) \times (B \cap D)?$$

Is

$$(A \times B) \setminus (C \times D) = (A \setminus C) \times (B \setminus D)?$$

Hint: See Exercise 1.53. ♠

(1.53) EXERCISE.

Use python to solve Exercise 1.52 by playing with (and extending) the code below.

python cell — not displayed in the pdf version.

♠

1.5 Numbers and their ordering

Our fundamental mathematical objects in this course are numbers. With logic and sets in our toolbox, we can introduce them precisely.

1.5.1 The natural numbers $\mathbb{N}$ and the integers $\mathbb{Z}$

The set of natural numbers is

$$\mathbb{N} = \{1, 2, 3, \dots\}. \tag{1.10}$$

The set of integers is

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}. \tag{1.11}$$

These are *infinite* sets, since they contain infinitely many elements as indicated by the dots ...

It makes sense to add and multiply two integers a and b. We will denote their addition or sum as $a+b$ and their multiplication or product as ab. A fundamental fact is that addition and multiplication are commutative i.e., $a+b=b+a$ and $ab=ba$.

Please notice right away that expressions like $a+b+c$ and abc are complete nonsense for three integers a, b and c. We only know how to add and multiply two integers, not three. A wonderful fact comes to our rescue:

$$(a+b)+c = a+(b+c) \tag{1.12}$$

$$(ab)c = a(bc). \tag{1.13}$$

You get the same result no matter if you start adding (multiplying) a and b or b and c and then adding (multiplying) c or a. So we may write $a+b+c$ and abc as a placeholder for one of the two ways of computing this expression in (1.12).

As you can see we use the symbol $+$ for addition, but no symbol for multiplication. This is the convention in (clean) mathematics as opposed to the $a*b$ coming from computer algebra (except perhaps in Mathematica or Wolfram language, where space is allowed for multiplication). However, when one of the factors is an actual number, we will use $\cdot$, so that for example 7 times 9 is written as $7 \cdot 9$ and a times 3 is written $a \cdot 3$.

1.5.2 The rational numbers $\mathbb{Q}$

The natural numbers $\mathbb{N}$ and the integers $\mathbb{Z}$ are well defined by their representations in (1.10) and (1.11).

(1.54) DEFINITION.

A rational number $a/b \in \mathbb{Q}$ consists of a numerator $a \in \mathbb{Z}$ and a denominator $b \in \mathbb{N}$.
If $a, c \in \mathbb{Z}$ and $b, d \in \mathbb{N}$. Then a/b and c/d are considered equal i.e.,

$$\frac{a}{b} = \frac{c}{d}$$

if and only if $ad = bc$.

(1.55) EXAMPLE.

So there are many different ways of representing a rational number, such as

$$\frac{3}{7} = \frac{6}{14} = \frac{9}{21}.$$

Here

$$\frac{3}{7} = \frac{9}{21} \qquad \text{since} \qquad 3 \cdot 21 = 7 \cdot 9.$$

In fact, a fraction stays the same when its numerator and denominator are multiplied by the same natural number. ♠

I will assume that you know how to add and multiply fractions, and that you **do not make mistakes like**

$$\frac{1}{2} + \frac{2}{3} = \frac{1+2}{2+3} = \frac{3}{5}. \tag{1.14}$$

You can see now that this has to be wrong! In fact, according to Definition 1.54 for this addition to make sense, it must satisfy

$$\frac{1}{2} + \frac{2}{3} = \frac{2}{4} + \frac{2}{3} = \frac{4}{7}.$$

This is nonsense, since $\frac{3}{5}$ is not equal to $\frac{4}{7}$ according to Definition 1.54 as $3 \cdot 7 \neq 4 \cdot 5$.

In fact, if you temporarily forgot how to add fractions, you can use the wisdom in Definition 1.54. You can replace $\frac{1}{2}$ by $\frac{3}{6}$ and $\frac{2}{3}$ by $\frac{4}{6}$ and then add the numerators as in

$$\frac{1}{2} + \frac{2}{3} = \frac{3}{6} + \frac{4}{6} = \frac{3+4}{6} = \frac{7}{6}.$$

The computation above says that it is straightforward to add pizza slices of the same size (one sixth), but that you need to think a bit when adding one half pizza slice and two pizza slices of size one third. In general,

$$\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd} \qquad \text{and} \qquad \frac{a}{b}\frac{c}{d} = \frac{ac}{bd}.$$

(1.56) QUIZ.

quiz — not displayed in the pdf version.

♠

1.5.3 The real numbers $\mathbb{R}$

(1.57) DEFINITION.

A real number is defined by

$$d_0.d_1d_2\ldots, \tag{1.15}$$

*where $d_0 \in \mathbb{Z}$ and $d_1, d_2, \ldots$ is an **infinite** sequence of integers (digits) in $\{0,1,2,3,4,5,6,7,8,9\}$.*

Informally (1.15) represents the real number

$$d_0 + \frac{d_1}{10} + \frac{d_2}{100} + \cdots = d_0 + d_1 \cdot 10^{-1} + d_2 \cdot 10^{-2} + \cdots$$

given by an infinite set of digits as opposed to a rational number, which is given finitely by an integer (numerator) and a natural number (denominator). The format in (1.15) is the usual (floating point) output from your

pocket calculator or computer algebra system. For example,

$$\begin{aligned}
\frac{1}{3} &= 0.333\ldots \\
\frac{3}{7} &= 0.42857142857\ldots \\
\frac{3}{14} &= 0.21428571428571428571\ldots \\
\frac{22}{7} &= 3.14285714285714285714285714\ldots \\
\frac{355}{113} &= 3.14159292035398230088495575\ldots \\
\pi &= 3.14159265358979323846264338\ldots
\end{aligned}$$

The decimal expansion of $355/113$ above looks chaotic, but it eventually repeats itself after 112 digits. In fact the decimal expansion of every rational number is periodic i.e., it repeats itself from a certain point.

In the definition (1.15) of a real number, we are forced to make identifications as in the definition of a rational number. I will not go into details here, but just notice that the identification

$$0.99999\cdots = 1.000\ldots$$

is forced upon us: if

$$x = 9\cdot 10^{-1} + 9\cdot 10^{-2} + \cdots .$$

Then

$$10x = 9 + 9\cdot 10^{-1} + 9\cdot 10^{-2} + \cdots = 9 + x.$$

Therefore we must have $x = 1$.

A real number that is not rational is called irrational. There are many more irrational numbers than rational ones. Famous ones are $\sqrt{2}, \pi$ and e. The irrational number $\sqrt{2}$ is a root in the polynomial $x^2 - 2$ with integer coefficients (it is an algebraic number). The numbers e and π are not even algebraic (they are transcendental).

(1.58) EXERCISE.

In the display of a calculator you see the number -0.8. How is this number represented in the representation given in (1.15)? ♠

1.5.4 Arithmetic rules for numbers

(1.59) PROPOSITION.

Suppose that A is one of the sets $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$. Then for numbers x, y, z all in A we have

(i) $1 \cdot x = x$

(ii) $xy = yx$

(iii) $x + y = y + x$

(iv) $(x + y) + z = x + (y + z)$

(v) $(xy)z = x(yz)$

(vi) $x(y + z) = xy + xz$

If A is one of $\mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$, then $0 \in A$ and for every $x \in A$,

7. $x + 0 = x$

8. $x + y = 0$ *for some number $y \in A$.*

Finally if A is $\mathbb{Q}$ or $\mathbb{R}$ and $x \in A$ is not 0,then

9. $xz = 1$ *for some number $z \in A$.*

The number z above is called the inverse *of x and is usually denoted x^{-1}. The number y above is called the* negative *of x and is usually denoted $-x$. We will also use the symbol $-$ (minus) defined as an operation on two numbers $x, y \in A$ as*

$$x - y := x + (-y).$$

(1.60) EXERCISE.

Argue precisely that $-(x - y) = y - x$ for $x, y \in A$ using Proposition 1.59. ♠

The long list of arithmetic rules above may seem complicated at first, but they are just a formal version of what you already know, such as for example, $\pi + 0 = \pi$, $2 + y = 0$ if $y = -2$ and $3z = 1$ for $z = \frac{1}{3} = 3^{-1}$. Beware however, that the precision in Proposition 1.59 is necessary when programming a computer.

The rules (iv) and (v) are called the *associative* laws for addition and multiplication respectively. The rule (vi) is called the *distributive* law. It connects multiplication with addition.

(1.61) EXERCISE.

We know that zero times any number is zero. Deduce this from the rules in Proposition 1.59 starting with $0 + 0 = 0$. ♠

(1.62) EXERCISE.

Verify that (vi) is true for some specific non-zero numbers. Also convince yourself that WolframAlpha actually accepts space (between numbers and variables) as multiplication. ♠

(1.63) EXERCISE.

Suppose that $x, y, z \in \mathbb{Q}$ and $w = xy + xz$. It seems that computing w involves two multiplications and one addition. Multiplications are expensive operations on a computer. Is there a way of computing w with only one multiplication and one addition? ♠

(1.64) EXERCISE.

Suppose that $n \in \mathbb{N}$. Use the distributive law to show that

$$n^2 + n = n(n+1).$$

♠

1.5.5 Ordering numbers

In real life, finding the optimal solution to a problem often involves comparing numbers and finding the minimal or maximal one. To do this you need to be able to compare numbers in a precise way.

Let us be a little rigorous and introduce the (usual) ordering on our numbers with addition and multiplication using almost full blown mathematical formalities. The natural order $<$ on the natural numbers $\mathbb{N}$ is

$$1 < 2 < 3 < \cdots$$

For the numbers $\mathbb{Q}$ and $\mathbb{R}$ it is less obvious how to define an order. Mathematical simplicity comes to the rescue here. It is enough to define what (the) positive numbers are! We want (the) positive numbers to satisfy the conditions below.

(1.65) DEFINITION.

A subset A_+ of positive numbers in a set A of numbers must satisfy

(i) For every $x \in A$ one and only one of the following conditions must hold

- *(α)* $-x \in A_+$
- *(β)* $x = 0$
- *(γ)* $x \in A_+$

(ii) If $x, y \in A_+$, then $x + y \in A_+$ and $xy \in A_+$.

For a set A_+ of positive numbers in A, we define

$$x < y \iff y - x \in A_+$$

and

$$x \leq y \iff (x = y) \vee (x < y).$$

We will write $x > 0$ if $x \in A_+$ and $x < 0$ if $-x \in A_+$.

(1.66) REMARK.

Notice that we only use arithmetic operations to define orders on numbers in Definition 1.65. This is also how computers compare numbers algorithmically. Also if $A = \mathbb{Z}$, putting $A_+ = \mathbb{N}$ makes all of the conditions in Definition 1.65 hold. If you are given an integer, it is 0, positive or negative. This is the content of (i) in Definition 1.65. Also given two natural numbers, their product and sum are also natural numbers. This is the content of (ii) in Definition 1.65.

(1.67) EXERCISE.

If $x \neq 0$, why is $x^2 > 0$? ♠

(1.68) EXERCISE.

Suppose that $a, x, y \in A$, where A is a set of numbers and $<$ given by a subset of positive numbers A_+ as in Definition 1.65. Show that

(i) $(x < y) \wedge (y < z) \implies x < z$

(ii) $(a > 0) \wedge (x < y) \implies ax < ay$

(iii) $(a < 0) \wedge (x < y) \implies ay < ax$

♠

1.5.6 Ordering $\mathbb{Z}$

As we saw in Remark 1.66, the natural order on $\mathbb{Z}$ is defined by $\mathbb{Z}_+ = \mathbb{N}$, so that $x < y$ if $y - x \in \mathbb{N}$ for $x, y \in \mathbb{Z}$. This completely agrees with our preconception that

$$\cdots < -3 < -2 < -1 < 0 < 1 < 2 < \cdots \tag{1.16}$$

To be precise, writing $\cdots < -3 < -2 < -1 < 0 < 1 < 2 < \cdots$ is nonsense, since $<$ is only defined for two integers.

(1.69) EXERCISE.

How is one supposed to interpret $0 < 1 < 2$ for example? Go ahead and formulate (1.16) correctly comparing only two integers at a time. How does Python interpret $-3 < -2 < -1 < 0 < 1 < 2$? Find out using the python cell below.

python cell — not displayed in the pdf version.

What about $1 < 5 > 3 < 4$? What about $0 < 1 > 2$? ♠

(1.70) QUIZ.

orderquiz — not displayed in the pdf version.

♠

1.5.7 Ordering $\mathbb{Q}$

We define the positive rational numbers as

$$\mathbb{Q}_+ = \left\{ \frac{m}{n} \in \mathbb{Q} \middle| m > 0 \right\} = \left\{ 1, \frac{1}{2}, \frac{1}{3}, \frac{2}{3}, \frac{1}{4}, \frac{3}{4}, \dots \right\}.$$

One can check that $\mathbb{Q}_+$ satisfies the conditions in Definition 1.65. So formally we get

(1.71) PROPOSITION.

For $\frac{a}{b}, \frac{c}{d} \in \mathbb{Q}$,

$$\frac{a}{b} < \frac{c}{d} \qquad \Longleftrightarrow \qquad ad < bc \qquad (\text{in } \mathbb{Z}).$$

Proof. We must check when

$$\frac{c}{d} - \frac{a}{b} = \frac{bc - ad}{bd} \in \mathbb{Q}_+.$$

This happens precisely when the numerator $bc - ad \in \mathbb{N}$ or $bc - ad > 0$. Therefore the condition in the proposition is satisfied. □

(1.72) EXERCISE.

Suppose that $a/b \in \mathbb{Q}$.

(i) Is it true in general that

$$\frac{a}{b} \leq \frac{a+1}{b+1}?$$

(ii) Is it true in some cases?

(iii) Suppose that $n \in \mathbb{N}$. Prove that

$$1 - \frac{a+n}{b+n} = \frac{b-a}{b+n}.$$

(iv) What happens to the rational number

$$\frac{a+n}{b+n}$$

when $n \in \mathbb{N}$ grows and becomes very big?

python cell — not displayed in the pdf version.

♠

Using Definition 1.71, you can check that $\frac{2}{3} < \frac{5}{7}$, since

$$2 \cdot 7 < 3 \cdot 5.$$

An easy, but surprising, way of finding a rational number strictly between these two is adding their numerators and denominators:

$$\frac{2}{3} < \frac{2+5}{3+7} < \frac{5}{7}.$$

We wil try to explain the first inequality in mathematical general terms going through a rather formal proof consisting of five steps. These steps are given in the quiz below. Your task is to drag from the left and drop them to the right in an order, so that the proof makes sense.

After that you are supposed, on your own, to write down a precise proof of the second inequality.

(1.73) QUIZ.

orderquiz — not displayed in the pdf version.

♠

(1.74) EXERCISE.

Similarly to the quiz above, assume that

$$\frac{a}{b} < \frac{c}{d}.$$

Write down a precise argument showing that

$$\frac{a+c}{b+d} < \frac{c}{d}.$$

♠

The exercise below shows that our trick for finding rational numbers in between two given rational numbers can be made into a machine for generating all positive rational numbers!

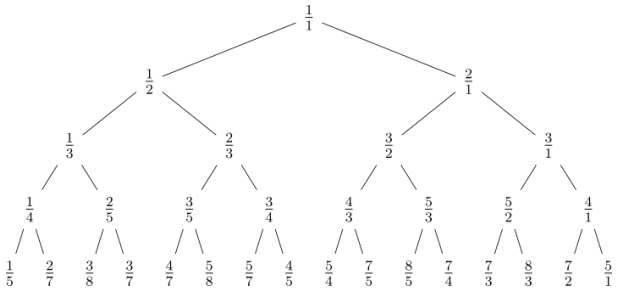
(1.75) EXERCISE.

Can you see the system in the fractions in the diagram below?

Hint: Begin by putting

$$\texttt{left} = \frac{0}{1} \quad \texttt{mediant} = \frac{1}{1} \quad \texttt{right} = \frac{1}{0}.$$

If you move left, keep `left`, put `right` equal to `mediant` and compute new `mediant` by adding fractions the naive way. If you move right keep the `right` etc.



Once you see the system, extend the diagram with the next level downwards.

Try to do binary search in a natural way in the tree for recovering the classical approximation $22/7$ for π. Here you actually need to descend to more than depth 10 in the tree! The famous approximation $355/113$ is above depth 25.

Is every positive fraction present in this diagram if one keeps adding levels?

Hint: Suppose that

$$\frac{p}{q} < \frac{r}{s}$$

and $qr - sp = 1$. Then for

$$\frac{p}{q} < \frac{p+r}{q+s} < \frac{r}{s},$$

we have $q(p+r) - (q+s)p = 1$ and $(q+s)r - (p+r)s = 1$. If $\frac{a}{b}$ is a positive fraction, such that

$$\frac{p}{q} < \frac{a}{b} < \frac{r}{s},$$

show that

$$a+b = (r+s)(qa-bp) + (p+q)(br-as) \geq p+q+r+s.$$

♠

1.5.8 Ordering $\mathbb{R}$

For the real numbers we define the positive numbers as

$$\mathbb{R}_+ = \{x \in \mathbb{R} \mid x = d_0.d_1d_2\ldots, d_0 \geq 0, x \neq 0\}.$$

Even though we have not precisely defined addition and multiplication of the real numbers, we claim that this definition of $\mathbb{R}_+$ satisfies the conditions of Definition 1.65.

One may prove that for every $x, y \in \mathbb{R}_+$, there exists $N \in \mathbb{N}$, such that $Nx > y$. This is the archimedean property of the real numbers.

(1.76) EXERCISE.

Given two distinct real numbers $\xi_1 < \xi_2$. Prove that there exists a rational number $r \in \mathbb{Q}$, such that

$$\xi_1 < r < \xi_2.$$

♠

One other crucial property is the completeness of $\mathbb{R}$. It says that a non-empty subset $S \subseteq \mathbb{R}$ with an upper bound B i.e., $\forall x \in S : x \leq B$, always has a smallest upper bound. The rational numbers do not share this property, since for example

$$S = \{x \in \mathbb{Q} \mid x^2 < 2\}$$

does not have a smallest upper bound inside $\mathbb{Q}$.

Armed with the language of logic and sets, and equipped with numbers, we now turn to the centerpiece of mathematics: proofs.

1.6 Mathematical proofs

We now have the language (logic and sets) and the objects (numbers) in place. This section is about the rules of the game: what a mathematical proof is, and the standard techniques for constructing one.

1.6.1 Proofs and inference rules

A proof begins with an assumption P and proceeds with a sequence of logical steps called inference rules leading to a conclusion Q.

You have been taught how to solve equations in steps leading to a solution. Each step turns out to be an inference rule and the conclusion is the solution. Let us see how for the simple equation $x+1=2$. Formally we want to prove

$$\forall x \in \mathbb{R} : x+1=2 \implies x=1.$$

The first inference rule is $a=b \implies a+c=b+c$ i.e., we are allowed to add the same number to both sides of an equality. This implies

$$x+1=2 \implies (x+1)-1=2-1=1.$$

To be very precise we now use Proposition 1.59 (iv) as an inference rule i.e.,

$$(x+1)-1=1 \implies x+(1-1)=1.$$

Then we use Proposition 1.598 to conclude

$$x+(1-1)=1 \implies x+0=1$$

and then finally, we get by Proposition 1.597 that

$$x+0=1 \implies x=1.$$

So to solve the equation $x+1=2$, we are actually using four (!) inference rules along the way.

1.6.2 The use of implication ($\implies$) and bi-implication ($\iff$)

As you have seen, $\implies$ and $\iff$ are applied to link propositions in a logical argument. For example,

$$x+1=2 \iff x=1 \qquad \text{or} \qquad \forall x \in \mathbb{Z} : x+1=2 \iff x=1.$$

However, for $x^2=1 \implies x=1$ we cannot link the two propositions by $\iff$, simply because $\forall x \in \mathbb{Z} : x^2=1 \implies x=1$ is false (for $x=-1$).

(1.77) EXERCISE.

Prove that

$$(x<y) \wedge ((y+3)<(z+10)) \implies (x+37)<(z+44)$$

for every $x,y,z \in \mathbb{Z}$ (see Definition 1.65 with $A=\mathbb{Z}$ for the precise definition of $<$). Write out every inference rule!

Hint: You need to be very precise here. What does $x<y$ mean precisely if $x,y \in \mathbb{Z}$? In Definition 1.65 you will see that it means that $y-x \in \mathbb{N}$. From this you need to deduce

(i) $(x<y) \wedge (y<z) \implies x<z$

(ii) $x<y \implies x+z<y+z$

for every $x,y,z \in \mathbb{Z}$ ♠

(1.78) EXERCISE.

Try out

LLM prompt — not displayed in the pdf version.

and go through the exercises given to you. ♠

1.6.3 More on mathematical proofs

Most professional mathematicians rarely think about the precise definition of a proof and would probably feel uncomfortable defining a proof precisely. During many years of training they have assimilated knowledge by experience. Therefore many proofs seem born out of witchcraft containing several magical devices.

However, many proofs appearing in respected mathematical journals, submitted by respected mathematicians, have turned out to contain errors. Recent developments in automated proof systems like *Coq* and Lean show promise in checking proofs like for example the famous four color theorem. These automated proof systems build on dependent type theory, which we will not go into.

(1.79) REMARK.

Check out The Natural Number Game at

`https://www.ma.imperial.ac.uk/~buzzard/xena/natural_number_game/index2.html`

Here you can see *Lean* in action with every step of a proof spelled out!

Informally a proof of a proposition q, consists in arguing that an implication $p \implies q$ is true by first assuming p. Usually this is done not only through one implication $p \implies q$, but through a series of intermediate implications

$$p \implies q_1 \implies q_2 \implies q_3 \implies \cdots \implies q_N,$$

where the last proposition q_N is q. If p is true, this will constitute a proof that $q_N = q$ is true. Just like in (1.16), there is an imprecision here. Can you tell what it is?

(1.80) EXAMPLE.

An integer is called even if it is divisible by 2. So the even integers are

$$\{\ldots,-4,-2,0,2,4,\ldots\}.$$

An integer is called odd if it is not even. So the odd integers are

$$\{\ldots,-5,-3,-1,1,3,\ldots\}.$$

Consider the proposition:

$$\forall n \in \mathbb{Z} : p(n) \implies p(n^2), \tag{1.17}$$

where $p(n) = (n \text{ is odd})$ i.e., the square of an odd integer is odd. This seems true for a first selection of examples: $3^2 = 9, 5^2 = 25, \ldots$.

What does it mean exactly for a number to be odd? This means that it is not divisible by 2 or that there exists another integer a, such that $n = 2a+1$. So

$$p(n) = \exists a \in \mathbb{Z} : n = 2a+1.$$

Therefore we need to show that

$$(\exists a \in \mathbb{Z} : n = 2a+1) \implies \left(\exists b \in \mathbb{Z} : n^2 = 2b+1\right).$$

Notice that I had to change a into b in the second proposition above. The two variables are not the same: a is associated with n and b is associated with n^2.

Let us assume that $n = 2a+1$. Now we need to argue that $n^2 = 2b+1$ for some $b \in \mathbb{Z}$. You stare at this for a while and notice that we should use the assumption $n = 2a+1$ in computing n^2:

$$n^2 = (2a+1)^2 = (2a)^2 + 2(2a) + 1^2 = 4a^2 + 4a + 1 = 2(2a^2 + 2a) + 1.$$

Thus, using our assumption we may conclude that if $n = 2a+1$, then

$$n^2 = 2b+1,$$

where $b = 2a^2+2a$. This completes the proof. ♠

The beauty here is that we have verified for all odd natural numbers that their square is odd. Not just a finite selection like $3, 7, 11, 13$.

In many ways a proof is like a detailed argument in a court case, except that the rules of mathematics are universal. You need the absolute truth in the court of mathematics (or science).

(1.81) EXERCISE.

Consider the proposition $q(n) = n$ is even. Prove that

$$\forall n \in \mathbb{Z} : q(n^2) \implies q(n).$$

Hint: Use that $q(n) = \neg p(n)$, where $p(n)$ is defined in Example 1.80. ♠

1.6.4 Proof by contradiction

A proposition p is either true or false. This seemingly obvious statement goes by the name of the law of excluded middle and dates back to the writings of Aristotle. The law of excluded middle is key in the example below, where you are left with the feeling that you have been deprived of a fair and genuine proof.

(1.82) EXAMPLE.

An irrational number is a (real) number that is not rational. It is a startling fact that such numbers exist, but they do! The square root $\sqrt{2}$ of two is an example. We will prove that there exists two irrational numbers α, β, such that α^β is rational.

Consider the proposition p given by

$$\gamma = \sqrt{2}^{\sqrt{2}} \quad \text{is rational.}$$

Either p is true or false. If p is true we are done putting $\alpha = \beta = \sqrt{2}$. If not, then p must be false and γ is irrational. But then

$$\gamma^{\sqrt{2}} = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = (\sqrt{2})^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$$

and we are done putting $\alpha = \gamma$ and $\beta = \sqrt{2}$.

So which one is it? Is

$$\sqrt{2}^{\sqrt{2}}$$

rational or irrational? This is really advanced mathematics based on the Gelfond-Schneider theorem. ♠

The law of excluded middle can be turned into a powerful proof technique called *proof by contradiction*.

Suppose we wish to establish that p is true. Then we turn things upside down by assuming that p is false i.e., that $\neg p$ is true. If we then by logical deduction can show that

$$\neg p \implies q,$$

for some proposition q, which is demonstrably false, then $\neg p$ cannot be true (since true $\implies$ false is false). Therefore $\neg p$ must be false and p must be true by the law of the excluded middle. This technique is used all the time!

(1.83) EXAMPLE.

Let us use proof by contradiction to show that the proposition p : ($\sqrt{2}$ is an irrational number) is true. Assuming that p is false, we must have that $\neg p$ is true. But $\neg p$ is the proposition p_1 (that $\sqrt{2}$ is a rational number)

$$\exists m,n \in \mathbb{N} : \sqrt{2} = \frac{m}{n}.$$

Here $p_1 \iff p_2$, where p_2 is the proposition

$$\exists m,n \in \mathbb{N} : \left(\sqrt{2} = \frac{m}{n}\right) \wedge ((m \text{ is odd}) \vee (n \text{ is odd})),$$

since we can assume that 2 is not a common divisor of m and n by Definition 1.54. However,

$$\sqrt{2} = \frac{m}{n} \iff \sqrt{2}n = m \iff 2n^2 = m^2 \implies m \text{ is even.}$$

The last implication above follows from Exercise 1.81. If m is even, then $m = 2k$ for some $k \in \mathbb{N}$. Therefore $m^2 = 4k^2$ and

$$2n^2 = 4k^2 \iff n^2 = 2k^2$$

so that n is also even. We have proved that p_2 implies the proposition p_3 given by

$$(m \text{ is even}) \wedge (n \text{ is even}).$$

But notice that p_3 is the negation of the second part of p_2 i.e., $p_3 \equiv \neg((m \text{ is odd}) \vee (n \text{ is odd}))$. So if p_2 were true, then p_3 would be true, since we proved that $p_2 \implies p_3$ is true. But p_3 would also be false, being the negation of a part of the true proposition p_2. This is impossible. Therefore we must have that p_2 is false and therefore that p_1 is false. But then according to the law of the excluded middle, we must have that $p = \neg p_1$ is true. ♠

(1.84) EXERCISE.

Consider the first n prime numbers

$$p_1 = 2, p_2 = 3, p_3 = 5, \ldots, p_n.$$

Check that

$$\begin{aligned} &p_1 \\ &p_1\,p_2 + 1 \\ &p_1\,p_2\,p_3 + 1 \\ &p_1\,p_2\,p_3\,p_4 + 1 \end{aligned}$$

are prime numbers by using the Python cell below (`factor` gives the prime factorization of a natural number).

python cell — not displayed in the pdf version.

Is it true in general that

$$p_1\,p_2 \cdots p_n + 1$$

is a prime number?

Assume that we know that every natural number must be divisible by a prime number. Prove that there are infinitely many prime numbers using proof by contradiction.

Hint: Show how the assumption that there are only finitely many prime numbers say

$$p_1, p_2, \ldots, p_n$$

leads to a contradiction by using that the natural number

$$p_1\, p_2 \ldots p_n + 1$$

must be divisible by a prime number. ♠

(1.85) EXERCISE.

Use proof by contradiction to show precisely that there does not exist a smallest positive rational number. ♠

1.6.5 Proof by induction

A precocious Gauss[3] proved the formula

$$1+2+\cdots+n = \frac{n(n+1)}{2} \tag{1.18}$$

at the age of seven displaying remarkable ingenuity for his age. Lesser mortals usually use induction to prove this formula. Gauss was asked along with his classmates to compute the sum of all natural numbers $1, 2, \ldots, 100$. Using his formula he quickly came up with the correct answer 5050. His classmates had to work for the entire lesson.

Suppose that the formula in (1.18) is viewed as a proposition $p(n)$. To prove the formula we need to prove it for all natural numbers (you can easily see that $p(1)$ and $p(2)$ are true) i.e., we need to prove

$$\forall n \in \mathbb{N} : p(n).$$

An induction proof is a way of proving this statement by showing two things:

(i) $p(1)$

(ii) $\forall n \in \mathbb{N} : p(n) \implies p(n+1)$

These two statements ensure that $p(1) \implies p(2)$. Therefore $p(2)$ must be true, since we assumed $p(1)$ true from the beginning. Similarly $p(2) \implies p(3)$ ensures that $p(3)$ is true and so on. In fact we have proved $p(n)$ for every $n \in \mathbb{N}$ using this technique. One can prove this using proof by contradiction and that every non-empty subset of $\mathbb{N}$ has a first element. In general if S is a subset of a set with an order $\leq$, then $s \in S$ is called a first element if

$$\forall x \in S : s \leq x.$$

A crucial rule (or axiom) is that every non-empty subset of $\mathbb{N}$ has a first element! Notice that this is false for $\mathbb{Z}$.

(1.86) THEOREM.

Suppose that $p(n)$ are infinitely many propositions given by $n \in \mathbb{N}$. Then

$$\forall n \in \mathbb{N} : p(n)$$

is true if

(i) $p(1)$ is true.

(ii) $(\forall n \in \mathbb{N} : p(n) \implies p(n+1))$ is true.

[3] See the article Gauss's Day of Reckoning for some history of this anecdote.

Proof. Suppose by contradiction that there exists $n \in \mathbb{N}$, such that $p(n)$ is false. Then the subset

$$S = \{n \in \mathbb{N} \mid \neg p(n)\} \subseteq \mathbb{N}$$

is non-empty. Therefore it has a first element $n_0 \in S$. Here $n_0 > 1$, since $p(1)$ is assumed to be true. So we know that $p(n_0 - 1)$ is true and that $p(n_0 - 1) \implies p(n_0)$ is true. But the latter implication is a contradiction, since true implies false is false. □

Let us see how an induction proof plays out in the above example with the statement $p(n)$ that

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}. \tag{1.19}$$

Clearly $p(1)$ is true. We need to prove $p(n) \implies p(n+1)$, so we assume that $p(n)$ holds i.e., that (1.19) is true. Then we may add $n+1$ to both sides of (1.19) to get

$$1 + 2 + \cdots + n + (n+1) = \frac{n(n+1)}{2} + (n+1).$$

Here the right hand side can be rewritten as

$$\frac{n(n+1) + 2(n+1)}{2} = \frac{(n+1)(n+2)}{2},$$

which is exactly what we want. This is the conjectured formula for the sum of the numbers $1, 2, \ldots, n, n+1$. Therefore we have proved that $p(n) \implies p(n+1)$ and the induction proof is complete.

(1.87) EXAMPLE.

For a real number $r \neq 1$, the extremely useful formula

$$1 + r + \cdots + r^n = \frac{1 - r^{n+1}}{1 - r} \tag{1.20}$$

holds. Let us prove this formula by induction. For $n = 1$ this amounts to the identity

$$1 + r = \frac{1 - r^2}{1 - r},$$

which is true since $1 - r^2 = (1 + r)(1 - r)$. We let $p(n)$ denote the identity in (1.20). We have seen that $p(1)$ is true. The induction step consists in proving $p(n) \implies p(n+1)$. We can prove this by adding r^{n+1} to the right hand side in (1.20):

$$\frac{1 - r^{n+1}}{1 - r} + r^{n+1} = \frac{1 - r^{n+1} + (1 - r)r^{n+1}}{1 - r} = \frac{1 - r^{n+2}}{1 - r}. \tag{1.21}$$

Real life application. In order to pay for a house you borrow P DKK at an interest of r per year. You want to pay off your debt over N years by paying a fixed amount each year. How much is the fixed yearly amount you need to pay?

Let us analyze the setup: suppose that the fixed yearly amount is Y. We will find an equation giving us Y in terms of P, N and r. Put $q = 1 + r$.

After one year you owe

$$qP - Y.$$

After two years you owe

$$q(qP - Y) - Y.$$

After three years you owe

$$q(q(qP - Y) - Y) - Y.$$

In general after n years you owe

$$q^n P - Y(1+q+\cdots+q^{n-1}).$$

Since we want to be debt free after N years, the yearly payment will have to satisfy

$$q^N P = Y(1+q+\cdots+q^{N-1}).$$

By the formula (1.20), we get

$$q^N P = Y\frac{1-q^N}{1-q}.$$

Here Y can be isolated giving the formula

$$Y = \frac{rP}{1-\left(\frac{1}{1+r}\right)^N}.$$

With an interest rate of four percent (roughly the level at the time of writing), you pay a fixed monthly amount of around 4774 DKK for borrowing one million DKK over 30 years.

python cell — not displayed in the pdf version.

♠

(1.88) EXERCISE.

Verify the computation (induction step) in (1.21) i.e., explain the operations used to go from the left to the right of the two equalities. ♠

(1.89) EXERCISE.

Locate the mistake in the following fake induction proof of the curious fact that $2^n = 2$ for every $n \in \mathbb{N}$.

Let $p(n)$ be the proposition $2^n = 2$. Then $p(1)$ is true.

We wish to prove that $p(n) \implies p(n+1)$ assuming that $p(1), \ldots, p(n)$ are true:

$$\begin{aligned} 2^{n+1} &= 2^n \cdot 2 \\ &= 2^n \cdot \frac{2^n}{2^{n-1}} \\ &= 2 \cdot \frac{2}{2} \text{ (by } p(n) \text{ and } p(n-1)) \\ &= 2. \end{aligned}$$

This shows that $p(n) \implies p(n+1)$ and therefore that $2^n = 2$ for every $n \in \mathbb{N} \setminus \{0\}$. ♠

(1.90) EXERCISE.

Prove by induction that the sum of the first n odd numbers is given by the formula

$$1+3+\cdots+(2n-1) = n^2,$$

i.e., for $n = 5$ we have

$$1+3+5+7+9 = 25.$$

♠

(1.91) EXERCISE.

Prove by induction that

$$1^2+2^2+3^2+\cdots+n^2=\frac{n(n+1)(2n+1)}{6}$$

i.e., for $n=3$, we have

$$1^2+2^2+3^2=14=\frac{3\cdot 4\cdot 7}{6}.$$

♠

(1.92) EXERCISE.

Prove by induction that

$$1^3+2^3+3^3+\cdots+n^3=\left(\frac{n(n+1)}{2}\right)^2$$

i.e., for $n=3$, we have

$$1^3+2^3+3^3=36=\left(\frac{3\cdot 4}{2}\right)^2.$$

♠

(1.93) EXERCISE.

Prove using the idea of induction that

$$2^n<n!$$

for $n\geq 4$.

♠

The last exercise related to induction concerns the famous pigeonhole principle. The statement itself looks innocent, well almost ridiculous, but it is very powerful. Even the go-to website mathoverflow for research mathematicians has a quite nice thread about this.

(1.94) EXERCISE.

Prove the following by induction on m: if n items are put into m containers and $n>m$, then at least one container must contain more than one item. ♠

1.7 The concept of a function

A function is a crucial concept in mathematics. In Python a simple function can be programmed like

python cell — not displayed in the pdf version.

The code above seems to take a number and returns the number plus one. This (f) is in fact a function taking as *input* a number and returning as *output* the number plus one. Notice that we do not even know which numbers we are talking about here. In mathematics we need to have a more precise notion of a function.

The above python function could more formally be denoted as $f : \mathbb{Z} \to \mathbb{Z}$ with $f(n) = n+1$ if we are dealing with the integers, but we cannot tell from the code.

> **Well, to be fair ....** To be completely fair, it is possible from Python 3.5 to add type annotations to functions, so that we could write
>
> ```
> def f(n: int) -> int: return(n+1)
> ```
>
> in the Python code to state that the function should take values in the integers and return integers.

The precise mathematical definition of a function in terms of sets is the following. A function $f : S \to T$ is a subset $f \subseteq S \times T$, such that $(s,t_1) \in f \wedge (s,t_2) \in f \implies t_1 = t_2$. In words it states that a function $f : S \to T$ is a subset f of $S \times T$, containing pairs having only one second coordinate for every first coordinate.

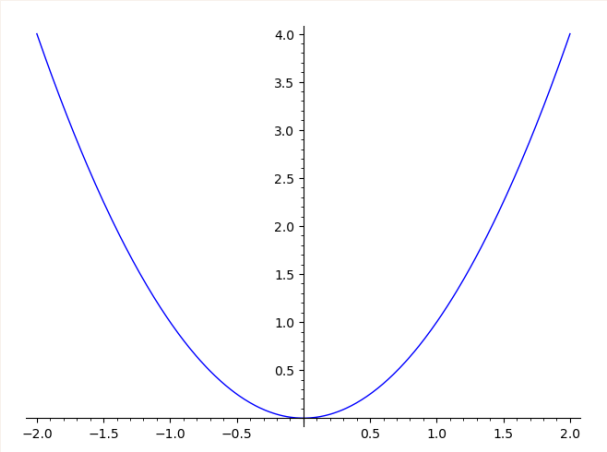
The everyday working definition of a function is more intuitive: a machine taking input from some set S and giving output in some set T. The uniqueness of the output is encoded in the mathematical definition of a function.

(1.95) DEFINITION.

Mathematically a function f takes values from a set S and returns values in a set T. In details, it is denoted $f : S \to T$ and the value associated with $s \in S$ is denoted $f(s) \in T$. Here S is called the domain *of f and T is called* the codomain *of f. Less, formally S is called the input set and T the output set for f.*

(1.96) REMARK.

Please notice that a function is a very, very general concept. It is not just something that you draw as a graph on a piece of paper. Of course, you can draw a function $f : \mathbb{R} \to \mathbb{R}$ like $f(x) = x^2$:



Generally, a function $f : S \to T$ is given by a machine, formula or algorithm that computes $f(x) \in T$ for every $x \in S$. Nothing more, nothing less. It really has nothing to do with a graph (even though graphs can sometimes be useful for visualizing certain functions like $f(x) = x^2$).

(1.97) EXAMPLE.

Good examples of functions can be found in the cryptographic hash functions. They are examples of complicated functions $f : S \to T$, where S is infinite and T finite. Here S could be data like plain text files and T could be a 256 bit number. This is the setup for the widely used `sha-256` cryptographic hash function. The whole point of a cryptographic hash function is that it must be humanly impossible to compute y with $f(y) = f(x)$ given $f(x)$[4]. In fact, `sha-256` is used in the Bitcoin block chain. The precise definition of `sha-256` can be found in FIPS PUB 180-4 approved by the Secretary of Commerce.

Other interesting functions output a bounded size digital footprint (checksum) of a file (like `md5`). This is very useful for checking data integrity of downloads over the internet. The `md5` hash is a 128 bit number.

Instead of listing 256 or 128 bits for the hash value one uses hexadecimal notation with digits in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 , a, b, c, d, e, f. A pair of hexadecimal digits then represents a byte or 8 bits. Output from `sha-256` and `md5` consist of 64 and 32 hexadecimal digits respectively. You are welcome to experiment with these two hash functions in the Python cell below.

python cell — not displayed in the pdf version.

♠

(1.98) EXERCISE.

What is the `sha-256` hash of your name? Change a few letters and recompute. Do you see any system? What about the `md5` hash function? Can you find two different strings with the same `md5` hash using your computer?

Hint: I have not answered the last question myself, but I am told that it is possible to find a collision for `md5` using a garden variety home computer. Browsing the internet, it seems that the two strings s_1 and s_2 given in hexadecimal notation[5] by

[4] A pair $x \neq y$ with $f(x) = f(y)$ is called a collision

[5] This notation represents a sequence of bytes given by pairs of hexadecimal digits

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

and

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

give a collision for `md5`. Verify that $s_1 \neq s_2$ and that they give the same `md5` hash. If you find a collision for `sha-256` you will become world famous.

Hint:

python cell — not displayed in the pdf version.

♠

1.7.1 When are two functions the same?

Suppose that $f(x) = x+1$. This way of defining a function is a bit sloppy. The domain and codomain is not defined. The correct way of defining a function also includes defining its domain and codomain as in Definition 1.95. Two functions f, g are the same when they have the same domain S and codomain T and $f(x) = g(x)$ for every $x \in S$.

(1.99) EXAMPLE.

The functions $f_1 : \mathbb{R} \to \mathbb{R}$ and $f_2 : \{x \in \mathbb{R} \mid x \geq 0\} \to \mathbb{R}$ given by $f_1(x) = x^2$ and $f_2(x) = x^2$ are not the same! Their domains are different. ♠

1.7.2 Notations for defining a function

If $f : S \to T$ is a function and S is a finite set, then you can define f using a simple table. This is best illustrated using an example. Suppose that $S = \{1,2,3\}, T = \mathbb{R}$ and

$$\begin{aligned} f(1) &= \sqrt{2} \\ f(2) &= \pi \\ f(3) &= -1. \end{aligned}$$

Then f is expressed in table form as

x	1	2	3
$f(x)$	$\sqrt{2}$	π	-1

This is a way of writing

$$f = \{(1,\sqrt{2}),(2,\pi),(3,-1)\} \subseteq \{1,2,3\} \times \mathbb{R}.$$

Very often the bracket (or *Tuborg* in Danish) notation is used. It is similar to `if-then-else` statements in programming:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2 & \text{if } x > 0 \end{cases} \tag{1.22}$$

defines the function $f : \mathbb{R} \to \mathbb{R}$ that outputs 0 if the input $x \leq 0$ and x^2 if $x > 0$. In python we may express this as

python cell — not displayed in the pdf version.

(1.100) EXERCISE.

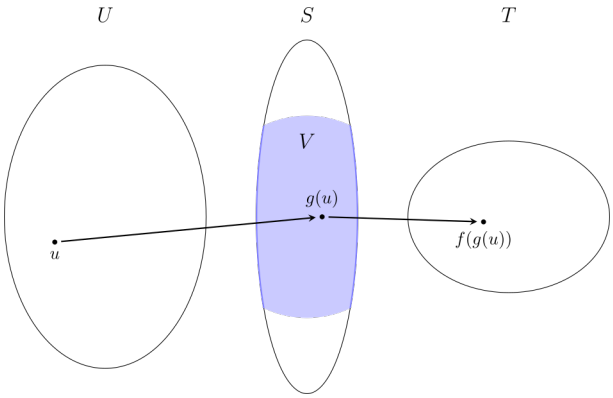
What is $f(-17)$ and $f(17)$ for the function defined in (1.22). Draw the graph of f. Come up with a function $f : S \to T$, where it does not make sense to draw a graph. ♠

1.7.3 Composition of functions

Given two functions $f : S \to T$ and $g : U \to V$, where $V \subseteq S$, we define a new function $f \circ g : U \to T$ by

$$(f \circ g)(u) = f(g(u)).$$

This notion calls for some reflection. We have a total of four sets in this definition: U, V, S and T and, not to forget, the condition that $V \subseteq S$. If this last condition was not satisfied it would be meaningless to apply the function f to $g(u)$. I hope the diagram below helps the understanding.



(1.101) REMARK.

The concept of a function is powerful and underlies functional programming in computer science: every computation can be realized as applying a composition of functions to an argument. This is exemplified in the computer language Haskell.

(1.102) EXERCISE.

Suppose that

$$U = \{1,2,3\}, \qquad S = \{1,2,3,4\} \qquad \text{and} \qquad T = \{7,8,9\}$$

and that $g : U \to S$ and $f : S \to T$ are given by the tables

x	1	2	3
$g(x)$	1	3	4

and

x	1	2	3	4
$f(x)$	7	8	9	7

Compute the table for $(f \circ g) : U \to T$. Show that $f \circ g$ is not injective (see Definition 1.106 below). Adjust the table for f so that $f \circ g$ becomes bijective. ♠

(1.103) EXERCISE.

Consider $f : \mathbb{R} \to \mathbb{R}^2$ and $g : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(t) = (t^2, t^3)$$
$$g((x,y)) = \cos(xy) + x\sin(x+y).$$

What is $(g \circ f)(t)$ as a function from $\mathbb{R}$ to $\mathbb{R}$ in terms of t? ♠

1.7.4 Functions from and into products

(1.104) DEFINITION.

Suppose that

$$B = A_1 \times A_2 \times \cdots \times A_n$$

as in (1.9). *Then the function* $\pi_i : B \to A_i$ *given by*

$$\pi_i(a_1, \ldots, a_i, \ldots, a_n) = a_i$$

is called the projection on the i-th coordinate.
If $f : A \to B$ *is a function, then*

$$f(a) = (f_1(a), \ldots, f_n(a)),$$

where $f_i = \pi_i \circ f$.

(1.105) EXAMPLE.

Suppose that $B = \mathbb{R} \times \mathbb{R} = \mathbb{R}^2$ and $(x,y) \in B$. Then

$$\pi_1((1,2)) = 1$$
$$\pi_2((1,2)) = 2.$$

Now suppose that $A = \mathbb{R}^3$ and that $f : A \to \mathbb{R}^3$ is given by

$$f((x,y,z)) = (x^2 + y, xy, z^3).$$

Then

$$f_1((1,2,3)) = 3$$
$$f_2((1,2,3)) = 2$$
$$f_3((1,2,3)) = 27.$$

♠

1.7.5 Injective and surjective functions

We now define three very important notions related to functions.

(1.106) DEFINITION.

Let $f : S \to T$ be a function. Then f is called

(i) injective, *if $f(x) = f(y) \implies x = y$ for every $x, y \in S$.*

(ii) surjective, *if for every $y \in T$, there exists $x \in S$, such that $f(x) = y$.*

(iii) bijective, *if it is both injective and surjective.*

(1.107) EXERCISE.

Is a cryptographic hash-function as defined in Example 1.97 injective? ♠

(1.108) EXERCISE.

Suppose that

$$S = \{1,2,3\} \qquad \text{and} \qquad T = \{1,2,3,4\}$$

and that the function $f : S \to T$ is defined by the table

x	1	2	3
$f(x)$	1	2	4

Is f injective? Is it surjective? Is it possible to adjust the table so that f becomes injective? Is it possible to adjust the table so that f becomes surjective? ♠

(1.109) EXERCISE.

Consider the function $f : S \to T$ given by

$$f(x) = x^2,$$

where $S = T = \mathbb{R}$. Is f injective? Is f surjective? Suggest how to change S and T so that $f : S \to T$ becomes bijective. ♠

(1.110) EXERCISE.

Consider the function $f : \mathbb{Z} \to \mathbb{Z}$ given by

$$f(x) = x + 1$$

Show that f is bijective. ♠

(1.111) EXERCISE.

Write down precisely how the truth table for $p \implies q$ may be expressed in terms of a function $f : S \to T$. What are the sets S and T in this case? ♠

1.7.6 The inverse function

If $f : S \to T$ is bijective, then we may define a function $g : T \to S$, so that $(f \circ g)(y) = y$ for every $y \in T$ and $(g \circ f)(x) = x$ for every $x \in S$. This function is denoted f^{-1}.

How do we define $f^{-1}(y)$ for $y \in T$? Well, since f is surjective, we may find $x \in S$ so that $y = f(x)$. Now, we simply define

$$f^{-1}(y) = x. \tag{1.23}$$

We cannot have $x_1 \neq x_2$ in S with $f(x_1) = f(x_2) = y$, since f is injective. We only have one choice for x in (1.23). Therefore (1.23) really is a good and sound definition.

(1.112) EXAMPLE.

Let $f : S \to S$, where $S = \{1,2,3\}$ be given by the table

x	1	2	3
$f(x)$	3	1	2

.

Then $f^{-1}$is given by the table

x	1	2	3
$f^{-1}(x)$	2	3	1

.

♠

(1.113) EXERCISE.

What if the definition of f in Example 1.112 is changed to

x	1	2	3
$f(x)$	3	2	2

.

Does f^{-1} make sense here? ♠

(1.114) EXERCISE.

What is the inverse function of $f : \mathbb{Z} \to \mathbb{Z}$ given by $f(x) = x + 1$? What is the inverse function of $g : S \to S$, where $g(x) = \sqrt{x}$ and $S = \{x \in \mathbb{R} \mid x \geq 0\}$? ♠

1.7.7 The preimage

(1.115) DEFINITION.

Consider a function

$$f : A \to B,$$

where A and B are sets. If $C \subseteq B$, then the preimage of C under f is defined by

$$f^{-1}(C) = \{x \in A \mid f(x) \in C\}.$$

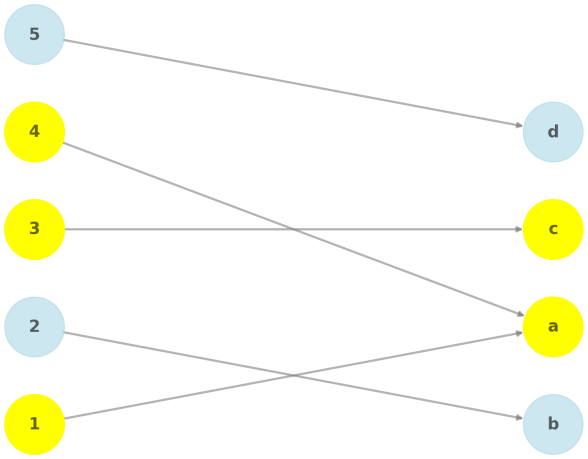
Definition 1.115 is short and sweet. Here is a first example of the preimage.

(1.116) EXAMPLE.

Consider the function $f : A \to B$, where $A = \{1,2,3,4,5\}$ and $B = \{a,b,c,d\}$ given by

x	1	2	3	4	5
$f(x)$	a	b	c	a	d

For $C = \{a,c\}$, $f^{-1}(C) = \{1,3,4\}$ as illustrated below.



♠

(1.117) EXERCISE.

What is $f^{-1}(C)$ when $A = \mathbb{R}, B = \mathbb{R}, f(x) = x^2 - 5x + 6$ and $C = (-\infty, 0]$? ♠

(1.118) QUIZ.

quiz — not displayed in the pdf version.

♠

1.7.8 Neural networks

Having defined functions and composition of functions, we can deflate the term (deep) neural network, which is often shrouded in magic and mystery.

A *neural network* is a special case of a function

$$f : A \to B, \tag{1.24}$$

where $A \subseteq \mathbb{R}^d$ and $B \subseteq \mathbb{R}^m$. Neural networks are often compositions of many intermediate functions called (hidden) layers.

(1.119) REMARK.

Recall from Definition 1.104 that a function such as (1.24) can be written

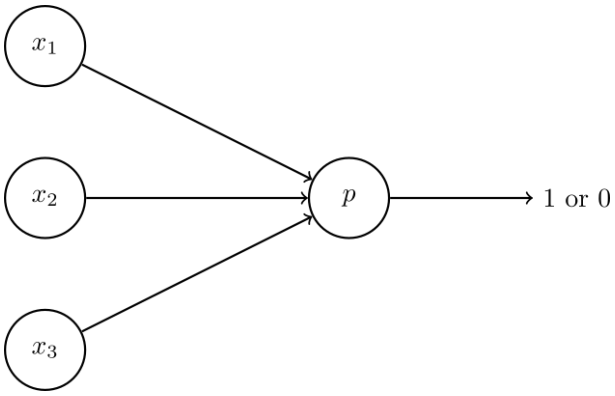
$$f(x_1,\dots,x_d) = (f_1(x_1,\dots,x_d),\dots,f_m(x_1,\dots,x_d)), \tag{1.25}$$

where $f_1,\dots,f_m$ are functions $A \to \mathbb{R}$. Check out Example 1.105.

In a neural network the functions $f_1, f_2, \dots, f_m$ are viewed as neurons[6]. Depending on their input they either fire or do not fire a signal. Classically this is modelled by the perceptron, which is a function $p : \mathbb{R}^d \to \mathbb{R}$ of the form

$$p(x_1,\dots,x_d) = \begin{cases} 1 & \text{if } w_1x_1 + \cdots + w_dx_d + b > 0 \\ 0 & \text{if } w_1x_1 + \cdots + w_dx_d + b \leq 0 \end{cases} \tag{1.26}$$

for fixed numbers $w_1,\dots,w_d$ (called weights) and a number b (called the bias). If the weighted sum $w_1x_1 + \cdots + w_dx_d$ is above the threshold $-b$, the neuron fires (returns the value 1). If not it does not fire (returns the value 0). (Some texts prefer writing the condition as $w_1x_1 + \cdots + w_dx_d > b$ and call b the threshold; the bias is then $-b$. We use the bias convention, which is the standard one in machine learning.)



(1.120) REMARK.

There is no magic hiding here. A *layer* is a function $\mathbb{R}^n \to \mathbb{R}^m$ whose coordinate functions as in (1.25) are neurons. A neural network is a composition of layers, and *deep* simply means that the composition consists of many layers. All the mystery of a deep neural network is contained in the choice of the numbers $w_1,\dots,w_d$ and b inside its neurons.

(1.121) EXAMPLE.

Perceptrons model weighted decisions of the kind you make every day. Should you bike to the university tomorrow? Suppose you weigh the chance of rain x_1 (in percent) against the time the bike saves you, x_2 (in minutes), and decide: bike exactly when

$$-x_1 + 4x_2 - 20 > 0.$$

This is the perceptron $p : \mathbb{R}^2 \to \mathbb{R}$ with weights $w_1 = -1$, $w_2 = 4$ and bias $b = -20$. The weights say how much each input matters and in which direction — rain counts against, saved time counts for, and one saved minute outweighs four percentage points of rain. The bias makes you demanding: with nothing speaking

[6]The weights $w_1,\dots,w_d$ below play the role of the strengths of the synapses feeding into a neuron.

for the bike you stay on the bus, since $p(0,0) = 0$. A 40% chance of rain and 20 saved minutes? Then $-40+80-20 = 20 > 0$: the neuron fires and you bike. The art is clearly in choosing the weights and the bias. Choosing them automatically from data is called *training* — the subject of Section 7.9. ♠

(1.122) EXERCISE.

Give weights w_1, w_2 and a bias b for a perceptron $p : \mathbb{R}^2 \to \mathbb{R}$ that computes the logical and function $\wedge$ i.e, p must satisfy

$$\begin{aligned} p(0,0) &= 0 \\ p(1,0) &= 0 \\ p(0,1) &= 0 \\ p(1,1) &= 1. \end{aligned}$$

Do the same for the logical or function $\vee$. ♠

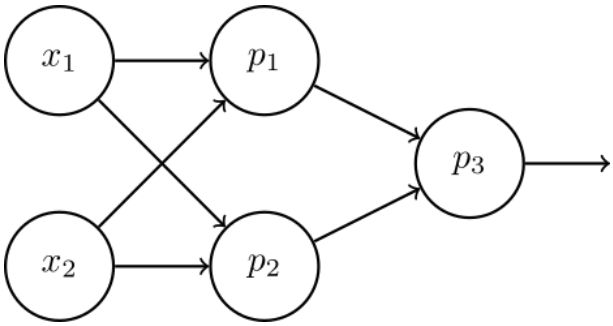
Not every simple rule is within reach of a single neuron, though. The next exercise looks innocent — it is one of the most famous problems in the history of artificial intelligence.

(1.123) EXERCISE.

Is it possible to find a perceptron $p : \mathbb{R}^2 \to \mathbb{R}$, such that

$$\begin{aligned} p(0,0) &= 0 \\ p(1,0) &= 1 \\ p(0,1) &= 1 \\ p(1,1) &= 0? \end{aligned}$$

What if you are allowed to use a neural network composed as $\mathbb{R}^2 \to \mathbb{R}^2 \to \mathbb{R}$ (one hidden layer) as sketched below? Try to build one before reading on — the next exercise walks through a construction.



Hint. A perceptron fires precisely on one side of a line in the plane. Draw the four points $(0,0), (1,0), (0,1)$ and $(1,1)$ and try to place a line with the two points where p must return 1 strictly on one side and the two points where p must return 0 on the other side. Is that possible?

♠

The function in Exercise 1.123 is the logical *exclusive or* (XOR): it returns 1 exactly when one, but not both, of its inputs is 1. It is also denoted $x \oplus y$ for input x, y.

(1.124) REMARK (A bit of history).

The perceptron was introduced by Frank Rosenblatt in 1958 and set off enormous enthusiasm — The New York Times reported it as "the embryo of an electronic computer that will be able to walk, talk, see, write, reproduce itself and be conscious of its existence". In 1969 Marvin Minsky and Seymour Papert published a famous analysis of what perceptrons cannot do, and their key example is exactly Exercise 1.123: a single perceptron cannot compute XOR. Enthusiasm and funding evaporated in what became known as an AI winter. The remedy — hidden layers, as in the exercise, combined with an efficient method for computing weights and biases from data (Section 7.9) — eventually brought neural networks back and led to the deep networks behind modern machine learning.

So a single perceptron cannot compute XOR — and history shows how costly that observation was. The rescue is composition, the very idea this chapter has been building. Here it is in slow motion.

(1.125) EXERCISE.

Consider the three perceptrons $p_1, p_2, p_3 : \mathbb{R}^2 \to \mathbb{R}$, where

$$p_1(x,y) = \begin{cases} 1 & \text{if } -x-y+\frac{3}{2} > 0 \\ 0 & \text{if } -x-y+\frac{3}{2} \leq 0 \end{cases}, \qquad p_2(x,y) = \begin{cases} 1 & \text{if } x+y-\frac{1}{2} > 0 \\ 0 & \text{if } x+y-\frac{1}{2} \leq 0 \end{cases},$$

and

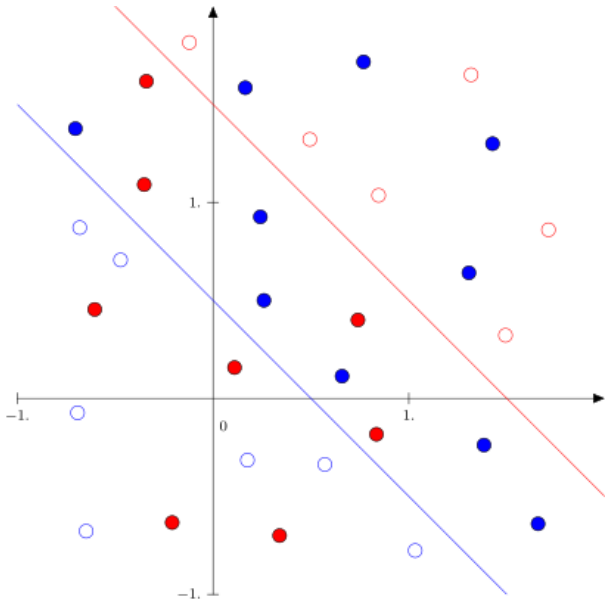
$$p_3(x,y) = \begin{cases} 1 & \text{if } x+y-\frac{3}{2} > 0 \\ 0 & \text{if } x+y-\frac{3}{2} \leq 0 \end{cases}.$$

Let $f(x,y) = p_3(p_1(x,y), p_2(x,y))$. Then f is a composite function $f = g \circ h$ of two functions $h : \mathbb{R}^2 \to \mathbb{R}^2$ and $g : \mathbb{R}^2 \to \mathbb{R}$. Write down these functions.

> **Hint.** Have a closer look at (1.25) in order to understand how functions from $\mathbb{R}^2$ to $\mathbb{R}^2$ are expressed. Notice that our notation is a bit inconsistent when it comes to types. For example, the function $p_1 : \mathbb{R}^2 \to \mathbb{R}$ should really be denoted $p_1((x,y))$ instead of $p_1(x,y)$, since it takes input from $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$. This is remedied in the (hopefully easy to understand) python code below.
>
> *python cell — not displayed in the pdf version.*
>
> *LLM prompt — not displayed in the pdf version.*

Compute $f(0,0), f(1,0), f(0,1)$ and $f(1,1)$. Compare with Exercise 1.123 — what have you just built?

Relate the perceptrons p_1 and p_2 to the illustration below. What do you think the red and blue line illustrate? What does it mean that a dot is solid compared to hollow? What is special about points between the red and blue lines? Try to relate $f(0,0), f(1,0), f(0,1)$ and $f(1,1)$ to the illustration.



(Illustration courtesy of William Heyman Krill).

♠

The point of the hidden layer. If you solved Exercise 1.125, you saw that $f(x,y) = 1$ precisely when

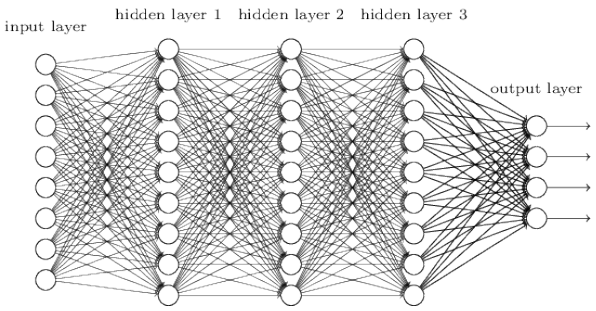
$$\frac{1}{2} < x + y < \frac{3}{2},$$

i.e., on the strip between the two lines in the illustration. This is something no single perceptron can do: a perceptron can only split the plane in two along one line. With a hidden layer, the neurons p_1 and p_2 each draw a line, and the output neuron p_3 combines the two half planes (p_3 fires exactly when both p_1 and p_2 fire). Composing layers builds complicated regions out of simple ones — that is what depth buys. In particular f answers the second question of Exercise 1.123: this little network computes XOR, which no single perceptron can.

You can experiment below. The green region shows where $f(x,y) = 1$. Try changing the weights and biases and rerun.

python cell — not displayed in the pdf version.

The output of one neuron can be used as input for other neurons in a potentially extremely complicated network:



The diagram above represents a neural network, which is a function $\mathbb{R}^8 \to \mathbb{R}^4$. This function is actually a composition (represented by the hidden layers 1, 2, 3 and the output layer):

$$\mathbb{R}^8 \to \mathbb{R}^9 \to \mathbb{R}^9 \to \mathbb{R}^9 \to \mathbb{R}^4.$$

All of the nodes above, except the ones in the input layer, represent perceptrons.

(1.126) EXERCISE.

A perceptron $p : \mathbb{R}^d \to \mathbb{R}$ as in (1.26) is specified by $d+1$ numbers: the weights $w_1, \ldots, w_d$ and the bias b. These numbers are called the *parameters* of the neuron. How many parameters does the neural network in the diagram above have in total?

> **Hint.** Count layer by layer. Each neuron in hidden layer 1 takes 8 inputs and therefore has $8+1=9$ parameters, and hidden layer 1 consists of 9 neurons.

♠

(1.127) REMARK.

Modern large language models are built in essentially the same way. When you read that a language model has 175 billion parameters, you now know exactly what is being counted: the weights and biases of its neurons. The diagrams just get (much) bigger.

Mathematically there is no reason to insist on the step function in (1.26). The general recipe behind a neuron is: form the weighted sum $w_1x_1 + \cdots + w_dx_d + b$ and feed the result into a fixed function $\mathbb{R} \to \mathbb{R}$ called the *activation function*. The perceptron uses the step function, which returns 1 for positive input and 0 otherwise. A smooth alternative is the sigmoid function, giving neurons of the form

$$\sigma(x_1, \ldots, x_d) = \frac{1}{1 + e^{-(w_1x_1 + \cdots + w_dx_d + b)}}.$$

Here $e \approx 2.718$ is the base of the exponential function, which you may recognize from high school — we will study it properly later in the book; for now the plot below tells you everything you need. Instead of jumping from 0 to 1, a sigmoid neuron glides smoothly between the two values. This matters: training a network (see below) relies on small changes of the weights and biases producing small changes of the output, and the sudden jump of the step function ruins exactly that. The sigmoid function plays the starring role in logistic regression (Section 7.8). Around 2011 it was discovered that an even simpler activation function, the rectified linear unit (ReLU) given by

$$\mathrm{ReLU}(x) = \max(0, x),$$

often works better in deep networks, and it is the default choice in most modern networks, including large language models.

python cell — not displayed in the pdf version.

(1.128) QUIZ.

quiz — not displayed in the pdf version.

♠

You may rightfully ask where the weights and biases of a neural network come from. They are not chosen by a clever human. They are computed by solving an optimization problem: one searches for the parameters that make the network reproduce a large collection of known examples as well as possible. This process is called *training*, and making sense of it is one of the main goals of this book. The story unfolds gradually over the coming chapters and culminates in Section 7.9, where you will train a genuine neural network yourself.

(1.129) REMARK (A chatbot is a function).

What does any of this have to do with the chatbot from Section 1.1? Everything. Strip away the chat window and a large language model is a function

$$f : A \to B,$$

where the input is the conversation so far, encoded as a vector, and the output is a list of numbers: one probability for each symbol the model could write next. The chatbot writes by *composing f with itself*: compute the probabilities, pick a next symbol accordingly, append it to the conversation and feed the longer text back into f — again and again, one symbol at a time, until the reply is finished. There is no module for "knowing things" and no module for "writing" — just a function, applied repeatedly, whose billions of weights and biases were computed by training. In Section 7.9.5 you will train a miniature version of f yourself and watch it invent Danish first names.

It is remarkable that the simple building blocks of this section suffice: a classical theorem, the universal approximation theorem, says that neural networks with just one hidden layer can approximate essentially any reasonable function as accurately as desired. Making "reasonable" precise requires the notion of a continuous function, which awaits you in Definition 5.65.

LLM prompt — not displayed in the pdf version.

2 LINEAR EQUATIONS

Modern mathematical terminology may seem abstract, but a lot of it comes from equation solving. We will talk about linear equations in this chapter to motivate the concept of matrices in the next chapter.

Linear equations are equations, where the unknowns only appear to the first power. For example, $x^2+x+1=0$ is not a linear equation in the unknown x, since x to the second power (x^2) appears in the equation, whereas $2x-3=1$ is. We may also consider several linear equations with several unknowns, such as

$$\begin{aligned} x+y+z &= 3 \\ x-y+z &= 1 \\ x+y-z &= 1 \end{aligned} \tag{2.1}$$

consisting of three linear equations with the three unknowns x, y and z. To be completely precise, a solution to (2.1) is a triple $(x,y,z) \in \mathbb{R}^3$, such that the predicate

$$(x+y+z=3) \wedge (x-y+z=1) \wedge (x+y-z=1)$$

is true.

(2.1) EXERCISE.

Try to come up with a solution to (2.1) i.e., find numbers x,y,z satisfying all three equations. Do not use a computer. Is there more than one solution?

Write down two linear equations with two unknowns, which do not have a solution. ♠

Do the exercise above before you evaluate the python code below, which uses the dreaded solve function from sympy. The solve function should always be used as a last resort.

python cell — not displayed in the pdf version.

2.1 One linear equation with one unknown

Very simple rules apply when solving linear equations.

Consider as an example the linear equation $2x-3=1$ in the unknown x. Solving this equation amounts to reducing to an expression $x =$ a number. This is called *isolating* x. The process is very mechanical:

$$\begin{aligned} 2x-3 &= 1 \\ &\Updownarrow \\ 2x-3+3 &= 1+3 \\ &\Updownarrow \\ 2x &= 4 \\ &\Updownarrow \\ \left(\frac{1}{2}\right)2x &= \left(\frac{1}{2}\right)4 \\ &\Updownarrow \\ x &= 2 \end{aligned}$$

If you look closely, you will see that we have used the rules

$$\begin{aligned} a = b \quad &\Longleftrightarrow \quad a + c = b + c \\ a = b \quad &\Longleftrightarrow \quad ta = tb, \end{aligned}$$

where a, b, c are numbers and t is a number $\neq 0$.

(2.2) EXERCISE.

Point out the mistake(s) in the argument[1] below showing that $2 = 1$.

$$\begin{aligned} a = b &\Longleftrightarrow \\ a^2 = ab &\Longleftrightarrow \\ a^2 - b^2 = ab - b^2 &\Longleftrightarrow \\ (a+b)(a-b) = b(a-b) &\Longleftrightarrow \\ a + b = b &\Longleftrightarrow \\ 2b = b &\Longleftrightarrow \\ 2 = 1. & \end{aligned}$$

♠

(2.3) QUIZ.

quiz — not displayed in the pdf version.

♠

(2.4) EXERCISE.

Diophantus's youth lasted $1/6$ of his life. He grew a beard after $1/12$ more. After $1/7$ more he got married. Five years later he had a son. The son lived half as long as the father and Diophantus died four years after the son. At what age did Diophantus die?

> **Link/Hint.** You can read about Diophantus and the solution to the puzzle in the Wikipedia entry about him. Please try solving the problem on your own first.

♠

2.2 Several linear equations with several unknowns

The linear equation $2x - 3 = 1$ has only one unknown with the unique solution $x = 2$. *If one linear equation has more than one unknown, then it has infinitely many solutions*. Consider as an example the linear equation $2x - 3y = 1$ with the unknowns x and y. Using the procedure as before, we get

$$\begin{aligned} 2x - 3y &= 1 \\ &\Updownarrow \\ x &= \frac{1}{2} + \frac{3}{2}y \end{aligned}$$

[1] This teaser was presented at the workshop for new teaching assistants, August 2020.

Here we are free to choose y in infinitely many ways giving infinitely many solutions $(x,y) \in \mathbb{R}^2$.

Several equations with several unknowns also make sense. Consider

$$\begin{aligned} x+y &= 3 \\ 2x-3y &= 1 \end{aligned} \tag{2.2}$$

Two numbers x and y form a solution $(x,y) \in \mathbb{R}^2$ if both equations are satisfied. From the example above, we know that

$$x = \frac{1}{2} + \frac{3}{2}y. \tag{2.3}$$

This can be inserted for x in the first equation and we get

$$3 = x+y = \frac{1}{2} + \frac{3}{2}y + y = \frac{1}{2} + \frac{5}{2}y.$$

Here we end up with one linear equation in one variable y. The solution is $y = 1$, which is inserted in the equation (2.3) giving $x = 2$. Therefore the solution to the equations is $(x,y) = (2,1)$.

(2.5) REMARK.

To be completely precise about these steps, let us use predicate logic. A solution to the system of equations above is a pair of numbers $(x,y) \in \mathbb{R}^2$ satisfying the predicate

$$(x+y=3) \wedge (2x-3y=1).$$

Now use the rules in Proposition 1.59 and substitution to rewrite systematically:

$$\begin{aligned} &(x+y=3) \wedge (2x-3y=1) \iff \\ &(x=3-y) \wedge (2x-3y=1) \iff \\ &(x=3-y) \wedge (2(3-y)-3y=1) \iff \\ &(x=3-y) \wedge (6-5y=1) \iff \\ &(x=3-y) \wedge (y=1) \iff \\ &(x=2) \wedge (y=1). \end{aligned}$$

Tracing back we have actually proved that

$$(x+y=3) \wedge (2x-3y=1) \iff (x=2) \wedge (y=1).$$

(2.6) EXERCISE.

Why is $x = 2 \wedge y = 1$ the only solution to the equations in (2.2)? How can we be so sure that there are no other values for x and y satisfying (2.2)? ♠

(2.7) QUIZ.

quiz — not displayed in the pdf version.

♠

2.3 Gauss elimination

When solving systems of several linear equations, it is natural to fix one of the equations, isolate an unknown and then insert in the other equations.

Let us study this procedure focusing on an example with two equations and three unknowns:

$$\begin{aligned} x+2y+z &= 8 \\ 2x+y+z &= 7 \end{aligned}$$

In the first equation we isolate $x = 8-2y-z$, which is then inserted into the second equation:

$$2(8-2y-z)+y+z = 7 \qquad \Longleftrightarrow \qquad -3y-z = -9.$$

It also makes perfect sense to multiply the first equation by 2 and subtract it from the second equation. This operation gives

$$-3y-z = -9.$$

It is not a coincidence that these two operations give the same result.

(2.8) THEOREM.

Suppose that

$$\begin{aligned} a_1x_1+a_2x_2+\cdots+a_nx_n &= c_1 \\ b_1x_1+b_2x_2+\cdots+b_nx_n &= c_2 \end{aligned}$$

are two linear equations in the unknowns $x_1,\ldots,x_n$ with $a_1 \neq 0$. The equation gotten by first isolating x_1 in the first equation and then inserting in the second equation is identical to the equation you get by adding the first equation multiplied by $-b_1/a_1$ to the second equation.

Proof. Isolating x_1 in the first equation inserted in the second equation gives the equation

$$b_1\left(\frac{c_1}{a_1}-\frac{a_2}{a_1}x_2-\cdots-\frac{a_n}{a_1}x_n\right)+b_2x_2+\cdots+b_nx_n = c_2 \tag{2.4}$$

Adding $-b_1/a_1$ multiplied to the first equation to the second equation gives

$$\left(b_2-\frac{b_1a_2}{a_1}\right)x_2+\cdots+\left(b_n-\frac{b_1a_n}{a_1}\right)x_n = c_2-\frac{b_1}{a_1}c_1 \tag{2.5}$$

Using basic arithmetic you can see that (2.4) can be rewritten to (2.5). □

Multiplying an equation by a number and then adding to another equation is easier to handle than the method of isolating and inserting. We have showed above that they produce the same result. Below is an extended example.

(2.9) EXAMPLE.

We wish to solve the system of equations

$$\begin{array}{rcrcrl} 2x & + & y & + & z & = 7 \\ x & + & 2y & + & z & = 8. \\ x & + & y & + & 2z & = 9 \end{array} \tag{2.6}$$

The first step is subtracting the third equation from the second:

$$\begin{array}{rcrcrcl} 2x & + & y & + & z & = 7 \\ x & + & 2y & + & z & = 8 \\ x & + & y & + & 2z & = 9 \end{array} \quad \Longleftrightarrow \quad \begin{array}{rcrcrcl} 2x & + & y & + & z & = 7 \\ & & y & - & z & = -1 \\ x & + & y & + & 2z & = 9 \end{array}$$

Then we multiply the third equation by 2 and subtract from the first:

$$\begin{array}{rcrcrcl} 2x & + & y & + & z & = 7 \\ & & y & - & z & = -1 \\ x & + & y & + & 2z & = 9 \end{array} \quad \Longleftrightarrow \quad \begin{array}{rcrcrcl} & - & y & - & 3z & = -11 \\ & & y & - & z & = -1 \\ x & + & y & + & 2z & = 9 \end{array}$$

Finally we add the second equation to the first:

$$\begin{array}{rcrcrcl} & - & y & - & 3z & = -11 \\ & & y & - & z & = -1 \\ x & + & y & + & 2z & = 9 \end{array} \quad \Longleftrightarrow \quad \begin{array}{rcrcrcl} & & & - & 4z & = -12 \\ & & y & - & z & = -1 \\ x & + & y & + & 2z & = 9 \end{array}$$

We have now reduced the original system of equations (2.6) to

$$\begin{array}{rcrcrcl} & & & - & 4z & = -12 \\ & & y & - & z & = -1 \\ x & + & y & + & 2z & = 9 \end{array},$$

where the first equation shows that $z = 3$. Now $z = 3$ can be inserted into the second equation, giving $y - 3 = -1$, which is solved by $y = 2$. Finally $y = 2$ and $z = 3$ are inserted into the third equations giving the equation $x + 8 = 9$, which is solved by $x = 1$.

One very important observation here is that $x = 1, y = 2$ and $z = 3$ is the only solution to (2.6). This is a logical consequence of the bi-implication arrows $\Longleftrightarrow$ throughout the above calculations.

python cell — not displayed in the pdf version.

♠

The elimination or substitution method for solving systems of linear equations is old and well known. Sir Isaac Newton described in 1720 the methods eloquently as follows.

> *And you are to know, that by each Æquation one unknown Quantity may be taken away, and consequently, when there are as many Æquations and unknown Quantities, all at length may be reduc'd into one, in which there shall be only one Quantity unknown.*

The mathematical rockstar Carl Friedrich Gauss used the method to determine the orbit for the asteroid Pallas. The mathematical analysis of the observations lead him to the famous least squares method and a system of six linear equations with six unknowns.

The method is known today by the term *Gaussian elimination* even though Gauss was not the first to introduce it. In fact it appeared already in *The Nine Chapters on the Mathematical Art*, which is an ancient Chinese mathematics book compiled over several centuries from the 10th century BCE to the 2nd century CE. This book contains several practical problems and their solutions. An example is

> There are three categories of corn. Three bundles of the first class, two of the second and one of the third make 39 measures. Two of the first, three of the second, and one of the third make 34 measures. Finally one of the first, two of the second and three of the third make 26 measures. How many measures of grain are contained in one bundle of each class?

(2.10) EXERCISE.

Solve the ancient corn problem above i.e., translate it into a system of three linear equations with three unknowns and apply Gauss elimination. ♠

(2.11) QUIZ.

quiz — not displayed in the pdf version.

♠

(2.12) QUIZ.

quiz — not displayed in the pdf version.

♠

(2.13) EXERCISE.

Find the solutions to

$$\begin{array}{rcrcrcr} x & + & 3y & + & z & = & 2 \\ -2x & - & 5y & + & 3z & = & 4. \end{array}$$

by expressing x and y in terms of z i.e., isolate x on the left hand side, such that

$$x = \dots$$
$$y = \dots,$$

where $\dots$ indicate an expression only in the unknown z. ♠

(2.14) EXERCISE.

Your enemy transmits secret codes (x_1, x_2, x_3, x_4) consisting of four integers x_1, x_2, x_3, x_4 over the internet. He does not transmit the code itself but an encrypted version (y_1, y_2, y_3, y_4) given by

$$\begin{array}{rcrcrcrcr} y_1 & = & 2x_1 & + & x_2 & + & 3x_3 & + & 4x_4 \\ y_2 & = & x_1 & + & 2x_2 & + & 3x_3 & + & 4x_4 \\ y_3 & = & 3x_1 & + & 3x_2 & + & x_3 & + & x_4 \\ y_4 & = & 4x_1 & + & 4x_2 & + & 2x_3 & + & 3x_4 \end{array}.$$

You have knowledge of the encryption method above and by listening in on a recent communication, you learn that the encryption $(15, 16, 12, 20)$ was sent. What was the original secret code before the encryption?

Extra credit. Suppose that you only know that the encryption scheme is

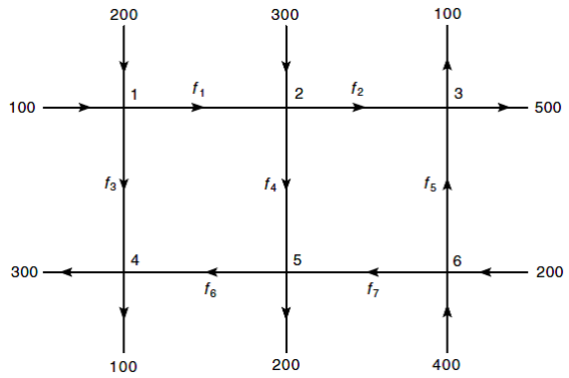
$$\begin{array}{rcrcrcrcr} y_1 & = & a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & a_{14}x_4 \\ y_2 & = & a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & a_{24}x_4 \\ y_3 & = & a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & + & a_{34}x_4 \\ y_4 & = & a_{41}x_1 & + & a_{42}x_2 & + & a_{43}x_3 & + & a_{44}x_4 \end{array},$$

and that you have no knowledge of the numbers $a_{11}, \dots, a_{44}$. How many transmissions do you need to know at the minimum to find these encryption numbers?

♠

(2.15) EXERCISE.

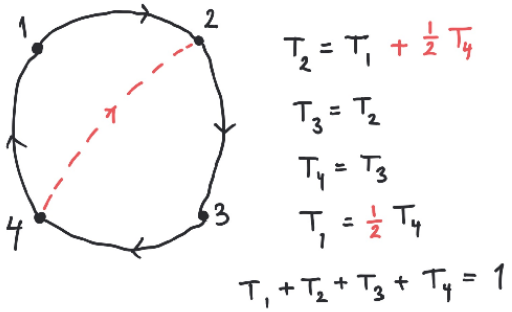
The diagram below shows a network of roads and 6 intersections. Every road is labeled by a number indicating the average number of cars per hour on the road. Some of these numbers $f_1, \ldots, f_7$ are unknowns. Write up a system of linear equations for finding $f_1, \ldots, f_7$.



Compute $f_1, f_2, f_3, f_4, f_5, f_6$ supposing that $f_1 = 200$ and $f_7 = 100$. ♠

(2.16) EXAMPLE.

This example relates to Google's famous PageRank algorithm.



Suppose we have a very simple internet with only four webpages as depicted above with arrows indicating that a webpage links to another.

We wish to study traffic in this network in the sense that we let a random websurfer jump from a given webpage to another by selecting a link randomly.

If you look at the network without the dashed red arrow, it is almost clear that a random websurfer will spend 25% of the time uniformly in each of the four nodes.

Suppose now that we introduce the dashed red arrow and let T_i denote the fraction of time the random websurfer spends at webpage i. The traffic into a webpage comes from the webpages linking to it: webpage 2 receives all the traffic from webpage 1 and half the traffic from webpage 4 (the other half goes to webpage 1). This

reasoning gives the linear equations written in the figure:

$$\begin{aligned} T_2 &= T_1 + \tfrac{1}{2}T_4 \\ T_3 &= T_2 \\ T_4 &= T_3 \\ T_1 &= \tfrac{1}{2}T_4 \\ T_1 + T_2 + T_3 + T_4 &= 1. \end{aligned} \tag{2.7}$$

Solving these shows that webpage 1 only gets $1/7 \approx 14\%$ of the time, while the other webpages get double this time each.

♠

(2.17) EXERCISE.

Solve the equations (2.7) using Gauss elimination and verify that $T_1 = 1/7$. Then decide: which webpage in the above diagram loses most traffic when a link is added from 3 to 1? ♠

(2.18) EXAMPLE.

You may try out the python code below to simulate a random tour of the small internet in Example 2.16.

python cell — not displayed in the pdf version.

The list (or matrix)

```
A = [[0,1,0,0], [0,0,1,0], [0,0,0,1], [1,1,0,0]]
```

encodes the links of the network. Python counts from zero, so node 0 is webpage 1 in the figure, node 1 is webpage 2 and so on. From `A` you can see that node 3 (webpage 4) links to nodes 0 and 1 (webpages 1 and 2) and that node 0 links to node 1. The command

```
simulate(0, 1000)
```

simulates a random surf with 1,000 clicks starting in node 0.

Compare the simulated counts with the solution to (2.7): the linear equations really do give the right result! ♠

2.4 Polynomials

Before going further into examples of linear equations we need to introduce (non-linear) functions called *polynomials*. A polynomial of degree n is a function $f : \mathbb{R} \to \mathbb{R}$ of the form

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, \tag{2.8}$$

where $a_0, \ldots, a_n$ are real numbers and $a_n \neq 0$. We call $a_0, \ldots, a_n$ the *coefficients* of f. The degree of the polynomial f is denoted $\deg(f)$. As an example,

$$x^3 - 2x + 17$$

is a polynomial of degree 3 with

$$a_3 = 1$$
$$a_2 = 0$$
$$a_1 = -2$$
$$a_0 = 17.$$

In addition to the polynomials defined in (2.8) with $a_n \neq 0$, we also view the function $f(x) = 0$ as a polynomial, called the *zero polynomial*. The zero polynomial does not[2] have a degree.

The set of all polynomials is denoted $\mathbb{R}[x]$, so that for example it makes sense to write

$$x^2 - 5x + 6 \in \mathbb{R}[x].$$

Polynomials are probably the most natural functions from $\mathbb{R}$ to $\mathbb{R}$ you can come up with. If you look at (2.8), you will see that the output is formed by using addition and multiplication (by x and selected real numbers).

You can compute with polynomials treating the *variable* x as a number obeying the rules in Proposition 1.59. For example,

$$(3x^2 + 2x + 1)(2x + 1) = 6x^3 + 7x^2 + 4x + 1.$$

In that sense polynomials obey the same arithmetic rules as numbers. A fundamental difference is that x is a placeholder or a symbol where you can insert a number from $\mathbb{R}$. In general a polynomial of degree m times a polynomial of degree n is a polynomial of degree $m+n$.

In the python cell below we let the `sympy` library do the expansion. The input format and commands for handling polynomials should be clear from the context.

python cell — not displayed in the pdf version.

You have already seen polynomials of degree one. They have the form

$$f(x) = ax + b,$$

where a and b are real numbers and $a \neq 0$. Similarly polynomials of degree two are called quadratic polynomials. They look like

$$f(x) = ax^2 + bx + c,$$

where a, b and c are real numbers and $a \neq 0$.

To get a feeling for the behavior of polynomials you should experiment in the python cell below. Try varying the degree and the coefficients of the polynomial in the plot. Also adjust the plot interval for the right view.

python cell — not displayed in the pdf version.

(2.19) EXERCISE.

Suppose that

$$f(x) = ax^2 + bx + c.$$

To compute $f(x)$ it seems that you need 3 multiplications ($a \cdot x \cdot x$ and $b \cdot x$) and 2 additions. Can you compute $f(x)$ with only 2 multiplications and 2 additions?

Try to generalize to the computation of $f(x)$, where f is a polynomial

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

of degree n (you should only need n multiplications and n additions here). ♠

[2] All its coefficients are zero!

2.4.1 Polynomial division

Division is sometimes referred to as long division when focusing on the method for division. Let us look at the situation for integers first.

The remainder of 14 divided by 4 is 2, since

$$14 = 3 \cdot 4 + 2.$$

Here the remainder 2 is strictly less than the divisor 4.

For polynomials we have a similar situation, where the degree is taken into account. For example, the remainder of $x^3 + x + 1$ divided by $x^2 + x + 1$ is $x + 2$, since

$$x^3 + x + 1 = (x - 1)(x^2 + x + 1) + (x + 2). \tag{2.9}$$

Here the degree of the remainder 1 is strictly less than the degree of the divisor 2.

The Python library `sympy` contains a wealth of functions for symbolic mathematics. In the cell below, the polynomial division (2.9) is computed with the `div` function; see the Polynomial Manipulation section of the `sympy` documentation.

python cell — not displayed in the pdf version.

The (division) algorithm for carrying out (long) division of polynomials is explained by an example in the video below.

(2.20) VIDEO.

VIDEO: https://youtu.be/SLIEUcOwVxA

(2.21) EXERCISE.

Watch the five minute video above and carry out (do not use a computer) the polynomial division alluded to in (2.9).

Also interact with a chatbot of your choice below.

LLM prompt — not displayed in the pdf version.

♠

The general result about division of polynomials is given below.

(2.22) THEOREM.

Let $d(x) \in \mathbb{R}[x]$ be a non-zero polynomial. Then for every polynomial $f(x) \in \mathbb{R}[x]$, there exists polynomials $q(x), r(x) \in \mathbb{R}[x]$, such that

$$f(x) = q(x)d(x) + r(x), \tag{2.10}$$

where $r(x) = 0$ or $\deg(r(x)) < \deg(d(x))$.

Proof. We will prove this using induction on $n = \deg(f)$. Suppose that

$$f(x) = a_n x^n + \cdots \qquad \text{and} \qquad d(x) = b_m x^m + \cdots$$

In general if $\deg(d(x)) = m > n$, then

$$f(x) = 0 \cdot d(x) + f(x)$$

satisfies the assumptions for the identity in (2.10) with $q(x) = 0$ and $r(x) = f(x)$.

If $m \leq n$, then $f(x) - a_n b_m^{-1} x^{n-m} d(x)$ is a polynomial of degree $< n$. So by induction we may find polynomials $q_0(x)$ and $r_0(x)$, such that

$$f(x) - a_n b_m^{-1} x^{n-m} d(x) = q_0(x)d(x) + r_0(x).$$

Therefore

$$f(x) = (q_0(x) + a_n b_m^{-1} x^{n-m})d(x) + r_0(x)$$

giving the desired result with $q(x) = q_0(x) + a_n b_m^{-1} x^{n-m}$ and $r(x) = r_0(x)$. □

2.4.2 Roots of polynomials

A real number $\alpha \in \mathbb{R}$ is called a *root* of the polynomial $f(x) \in \mathbb{R}[x]$ if $f(\alpha) = 0$. This is a very fundamental definition. It is mirrored beautifully in the following result.

(2.23) PROPOSITION.

A real number α is a root of the polynomial $f(x) \in \mathbb{R}[x]$ if and only if

$$f(x) = q(x)(x - \alpha),$$

for some polynomial $q(x) \in \mathbb{R}[x]$.

Proof. By Theorem 2.22, we may write

$$f(x) = q(x)(x - \alpha) + r(x), \tag{2.11}$$

where $r(x) = 0$ or $r(x)$ is a non-zero polynomial of degree zero i.e., a non-zero constant. Now the result follows, since $f(\alpha) = q(\alpha)(\alpha - \alpha) + r(\alpha) = r(\alpha)$ using (2.11). □

(2.24) EXERCISE.

Is there an easy way of deciding if a polynomial $d(x) = ax + b$ of degree one divides a polynomial $f(x)$ without performing the (long) division of $f(x)$ by $d(x)$. Here divides means that $f(x) = q(x)d(x)$ for some polynomial $q(x)$. ♠

A quadratic polynomial

$$ax^2 + bx + c$$

has at most two roots. If its *discriminant* $b^2 - 4ac$ is negative, there are no roots at all. If $b^2 - 4ac \geq 0$, the roots are given by the formula (one root for $+$ and one for $-$ in $\pm$ below; the two coincide when the discriminant is zero)

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{2.12}$$

Deriving the formula (2.12) comes from a classical algebraic trick called *completing the square*. Looking at the quadratic equation $ax^2+bx+c=0$, what bothers us is the term bx. If $b=0$ we could solve the equation rewriting to

$$x^2=-\frac{c}{a}$$

and then taking square roots. The first step in this direction is rewriting the equation

$$ax^2+bx+c=0$$

to

$$x^2+\frac{b}{a}x=-\frac{c}{a}. \tag{2.13}$$

We would like to add a number d^2 to both sides of (2.13) so that the left hand side comes to look like

$$(x+d)^2=x^2+2xd+d^2. \tag{2.14}$$

This is what is called completing the square.
Comparing the left hand side of (2.13) with the right hand side of (2.14), we find that

$$d=\frac{b}{2a}$$

works. Therefore (2.13) implies

$$\left(x+\frac{b}{2a}\right)^2=-\frac{c}{a}+\left(\frac{b}{2a}\right)^2.$$

This identity can be rewritten into the formula (2.12) for solving the quadratic equation.

For polynomials of degree three (cubic polynomials) there is a formula, but these days nobody remembers it. Also for polynomials of degree four (quartic polynomials) there is a formula. But for polynomials of degree five (quintic polynomials) and up, one can prove that a formula cannot exist!

An exceedingly important result is stated and proved below: the degree of a polynomial is an upper bound for its number of roots.

(2.25) THEOREM.

A non-zero polynomial $f(x)\in\mathbb{R}[x]$ of degree $n>0$ can have at most n roots.

Proof. We will prove this by induction starting with $n=1$. Here $f(x)=ax+b$ for $a,b\in\mathbb{R}$ with $a\neq 0$ and

$$f(\alpha)=0 \iff \alpha=-a^{-1}b.$$

Therefore $f(x)$ has precisely one root. Suppose now that we have proved that polynomials of degree n have at most n roots. Assume that $f(x)$ is a polynomial of degree $n+1$. If $f(x)$ has no roots, we are done with the proof. Suppose that $f(\alpha)=0$ i.e., α is a root in f. Then

$$f(x)=q(x)(x-\alpha)$$

by Proposition 2.23. Here $q(x)$ has to be a polynomial of degree n and therefore by induction, $q(x)$ has at most n roots. However, if $f(\beta)=q(\beta)(\beta-\alpha)=0$, then either $\beta=\alpha$ or $q(\beta)=0$. We have proved that $f(x)$ cannot have more than $n+1$ roots. □

(2.26) REMARK.

Theorem 2.25 has a few interesting consequences. First it implies that two identical polynomials i.e., $f(x) = g(x)$ for every $x \in \mathbb{R}$ must have the same coefficients.

Secondly if two polynomials $f(x)$ and $g(x)$ of degree n satisfy $f(x_i) = g(x_i)$ for distinct points $x_1, \ldots, x_{n+1}$, then $f(x) = g(x)$.

(2.27) EXERCISE.

In Remark 2.26 it is stated that if two polynomials $f(x)$ and $g(x)$ of degree n satisfy $f(x_i) = g(x_i)$ for distinct points $x_1, \ldots, x_{n+1}$, then $f(x) = g(x)$. How does this follow from Theorem 2.25? ♠

It might happen that a polynomial of degree n has precisely n roots, but it could have less or even no roots: the polynomials

$$x^2 + 1, x^4 + 1, x^6 + 1, \ldots$$

have no roots, whereas for example

$$x^2 - 2x + 1$$

is a quadratic polynomial with only one root. However polynomials of degree $1, 3, 5, \ldots$ always have at least one root.

(2.28) THEOREM.

A polynomial of odd degree always has a root.

The proof of this result is beyond our scope now and will have to wait for tools from analysis (Chapter 5).

(2.29) EXERCISE.

Compute (without using a computer!) the roots of the quartic

$$x^4 - 5x^2 + 6.$$

♠

(2.30) EXERCISE.

Give an example of a polynomial of degree 17 with precisely one root. ♠

(2.31) EXERCISE.

Suppose that α, β are two roots of the quadratic polynomial

$$f(x) = x^2 + bx + c.$$

How can b and c be computed in terms of α and β? Show concretely how this can be applied to the polynomial $g(x) = x^2 - 5x + 6$: if you know that $g(2) = 0$ how can you easily find the other root?

Hint: Show that $f(x) = (x - \alpha)(x - \beta)$ and use this. ♠

(2.32) QUIZ.

quiz — not displayed in the pdf version.

♠

2.5 Polynomials through given points

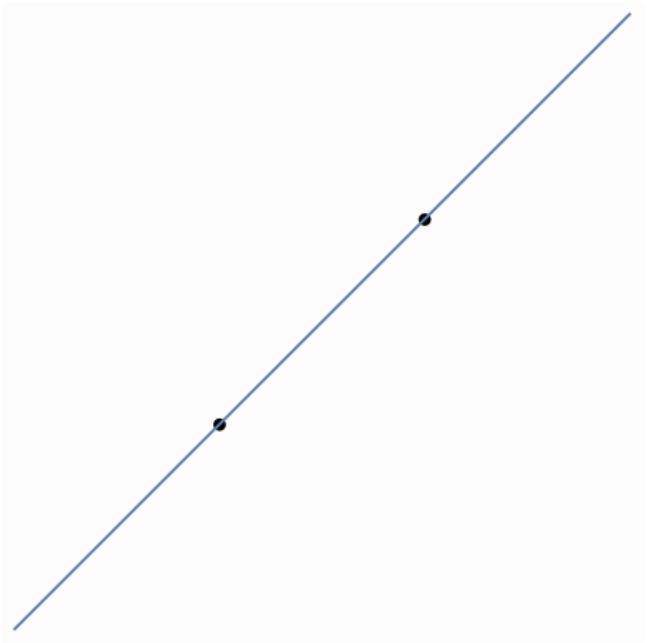
A line in the plane is given by its equation $y = ax + b$, where a is the slope and b is the intersection with the y-axis. Two lines in the plane are either parallel or intersect in a single point.

(2.33) EXERCISE.

The two lines $y = x + 1$ and $y = -x + 2$ have a single point of intersection. Compute this point.

Give an example of two parallel lines and their equations. ♠

Through two (distinct) points (x_1, y_1) and (x_2, y_2) with $x_1 \neq x_2$ passes a unique line:



You can find the equation for this line by solving two equations with two unknowns a and b:

$$x_1 a + b = y_1$$
$$x_2 a + b = y_2$$

We might as well apply Gauss elimination to solve this system. First we subtract the second equation from the first. This gives $(x_1 - x_2)a = y_1 - y_2$. Therefore

$$a = \frac{y_1 - y_2}{x_1 - x_2}.$$

Inserting this a in the first equation we get

$$b = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}.$$

We can also in a quite explicit way just write

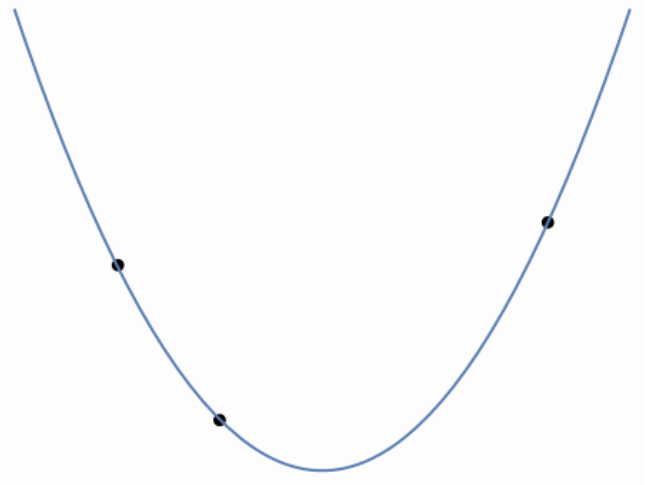
$$y = f(x) = y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x - x_1}{x_2 - x_1}. \tag{2.15}$$

The function $f(x)$ in (2.15) is a polynomial of degree one with $f(x_1) = y_1$ and $f(x_2) = y_2$.

In almost the same way we may find a unique quadratic polynomial

$$y = ax^2 + bx + c$$

through three points $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) with distinct x-values:



Here we end up with three linear equations in the unknowns a, b and c:

$$\begin{aligned} x_1^2 a + x_1 b + c &= y_1 \\ x_2^2 a + x_2 b + c &= y_2 \\ x_3^2 a + x_3 b + c &= y_3 \end{aligned} \tag{2.16}$$

It is not immediately obvious that this system of equations has a solution. But watch the following trick evolve.

We may explicitly construct the quadratic polynomial passing through the three points as

$$\begin{aligned} y = f(x) = & y_1 \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \\ & + y_3 \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} \end{aligned} \tag{2.17}$$

Take a moment and verify that $f(x_1) = y_1, f(x_2) = y_2$ and $f(x_3) = y_3$. This also proves that the system of equations in (2.16) can be solved.

(2.34) REMARK.

Notice in (2.17) that

$$y = y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x),$$

where (for example) L_1 is a polynomial of degree two satisfying

$$L_1(x_1) = 1, \quad L_1(x_2) = 0, \quad \text{and} \quad L_1(x_3) = 0.$$

What about L_2 and L_3 with respect to x_1, x_2 and x_3?

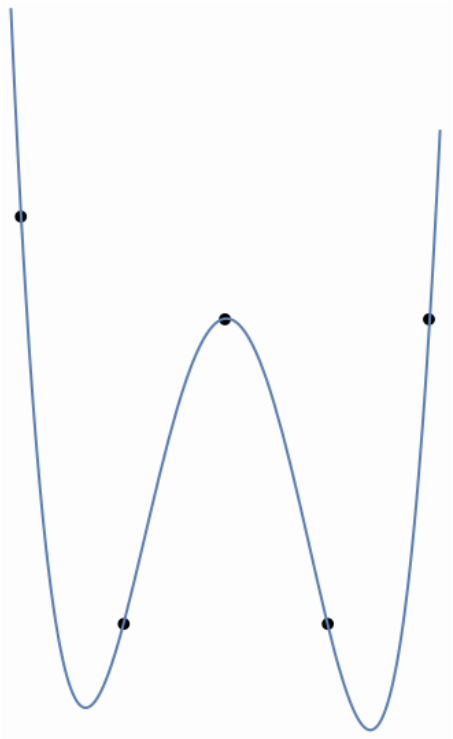
(2.35) EXERCISE.

Compute the polynomial you get when you apply (2.17) to $x_1 = 1, x_2 = 2, x_3 = 3$ and $y_1 = 1, y_2 = 2, y_3 = 3$. How do you explain this result in terms of the points $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3) plotted in plane? ♠

The natural generalization is that there exists a unique polynomial of degree $\leq n$ passing through $n+1$ points $(x_1, y_1), \ldots, (x_{n+1}, y_{n+1})$ with distinct x-values.

The rather miraculous trick above in (2.17) is called Lagrange interpolation and can be generalized to polynomials of arbitrary degree.

Below is an example of five points defining a unique polynomial of degree four.



2.5.1 The magic of Lagrange polynomials

(2.36) EXAMPLE.

Let us explain with a simple numerical example what happens in (2.17). Suppose we wish to find a polynomial $f(x) = a_0 + a_1 x + a_2 x^2$ through the points

$$(1,2), \qquad (2,3) \qquad \text{and} \qquad (3,5).$$

More precisely we wish to find numbers a_0, a_1 and a_2, such that

$$\begin{aligned} f(1) &= a_0 + a_1 + a_2 = 2 \\ f(2) &= a_0 + 2a_1 + 4a_2 = 3 \\ f(3) &= a_0 + 3a_1 + 9a_2 = 5. \end{aligned}$$

This is a system of three linear equations which in this case has a unique solution in a_0, a_1 and a_2.
We may, however, attack this problem in another way. Suppose that $L_1(x), L_2(x)$ and $L_3(x)$ are polynomials of degree at most two, such that

$$\begin{array}{lll} L_1(1) = 1 & L_1(2) = 0 & L_1(3) = 0 \\ L_2(1) = 0 & L_2(2) = 1 & L_2(3) = 0 \\ L_3(1) = 0 & L_3(2) = 0 & L_3(3) = 1 \end{array}$$

Then

$$f(x) = 2L_1(x) + 3L_2(x) + 5L_3(x)$$

really is the polynomial we wish to find. The insight is that these $L_1(x), L_2(x)$ and $L_3(x)$ can be explicitly written down as

$$\begin{aligned} L_1(x) &= \frac{(x-2)(x-3)}{(1-2)(1-3)} \\ L_2(x) &= \frac{(x-1)(x-3)}{(2-1)(2-3)} \\ L_3(x) &= \frac{(x-1)(x-2)}{(3-1)(3-2)}. \end{aligned}$$

♠

Example 2.36 can be generalized: suppose we have $n+1$ distinct numbers

$$x_1, x_2, \ldots, x_{n+1}. \tag{2.18}$$

Then these numbers give $n+1$ polynomials each of degree n:

$$L_i(x) = \frac{1}{C_i}(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_{n+1}),$$

where $C_i = (x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_{n+1})$ for $i = 1, \ldots, n+1$.

The polynomial $L_i(x)$ is called the i-th Lagrange basis polynomial associated to the $n+1$ numbers $x_1, \ldots, x_{n+1}$. It satisfies $L_i(x_1) = \cdots = L_i(x_{i-1}) = 0$, $L_i(x_i) = 1$ and $L_i(x_{i+1}) = \cdots = L_i(x_{n+1}) = 0$ i.e., $L_i(x)$ is equal to zero evaluated at all of the numbers $x_1, \ldots, x_{n+1}$ except at x_i where it evaluates to 1.

The Lagrange basis polynomials associated to $x_1,\dots,x_{n+1}$ allow us to construct a polynomial f of degree $\leq n$ through $n+1$ points $(x_1,y_1),\dots,(x_{n+1},y_{n+1})$ i.e., a polynomial f such that

$$\begin{aligned} f(x_1) &= y_1 \\ &\vdots \\ f(x_{n+1}) &= y_{n+1} \end{aligned}$$

simply as

$$f(x) = y_1 L_1(x) + \cdots + y_{n+1} L_{n+1}(x).$$

However, $f(x)$ does not have to have degree n. For example, it could come out as a line through three points $(x_1,y_1),(x_2,y_2)$ and (x_3,y_3) (see Exercise 2.35).

> **See the magic in a plot.** The cell below plots the three Lagrange basis polynomials from Example 2.36 together with the interpolating polynomial $f(x) = 2L_1(x)+3L_2(x)+5L_3(x)$. Notice how every L_i takes the value 1 at its own point x_i and 0 at the two others — this is exactly what makes the construction work. Edit the points and rerun.
>
> *python cell — not displayed in the pdf version.*

(2.37) EXERCISE.

Compute $a_0,a_1,a_2,a_3 \in \mathbb{R}$ so that

$$\begin{aligned} f(-2) &= -1 \\ f(-1) &= 1 \\ f(1) &= 1 \\ f(2) &= 1, \end{aligned}$$

where

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

You can do this either by Lagrange interpolation or by solving linear equations. Which one do you prefer? ♠

(2.38) EXAMPLE.

Can you predict the next number in the sequence starting with

$$15, \quad 34, \quad 65, \quad 111, \quad 175, \quad 260, \quad 369? \tag{2.19}$$

This question was posed[3] by the tutors in a class session for new computer science students. Let us put the sequence (2.19) inside a table like

n	1	2	3	4	5	6	7
$f(n)$	15	34	65	111	175	260	369

,

where $f:\mathbb{N}\to\mathbb{N}$ is the secret function responsible for the sequence. We would like to compute $f(8)$. Assuming that $f(n)$ is a polynomial function, we may simply compute the unique polynomial of degree ≤ 6 through the 7 points

$$(1,15), \quad (2,34), \quad (3,65), \quad (4,111), \quad (5,175), \quad (6,260), \quad (7,369).$$

[3] Thanks to Tobias Bendsen Poulsen for notifying me about this.

We know how to do this either by solving linear equations or computing with Lagrange polynomials. It turns out that `sympy` has a built in function helping us here.

python cell — not displayed in the pdf version.

Press Run to see what the next number is in the sequence (computed using the secret polynomial). See also the description of Neville's algorithm in Wikipedia for an easier approach to computing $f(8)$.

♠

2.6 Shamir secret sharing

Lagrange interpolation is used in cryptography in Shamir's secret sharing. Secret sharing is important in many practical situations. Here is an example quoted from Wikipedia:

> A company needs to secure their vault's passcode. They could encrypt it, but what if the beholder of the secret key is unavailable or turns rogue?
>
> One needs to distribute the secret. This is where SSS comes in. It can be used to encrypt the vault's passcode and generate a certain number of shares, where a certain number of shares can be allocated to each executive within the company. Now, only if they pool their shares can they unlock the vault. The threshold can be appropriately set for the number of executives, so the vault is always able to be accessed by the authorized individuals. Should a share or two fall into the wrong hands, they couldn't open the passcode unless the other executives cooperated.

The mathematics that takes care of this is surprisingly simple. Suppose the secret is the number a_0. Then we construct the polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m \tag{2.20}$$

for some other numbers $a_1, \ldots, a_m$. We know that this polynomial is uniquely given by its values in $m+1$ distinct numbers (see Remark 2.26). So if there are n trusted people we could distribute the shares

$$(1, f(1)), (2, f(2)), \ldots, (n, f(n))$$

to them. Here we suppose that $n > m$. In this setting, if there are less than $m+1$ of the people present they cannot open the vault. If $m+1$ or more people are present they can reconstruct the polynomial in (2.20), find the secret code a_0 and open the vault.

(2.39) EXERCISE.

You are in a study group consisting of four people. The professor has decided that you submit your project using a secret code that is distributed to the group members with Shamir secret sharing. At least three group members need to agree on submission.

On the day of the deadline three group members with shares

$$(1, 7035), \quad (2, 19748) \quad \text{and} \quad (3, 39373)$$

are present. What is the secret code they may use to submit their project? ♠

> **Python.** You can check your answer to the exercise above (and experiment with secrets of your own) using `interpolate`.
>
> *python cell — not displayed in the pdf version.*

(2.40) REMARK.

Real implementations of Shamir secret sharing do not use real numbers, but arithmetic modulo a large prime number. Among other things, this prevents a group of fewer than $m+1$ people from learning partial information about the secret.

2.7 Fitting data

Given a data set

$$\mathscr{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

one would often like to find a model (i.e. some function) that describes the data well. With Lagrange interpolation we can find a polynomial f fitting the sample data $\mathscr{D}$ perfectly, i.e. satisfying $f(x_i) = y_i$ for $i = 1, 2, \ldots n$. Is f an optimal model? For the given data set it seems so, but we have been a bit imprecise in formulating the goal of a model.

Actually, we are not very interested in modeling the data at hand with extreme precision. What we want is a model that fits new data well. Let us look at a concrete example.

Consider the data set

$$\mathscr{D} = \{(0, 0.06), (0.5, 0.33), (1, 0.56), (1.5, 1.35), (2, 1.48), (2.5, 1.15),$$
$$(3, 1.45), (3.5, 1.12), (4, 0.68), (4.5, 0.22), (5, -0.10)\}$$

The data points (x_i, y_i), $i = 1, 2, \ldots, 11$, were generated as $(x_i, p(x_i) + \varepsilon_i)$ where $p(x) = -0.2x^2 + x$ is a quadratic polynomial and $\varepsilon_i \in [-0.4, 0.4]$ is a random number to simulate noise. The polynomial p is the best possible model for unknown data as there will always be noise that cannot be modeled. In real life p is what needs to be modeled based on the available data.

In the figure below is a fit with a degree 2 and degree 10 polynomial respectively. As we see, the degree 2 polynomial is pretty close to the target p compared to the degree 10 polynomial that nevertheless fits the data $\mathscr{D}$ perfectly. Generally a simple model is preferred over a complex, as the latter will have a tendency to fit noise. This phenomenon is called overfitting and is an extremely important topic.

An interactive version of this illustration with a little more bells and whistles can be found here, and you can reproduce the figure — and experiment with other degrees — in the cell below.

python cell — not displayed in the pdf version.

(2.41) FIGURE.

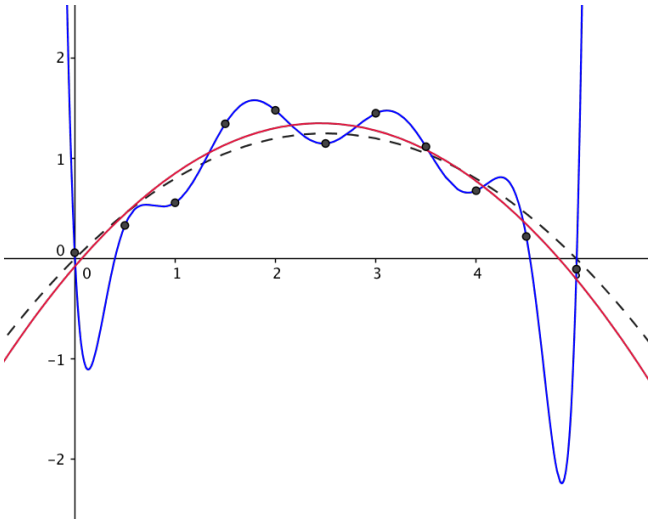


Figure 2.1: Fitting a degree 2 polynomial (in red) and a degree 10 polynomial (in blue) to the sample data $\mathscr{D}$. The target function p is the dashed curve. We see that the simple quadratic fit is much closer to the target function and hence performs better on new data.

In Section 5.4 we will see how the degree two polynomial fit was obtained, using the method of least squares. This is a nice example of a convex optimization problem.

3 MATRICES

Handling linear equations and keeping track of the unknowns can be a pain. At a certain point one needs to simplify the notation. This is done introducing matrices.

For example, the system of equations

$$\begin{array}{rcrcrcr} & & 2y & + & 4z & = & -2 \\ 3x & + & 2y & + & 7z & = & 4 \end{array} \tag{3.1}$$

can be represented by the rectangular array (matrix)

$$\begin{pmatrix} 0 & 2 & 4 & -2 \\ 3 & 2 & 7 & 4 \end{pmatrix} \tag{3.2}$$

of numbers. Many of the operations we do to solve linear equations might as well be done on this array forgetting about the unknowns.

3.1 Matrices

3.1.1 Definitions

A rectangular array of numbers is called a *matrix*. A matrix with m *rows* and n *columns* is called an $m \times n$ (m by n) matrix. The notation for an $m \times n$ matrix A is

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{pmatrix}, \tag{3.3}$$

where $A_{ij} = a_{ij}$ denotes the number or *entry* in the i-th row and j-th column. If the matrix in (3.2) is denoted A, then it has 2 rows and 4 columns with $A_{14} = -2$.

Two matrices are equal if they have the same number of rows and columns and their entries are identical.

A very useful (and famous) open source library in python (with 1000+ contributors) for handling matrices is `NumPy`. Here is how the matrix in (3.2) is entered in `NumPy`.

python cell — not displayed in the pdf version.

1. A matrix whose entries are all 0 is called a zero matrix. It is denoted simply by 0, when it is clear from the context what its numbers of rows and columns are.

2. A matrix is called *square* (in some texts: quadratic) if it has an equal number of rows and columns.

   The first two matrices below are square, whereas the third is not.

$$\begin{pmatrix} 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \qquad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

3. The *diagonal* in a matrix is defined as the entries in the matrix with the same row- and column indices.

   Below we have a 3×4 matrix with the diagonal elements marked

$$\begin{pmatrix} 1 & 3 & 0 & 1 \\ 3 & 2 & 1 & 5 \\ 1 & 0 & 3 & 6 \end{pmatrix}.$$

   A matrix is called a *diagonal matrix*, if all its entries outside the diagonal are $= 0$. Below is an example of a square diagonal matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

4. A matrix is called a *row vector* if it has only one row.

   For example,

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

   is a row vector with three columns.

5. A matrix is called a *column vector* if it has only one column.

   For example,

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

   is a column vector with three rows.

6. The rows in a matrix are called the *row vectors* of the matrix. The i-th row in a matrix A is denoted A_i.

   The matrix A in (3.2) contains the row vectors

$$A_1 = \begin{pmatrix} 0 & 2 & 4 & -2 \end{pmatrix} \qquad \text{and} \qquad A_2 = \begin{pmatrix} 3 & 2 & 7 & 4 \end{pmatrix}.$$

7. The columns in a matrix are called the *column vectors* of the matrix. The j-th column in a matrix A is denoted A^j[1].

   The matrix A in (3.2) contains the column vectors

$$A^1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \quad A^3 = \begin{pmatrix} 4 \\ 7 \end{pmatrix} \quad \text{and} \quad A^4 = \begin{pmatrix} -2 \\ 4 \end{pmatrix}.$$

8. A row- or column vector is referred to as a *vector*.

9. Even though we have used the notation $\mathbb{R}^n$ for the n-th cartesian product of $\mathbb{R}$, we will use $\mathbb{R}^n$ henceforth to denote the set of column vectors with n rows (entries). This definition is almost identical with the previous one, except that the tuple is formatted as a column vector.

   Illustrated by an example,

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \in \mathbb{R}^3 \qquad \text{instead of} \qquad (1,2,3) \in \mathbb{R}^3.$$

[1] Not to be confused with powers of the matrix A introduced later.

3.2 Linear maps

In the first chapter we encountered a miniature version of a neural network. Neural networks are generally incredibly complicated functions from $\mathbb{R}^n$ to $\mathbb{R}^m$. The function $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x^7 y + \cos(xy) e^{x^2+y^2-1} \\ 2xy^2 - \sin(x+y)(x^3+y^3) \end{pmatrix},$$

even though it looks complicated, is simple in comparison.

You probably agree that the function $g : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$g\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2x+3y \\ 3x-2y \end{pmatrix}$$

is even simpler. This function (or map) is an example of a linear map.

In general, a *linear map* $f : \mathbb{R}^n \to \mathbb{R}^m$ has the form

$$f\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \end{pmatrix},$$

where $a_{11}, \ldots, a_{mn}$ are mn real numbers.

Using matrices we will use the notation

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \end{pmatrix}.$$

In this way, we can write the map f as

$$f(v) = Av,$$

where A is the $m \times n$ matrix

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

and v is the vector

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

in $\mathbb{R}^n$.

(3.1) REMARK.

The rows of Av are weighted sums $a_{i1}x_1 + \cdots + a_{in}x_n$ — exactly the sums computed by the neurons of (1.26) in the first chapter. In fact, a layer of a neural network with n inputs and m neurons is the map $v \mapsto \sigma(Av + b)$, where A is the $m \times n$ matrix holding the weights of the m neurons, b is a vector of their biases and the activation function σ is applied to each entry. Linear maps are the computational backbone of neural networks.

Basically a linear map is a system of linear equations without the right hand sides (and the $=$ signs). In fact, we may write the system of linear equations in (3.1) as

$$\begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 7 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \end{pmatrix}.$$

(3.2) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ be the linear map given by the 2×2 matrix

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$

Does there exist $u \in \mathbb{R}^2$, such that

$$f(u) = \begin{pmatrix} 3 \\ 7 \end{pmatrix}?$$

Quite generally, can we find $u \in \mathbb{R}^2$, such that

$$f(u) = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}?$$

for arbitrary $b_1, b_2 \in \mathbb{R}$? ♠

(3.3) EXERCISE.

Suppose you know that $f : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map and that you have a black box giving you output $f(v) \in \mathbb{R}^m$ if you supply the input $v \in \mathbb{R}^n$. How would you find the matrix defining f? ♠

3.3 Matrix multiplication

Suppose we are given two linear maps $f : \mathbb{R}^2 \to \mathbb{R}^2$ and $g : \mathbb{R}^2 \to \mathbb{R}^2$. Then it turns out that the composition $f \circ g : \mathbb{R}^2 \to \mathbb{R}^2$ is also a linear map. A word of advice: the computations below look large and intimidating. They are not. It is important that you carry them out on your own. Do not look and copy or tell yourself that it looks okay. Do the computations yourself and ask me or fellow students if you get stuck.

Let us look at an example. Suppose that

$$g\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ -1 & -2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{and} \qquad f\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & -2 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}.$$

Then

$$(f \circ g)\begin{pmatrix} x \\ y \end{pmatrix} = f\left(g\begin{pmatrix} x \\ y \end{pmatrix}\right) = f\left(\begin{pmatrix} 2 & 3 \\ -1 & -2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} 1 & 2 \\ 1 & -2 \end{pmatrix}\left(\begin{pmatrix} 2 & 3 \\ -1 & -2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}\right)$$

$$= \begin{pmatrix} 1 & 2 \\ 1 & -2 \end{pmatrix}\begin{pmatrix} 2x+3y \\ -x-2y \end{pmatrix} = \begin{pmatrix} -y \\ 4x+7y \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 4 & 7 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}.$$

In terms of the matrices of the linear maps, we write this as

$$\begin{pmatrix} 1 & 2 \\ 1 & -2 \end{pmatrix}\begin{pmatrix} 2 & 3 \\ -1 & -2 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 4 & 7 \end{pmatrix} \tag{3.4}$$

There is nothing special about the numbers in this example. We might as well do the computation in general: suppose that

$$g\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{and} \qquad f\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix}.$$

Then

$$(f\circ g)\begin{pmatrix}x\\y\end{pmatrix}=f\left(g\begin{pmatrix}x\\y\end{pmatrix}\right)=f\left(\begin{pmatrix}b_{11}&b_{12}\\b_{21}&b_{22}\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}\right)=\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}\left(\begin{pmatrix}b_{11}&b_{12}\\b_{21}&b_{22}\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}\right)$$

$$=\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}\begin{pmatrix}b_{11}x+b_{12}y\\b_{21}x+b_{22}y\end{pmatrix}=\begin{pmatrix}a_{11}(b_{11}x+b_{12}y)+a_{12}(b_{21}x+b_{22}y)\\a_{21}(b_{11}x+b_{12}y)+a_{22}(b_{21}x+b_{22}y)\end{pmatrix}$$

$$=\begin{pmatrix}(a_{11}b_{11}+a_{12}b_{21})x+(a_{11}b_{12}+a_{12}b_{22})y\\(a_{21}b_{11}+a_{22}b_{21})x+(a_{21}b_{12}+a_{22}b_{22})y\end{pmatrix}$$

$$=\begin{pmatrix}a_{11}b_{11}+a_{12}b_{21}&a_{11}b_{12}+a_{12}b_{22}\\a_{21}b_{11}+a_{22}b_{21}&a_{21}b_{12}+a_{22}b_{22}\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}.$$

Again, in terms of the matrices of the linear maps, we write this as

$$\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}\begin{pmatrix}b_{11}&b_{12}\\b_{21}&b_{22}\end{pmatrix}=\begin{pmatrix}a_{11}b_{11}+a_{12}b_{21}&a_{11}b_{12}+a_{12}b_{22}\\a_{21}b_{11}+a_{22}b_{21}&a_{21}b_{12}+a_{22}b_{22}\end{pmatrix}\tag{3.5}$$

The equation above is the formula for matrix multiplication for two 2×2 matrices, precisely as it was introduced by Cayley around 1857.

Upon closer inspection (and colored in (3.5) for $i=2$ and $j=1$), you will see that the number in the i-th row and j-th column in the product matrix is the *row-column multiplication* between the i-th row and the j-th column in the two matrices:

The *row-column multiplication* between a row vector

$$x=(x_1x_2\ldots x_n)\qquad\text{and a column vector}\qquad y=\begin{pmatrix}y_1\\y_2\\\vdots\\y_n\end{pmatrix}$$

with the same number of entries is defined as

$$xy=x_1y_1+x_2y_2+\cdots+x_ny_n.$$

(3.4) DEFINITION.

Let A be an $m\times n$ matrix and B an $n\times r$ matrix. Then the matrix product AB is defined as the $m\times r$ matrix C given by the row-column multiplication

$$C_{ij}=A_iB^j=A_{i1}B_{1j}+A_{i2}B_{2j}+\cdots+A_{in}B_{nj}$$

for $1\leq i\leq m$ and $1\leq j\leq r$.

If A is an $m\times n$ matrix and B is an $r\times s$, then the matrix product AB only makes sense if $n=r$: the number of columns in A must equal the number of rows in B.

(3.5) QUIZ.

quiz — not displayed in the pdf version.

♠

(3.6) VIDEO.

I have been told that my pronunciation of column *in the video below is wrong. In the area of the US, where I got my PhD, people for some reason had this (Irish?) rare pronunciation.*

VIDEO: https://youtu.be/KVpBEynN3IM

Using matrix product notation, the system of linear equations in (3.1) can now be written as

$$\begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 7 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \end{pmatrix}$$

Here we multiply a 2×3 with a 3×1 matrix. The row-column multiplication gives the 2×1 matrix

$$\begin{pmatrix} 2y+4z \\ 3x+2y+7z \end{pmatrix}.$$

This matrix must equal the 2×1 matrix on the right hand side for (3.1) to be true. This is in agreement with our convention for writing linear maps in section 3.2.

(3.7) QUIZ.

quiz — not displayed in the pdf version.

♠

3.3.1 Matrix multiplication in `numpy`

Matrix multiplication in `numpy` is written with the `@` operator (the function `np.dot` computes the same thing; beware that `*` means entry-wise multiplication for arrays):

python cell — not displayed in the pdf version.

3.3.2 The identity matrix

The identity matrix I_n of order n is the $n \times n$ diagonal matrix with 1 in the diagonal. Below is the identity matrix of order 5.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The identity matrix I_n has the crucial property that

$$I_n A = A I_n = A \tag{3.6}$$

for all $n \times n$ matrices A.

python cell — not displayed in the pdf version.

(3.8) EXERCISE.

Prove that the two identities in (3.6) are true for $n \times n$ matrices. ♠

3.3.3 Examples of matrix multiplication

Matrix multiplication is omnipresent in mathematics. Below we give an example, which is a baby version of Google's famous page rank algorithm.

(3.9) EXAMPLE.

Suppose that 20% of the people living in the suburbs move to the big city and that 30% of the people living in the big city move to the suburbs per year.

Aiming for a model using probabilities, let us be a bit more precise.

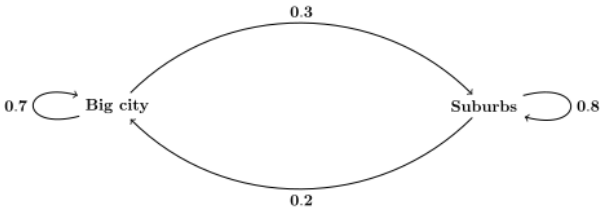
(i) If you live in the suburbs, the probability that you move to the big city is 0.2,

(ii) If you live in the suburbs, the probability that you do not move is 0.8.

(iii) If you live in the big city the probability that you move to the suburbs is 0.3.

(iv) If you live in the big city the probability that you do not move is 0.7.

All of the above probabilities are per year and can be illustrated in the diagram below



We are interested in predicting, using this model, how many people live in the big city and the suburbs given that we know how many people live in the big city, x_0 and in the suburbs y_0 to begin with i.e., setting the time $t = 0$ (years).

How many people x_1 and y_1 live in the two places after the first year ($t = 1$)?

The population of the big city will decrease by 30%, but there are newcomers amounting to 20% of the population in the suburbs. Therefore

$$x_1 = 0.7x_0 + 0.2y_0.$$

In the same way,

$$y_1 = 0.3x_0 + 0.8y_0.$$

Using matrix multiplication, these two equations can be written

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}.$$

For $t = 2$ years, we can repeat the procedure and the result becomes

$$\begin{aligned}\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} &= \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \left(\begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \right) \\ &= \left(\begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \right) \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = P^2 \begin{pmatrix} x_0 \\ y_0 \end{pmatrix},\end{aligned} \tag{3.7}$$

where

$$P = \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix}. \tag{3.8}$$

In general we have the formula

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = P^n \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \tag{3.9}$$

giving the distribution of the populations for $t = n$ years.

Let us experiment a little:

$$\begin{aligned} P^2 &= \begin{pmatrix} 0.55 & 0.3 \\ 0.45 & 0.7 \end{pmatrix} \\ P^3 = PP^2 &= \begin{pmatrix} 0.475 & 0.35 \\ 0.525 & 0.65 \end{pmatrix} \\ P^4 = PP^3 &= \begin{pmatrix} 0.4375 & 0.375 \\ 0.5625 & 0.625 \end{pmatrix} \\ &\vdots \\ P^{15} &= \begin{pmatrix} 0.400018 & 0.399951 \\ 0.599982 & 0.600012 \end{pmatrix} \\ P^{16} &= \begin{pmatrix} 0.400009 & 0.399994 \\ 0.599991 & 0.600006 \end{pmatrix} \end{aligned}$$

It seems that the distribution stabilizes around 40% living in the big city and 60% living in the suburbs of the original total population.

python cell — not displayed in the pdf version.

The matrix P is an example of a stochastic 2×2 matrix. In general, a square matrix is called a *stochastic matrix* if its entries are ≥ 0 and the sum of the entries in each of its column vectors is 1. ♠

(3.10) EXERCISE.

The stabilization observed in Example 3.9 can be computed directly: find the stable distribution by solving the system of linear equations

$$P \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{together with} \qquad x + y = 1,$$

where P is the stochastic matrix in (3.8). Compare with the powers P^n computed above. ♠

(3.11) EXAMPLE.

A simple example of the page rank algorithm is given in Example 2.16. There you encountered the equations

$$\begin{aligned} T_2 &= T_1 + \tfrac{1}{2}T_4 \\ T_3 &= T_2 \\ T_4 &= T_3 \\ T_1 &= \tfrac{1}{2}T_4 \\ T_1 + T_2 + T_3 + T_4 &= 1. \end{aligned}$$

In terms of matrix multiplication the first four equations can be rewritten to

$$\begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix}.$$

Putting

$$P = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

we get a stochastic matrix and may again iterate and compute $P, P^2, P^3, \ldots$.

python cell — not displayed in the pdf version.

Is there a connection between the entries of P^N, where N is very big and the solutions to the linear equations? ♠

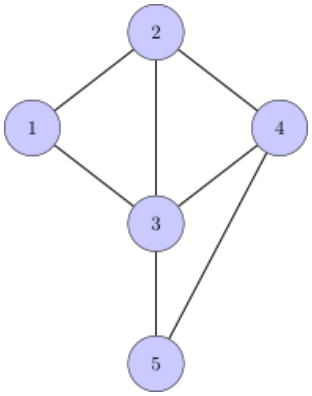
(3.12) EXERCISE.

In the end of Example 3.9 (above) a stochastic matrix is defined. Show that the matrix product of two $n \times n$ stochastic matrices is a stochastic matrix. ♠

Below is an example, where matrix multiplication occurs in networks.

(3.13) EXAMPLE.

Suppose we have five cities connected with roads as shown below



This network has a so called 5×5 *adjacency matrix*, where city i is associated with the i-th row and i-th column. A 1 in the matrix in the (i,j) entry means that there is a road from city i to city j, whereas a 0 means that city i and city j are not connected by a road:

$$A=\begin{pmatrix}0&1&1&0&0\\1&0&1&1&0\\1&1&0&1&1\\0&1&1&0&1\\0&0&1&1&0\end{pmatrix}.$$

Here

$$A^2=\begin{pmatrix}2&1&1&2&1\\1&3&2&1&2\\1&2&4&2&1\\2&1&2&3&1\\1&2&1&1&2\end{pmatrix}\quad\text{and}\quad A^3=\begin{pmatrix}2&5&6&3&3\\5&4&7&7&3\\6&7&6&7&6\\3&7&7&4&5\\3&3&6&5&2\end{pmatrix}.$$

What is the interpretation of A^2, A^3 and A^n in general? It turns out that the entry (i,j) in the matrix A^n exactly is the number of paths of length n from city i to city j.

For example, there are 3 paths from city 1 to city 5 of length 3 corresponding to the paths $1245, 1345, 1235$. The 2 paths from city 1 to city 1 of length 3 are $1231, 1321$ and the 5 paths of length 3 from city 1 to city 2 are $1342, 1242, 1312, 1212, 1232$.

A deeper explanation. Suppose that we have a network with m cities and adjacency matrix A.

The general proof of the observations above in our special example, builds on the fact that a path of length n from city i to city j has to end with a road from a neighboring city k to j. For every one of these neighboring cities, we may count the number of paths of length $n-1$ from city i. If A^{n-1}_{gh} is the number of paths of length $n-1$ from city g to city h, then matrix multiplication tells us that

$$A^n_{ij}=A^{n-1}_{i1}A_{1j}+\cdots+A^{n-1}_{im}A_{mj}$$

This number is exactly the number of paths of length n from city i to city j, since $A_{kj}=1$ only when k is a neighboring city to city j (and 0 otherwise).

♠

3.4 Matrix arithmetic

Matrix multiplication is very different from ordinary multiplication of numbers: it is not commutative. Consider the matrices

$$A=\begin{pmatrix}0&1\\0&0\end{pmatrix}\qquad\text{and}\qquad B=\begin{pmatrix}0&0\\1&0\end{pmatrix}.$$

Then

$$AB=\begin{pmatrix}1&0\\0&0\end{pmatrix}\qquad\text{and}\qquad BA=\begin{pmatrix}0&0\\0&1\end{pmatrix}$$

i.e., $AB\neq BA$.

python cell — not displayed in the pdf version.

Addition of matrices is like ordinary addition, except that you add all the entries of the involved matrices.

3.4.1 Matrix addition

Addition of two matrices with the same number of rows and columns is defined below.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & \cdots & a_{1n}+b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & \cdots & a_{mn}+b_{mn} \end{pmatrix}.$$

The zero matrix 0 is the neutral element for matrix addition:

$$A+0=0+A=A$$

for every $m \times n$ matrix A, where 0 here denotes the $m \times n$ zero matrix.

(3.14) EXERCISE.

Give an example of a non-zero 2×2 matrix, such that

$$A^2 = 0.$$

♠

3.4.2 Multiplication of a number and a matrix

A matrix may be multiplied by a number λ by multiplying each entry by the number:

$$\lambda \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} \lambda a_{11} & \cdots & \lambda a_{1n} \\ \vdots & \ddots & \vdots \\ \lambda a_{m1} & \cdots & \lambda a_{mn} \end{pmatrix}.$$

(3.15) EXERCISE.

Does there exist a number λ, such that

$$\lambda \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 8 & 10 & 15 \end{pmatrix}?$$

♠

(3.16) EXERCISE.

Let A be a 2×2 matrix, such that

$$AB = BA,$$

for every other 2×2 matrix B. Show that A is a diagonal matrix of the form

$$A = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix},$$

where $a \in \mathbb{R}$ i.e., $A = aI_2$. ♠

3.4.3 The distributive law

Ordinary numbers a,b,c satisfy $a(b+c)=ab+ac$. This rule also holds for matrices and is called the distributive law (multiplication is distributed over plus)

(3.17) PROPOSITION.

Let B and C be $m\times n$ matrices, A an $r\times m$ matrix and D an $n\times s$ matrix. Then

$$A(B+C)=AB+AC \qquad \textit{and} \qquad (B+C)D=BD+CD.$$

Proof. Let us start by looking at $A(B+C)=AB+AC$. Here it suffices to do the proof, when A is a row vector and B,C column vectors, since

$$(A(B+C))_{ij}=A_i(B+C)^j=A_i(B^j+C^j).$$

For $(B+C)D=BD+CD$, we may reduce to the case, where B,C are row vectors and D a column vector, since

$$((B+C)D)_{ij}=(B+C)_iD^j=(B_i+C_i)D^j.$$

Both of these cases follow using the distributive law for ordinary numbers. □

(3.18) EXERCISE.

Suppose that A and B are two 2×2 matrices. Is it true that

$$(A+B)^2=A^2+B^2+2AB?$$

What about

$$(A+B)(A-B)=A^2-B^2?$$

♠

3.4.4 The miraculous associative law

It does not make sense to multiply three matrices A,B and C. We have only defined matrix multiplication for two matrices. There are two natural ways of evaluating ABC:

$$(AB)C \qquad \text{and} \qquad A(BC).$$

We can begin by multiplying A by B and then multiply C from the right. However, we may just as well start by multiplying B by C and then multiply A from the left.

It is in no way clear, that these two computations give the same result!

That this turns out to be true, is just one of many miracles in the universe (there is a rather cool mathematical explanation, though, addressed in an exercise below).

(3.19) THEOREM.

Let A be an $m\times n$ matrix, B an $n\times r$ matrix and C an $r\times s$ matrix. Then

$$(AB)C=A(BC).$$

Proof. We must prove that

$$((AB)C)_{ij} = (A(BC))_{ij}$$

for $1 \leq i \leq m$ and $1 \leq j \leq s$. The left hand side can be written

$$\begin{aligned}(AB)_i C^j &= (A_i B^1, \ldots, A_i B^r) C^j \\ &= (A_i B^1) C_{1j} + (A_i B^2) C_{2j} + \cdots + (A_i B^r) C_{rj}.\end{aligned} \tag{3.10}$$

The right hand side is

$$A_i (BC)^j = A_i \begin{pmatrix} B_1 C^j \\ \vdots \\ B_n C^j \end{pmatrix} = A_{i1}(B_1 C^j) + \cdots + A_{in}(B_n C^j). \tag{3.11}$$

Writing the row-column multiplications in (3.10), we get

$$\begin{aligned} &A_{i1}B_{11}C_{1j} + \cdots + A_{in}B_{n1}C_{1j} + \\ &A_{i1}B_{12}C_{2j} + \cdots + A_{in}B_{n2}C_{2j} + \\ &\vdots \\ &A_{i1}B_{1r}C_{rj} + \cdots + A_{in}B_{nr}C_{rj}. \end{aligned} \tag{3.12}$$

Writing the row-column multiplications in (3.11), we get

$$\begin{aligned} &A_{i1}B_{11}C_{1j} + \cdots + A_{i1}B_{1r}C_{rj} + \\ &A_{i2}B_{21}C_{1j} + \cdots + A_{i2}B_{2r}C_{rj} + \\ &\vdots \\ &A_{in}B_{n1}C_{1j} + \cdots + A_{in}B_{nr}C_{rj}. \end{aligned} \tag{3.13}$$

The rows in the sum in (3.12) correspond to the columns in the sum (3.13). Therefore these sums are equal and $((AB)C)_{ij} = (A(BC))_{ij}$. □

(3.20) REMARK.

The associative law $(AB)C = A(BC)$ is true, but in computing ABC there can be a (big) difference in the number of multiplications in the two computations $A(BC)$ and $(AB)C$ i.e., efficiency is not associative for matrix multiplication. In the notation of Theorem 3.19, computing $(AB)C$ requires

$$mnr + mrs = mr(n+s)$$

multiplications, whereas computing $A(BC)$ requires

$$nrs + mns = ns(m+r)$$

multiplications. If for example $m = 10000, n = 10, r = 10000$ and $s = 10$, then computing $(AB)C$ requires $2 \cdot 10^9$ multiplications, whereas computing $A(BC)$ requires $2 \cdot 10^6$ multiplications!

(3.21) EXERCISE.

Verify the associative law for the three matrices

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \qquad \text{and} \qquad C = \begin{pmatrix} 6 & 3 \\ 5 & 2 \\ 4 & 1 \end{pmatrix}$$

by showing by explicit computation that

$$(AB)C = A(BC).$$

♠

(3.22) EXERCISE.

There is in fact a high tech explanation that the associative law for matrices holds. An explanation that makes the calculations in the above proof superfluous and shows the raw power of abstract mathematics: suppose that $f : \mathbb{R}^n \to \mathbb{R}^m, g : \mathbb{R}^r \to \mathbb{R}^n$ and $h : \mathbb{R}^s \to \mathbb{R}^r$ are linear maps. Then $f \circ (g \circ h)$ and $(f \circ g) \circ h$ are both linear maps from $\mathbb{R}^s \to \mathbb{R}^m$, such that

$$(f \circ (g \circ h))(x) = ((f \circ g) \circ h)(x) = f(g(h(x)))$$

for every $x \in \mathbb{R}^s$. How does this relate to the associative law for matrix multiplication? ♠

3.5 The inverse matrix

You are allowed to divide by a number provided it is $\neq 0$. Does it make sense to divide by matrices?

It does, but there are some matrices that correspond to the number 0 that we are not allowed to divide by.

(3.23) EXERCISE.

Let A, B and C be $n \times n$ matrices. Show that

$$BA = I_n$$

and

$$AC = I_n$$

implies that $B = C$. ♠

(3.24) DEFINITION.

An $n \times n$ matrix A is called invertible, if there exists an $n \times n$ matrix B, such that

$$AB = BA = I_n.$$

In this case, B is called the inverse matrix of A and denoted A^{-1}.

As a reality check, you should convince yourself that the 2×2 matrix

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

is not invertible. In fact, here the associative law from Theorem 3.19 is incredibly useful: if A is an invertible matrix with inverse matrix B and $AC = 0$, then $C = 0$:

$$AC = 0 \implies B(AC) = B0 = 0 \implies (BA)C = I_n C = C = 0.$$

But for our matrix above,

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

so it sends a non-zero vector to 0 and cannot be invertible.

(3.25) EXERCISE.

Show that a square matrix with a column or row consisting entirely of zeros cannot be invertible. ♠

(3.26) EXERCISE.

Suppose that

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

with $D = ad - bc \neq 0$. Prove that A is invertible with

$$A^{-1} = \frac{1}{D} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

♠

(3.27) EXERCISE.

When is a square diagonal matrix invertible? Look first at the 2×2 case:

$$\begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}.$$

♠

The inverse matrix can be computed in `numpy`:

python cell — not displayed in the pdf version.

The inverse matrix enters the picture when solving n linear equations with n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

can be rewritten using matrix notation as

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

or more compactly as $Ax = b$.

If A is invertible, then the associative law gives the following:

$$\begin{aligned} Ax = b &\iff \\ A^{-1}(Ax) = A^{-1}b &\iff \\ (A^{-1}A)x = A^{-1}b &\iff \\ Ix = A^{-1}b &\iff \\ x = A^{-1}b. & \end{aligned}$$

The inverse matrix gives the solution to the linear equations $Ax = b$ just by one matrix multiplication!

(3.28) EXAMPLE.

The system of linear equations

$$\begin{array}{rcrcl} 5x & + & 3y & = & 13 \\ 3x & + & 2y & = & 8 \end{array} \tag{3.14}$$

can be rewritten using matrix multiplication to

$$Av = b,$$

where

$$A = \begin{pmatrix} 5 & 3 \\ 3 & 2 \end{pmatrix}, \qquad v = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{and} \qquad b = \begin{pmatrix} 13 \\ 8 \end{pmatrix}.$$

Here A is invertible and

$$A^{-1} = \begin{pmatrix} 2 & -3 \\ -3 & 5 \end{pmatrix}.$$

One simple matrix multiplication

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 & -3 \\ -3 & 5 \end{pmatrix} \begin{pmatrix} 13 \\ 8 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

shows the solution we expect from (3.14). ♠

The product of two invertible matrices (when this makes sense) is an invertible matrix. This is the content of the following result.

(3.29) PROPOSITION.

The product AB of two invertible matrices A and B is invertible and $(AB)^{-1} = B^{-1}A^{-1}$.

Proof. We must check that

$$(B^{-1}A^{-1})(AB) = I \qquad \text{and} \qquad AB(B^{-1}A^{-1}) = I.$$

Let us check the first condition using the associative law:

$$\begin{aligned} (B^{-1}A^{-1})(AB) &= ((B^{-1}A^{-1})A)B \\ &= (B^{-1}(A^{-1}A))B \\ &= (B^{-1}I)B = B^{-1}(IB) = B^{-1}B = I, \end{aligned}$$

where I denotes the identity matrix. The condition $AB(B^{-1}A^{-1}) = I$ is verified in the same way. □

(3.30) EXERCISE.

We have defined a matrix A to be invertible if there exists a matrix B, such that $AB = I$ and $BA = I$. Suppose that only $BA = I$. Can we then conclude that $AB = I$?

Find the mistake in the argument below.

Suppose that $BA = I$. Then for every $y \in \mathbb{R}^n$ we have $Ax = y \implies x = (BA)x = By$. Therefore $A(By) = (AB)y = y$ for every $y \in \mathbb{R}^n$ and we have proved that $AB = I$.

♠

(3.31) EXERCISE.

Let

$$N = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Compute the powers N^k for $k \geq 2$ i.e., $N^2, N^3, \ldots$. Now let

$$A = I + N,$$

where $I = I_4$. Show that A is invertible, and

$$A^{-1} = I - N + N^2 - N^3.$$

Compute A^{-1}.

Do you see a way of generalizing this computation to $n \times n$ matrices N with a property shared by the 4×4 matrix above?

♠

3.5.1 Well, how do I find the inverse of a matrix?

Finding the inverse of a matrix or deciding that the matrix is not invertible is a matter of solving systems of linear equations.

Given an $n \times n$ matrix A, we need to see if there exists an $n \times n$ matrix B, such that

$$AB = I, \tag{3.15}$$

where $I = I_n$ is the identity matrix of order n. We can do this by computing the columns of B. From the definition in (3.15), the j-th column B^j of B must satisfy

$$AB^j = I^j. \tag{3.16}$$

This follows from the definition of matrix multiplication!

The identity in (3.16) is a system of n linear equations in n unknowns. The unknowns are the entries in the j-th column B^j of the inverse matrix A^{-1} (if it exists).

(3.32) EXAMPLE.

Suppose that A is a 2×2 matrix. Then the inverse matrix B (if it exists) can be computed from the systems of linear equations below.

$$AB^1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad AB^2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Writing

$$\begin{pmatrix} x \\ y \end{pmatrix} = B^1 \qquad \text{and} \qquad \begin{pmatrix} u \\ v \end{pmatrix} = B^2$$

for the first and second columns, the systems of linear equations can be written as

$$\begin{array}{rcrcl} A_{11}x & + & A_{12}y & = & 1 \\ A_{21}x & + & A_{22}y & = & 0 \end{array} \qquad \text{and} \qquad \begin{array}{rcrcl} A_{11}u & + & A_{12}v & = & 0 \\ A_{21}u & + & A_{22}v & = & 1\text{'} \end{array}$$

where

$$B = \begin{pmatrix} x & u \\ y & v \end{pmatrix}.$$

A concrete example along with a useful way of keeping track of the computation is presented in the video below.

(3.33) VIDEO.

VIDEO: https://youtu.be/4BX3eW4qo3g

♠

(3.34) EXERCISE.

Compute the inverse of the matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

by employing the method of solving linear equations above. Explain the steps in your computation. You may find it useful to collect inspiration from the video in Example 3.32. ♠

3.6 The transposed matrix

The transpose of an $m \times n$ matrix A is the $n \times m$ matrix $A^\top$ given by

$$A^\top_{ij} = A_{ji},$$

where $1 \leq i \leq m$ and $1 \leq j \leq n$. As an example,

$$\begin{pmatrix} 0 & 2 & 4 & -2 \\ 3 & 2 & 7 & 4 \end{pmatrix}^\top = \begin{pmatrix} 0 & 3 \\ 2 & 2 \\ 4 & 7 \\ -2 & 4 \end{pmatrix}.$$

Notice also that $(A^\top)^\top = A$ for an arbitrary matrix A.

(3.35) PROPOSITION.

Let A be an $m \times r$ matrix and B an $r \times n$ matrix. Then

$$(AB)^\top = B^\top A^\top.$$

Proof. By definition $(AB)^\top_{ij} = (AB)_{ji}$. This entry is given by row-column multiplication of the j-th row in A and the i-th column in B, which is the row-column multiplication of the i-th row in $B^\top$ and the j-th column in $A^\top$. □

(3.36) EXERCISE.

Let A be a square matrix. Prove that A is invertible if and only if $A^\top$ is invertible. ♠

(3.37) EXERCISE.

In the python cell below, you are supposed to experiment a bit by entering an arbitrary matrix B and studying the square matrix $BB^\top$. Is there anything special about this product? Press the *Further explanation* button below the cell to display the rest of the exercise after(!) you have completed your experimentation.

python cell — not displayed in the pdf version.

> **Further explanation.** A square matrix A is called symmetric if $A = A^\top$. Prove that
>
> $$BB^\top$$
>
> is a symmetric matrix, where B is an arbitrary matrix.

♠

3.7 Symmetric matrices

A (square) matrix A is called symmetric if $A = A^\top$. Visually, this means that A is symmetric around the diagonal like the 3×3 matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 3 & 4 & 6 \end{pmatrix},$$

but not like the 3×3 matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

(3.38) EXERCISE.

Show that

$$B^\top AB$$

is a symmetric matrix, when A is a symmetric matrix and B is an arbitrary matrix. Both matrices are assumed square of the same dimensions. ♠

If A is a symmetric $d \times d$ matrix, we define the function $f_A : \mathbb{R}^d \to \mathbb{R}$ given by

$$f_A(v) = v^\top Av.$$

This definition is rather compact. Let us consider the following example for $d = 2$.

$$A = \begin{pmatrix} a & c \\ c & b \end{pmatrix} \quad \text{and} \quad v = \begin{pmatrix} x \\ y \end{pmatrix}.$$

Then

$$\begin{aligned} v^\top Av &= \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a & c \\ c & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} ax+cy \\ cx+by \end{pmatrix} \\ &= x(ax+cy) + y(cx+by) = ax^2 + by^2 + 2cxy. \end{aligned}$$

You are also encouraged to watch the short video below for an example with concrete numbers.

(3.39) VIDEO.

VIDEO: https://youtu.be/YNwrLMJ8byM

Inside the set of the symmetric matrices we find two very important subsets of matrices: the positive definite and the positive semi-definite matrices. They correspond to positive and non-negative real numbers.

3.7.1 Positive definite matrices

A symmetric matrix A is called *positive definite* if

$$f_A(v) > 0$$

for every $v \in \mathbb{R}^d \setminus \{0\}$. Probably the first example of a positive definite 2×2-matrix one thinks of is A being the identity matrix. Here

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 + y^2.$$

Of course, here $x^2 + y^2 = 0$ if and only if $x = y = 0$ or $\begin{pmatrix} x \\ y \end{pmatrix} = 0$.

(3.40) EXERCISE.

Give examples of (non-zero) 1×1 and 2×2 matrices that are positive definite and ones that fail to be positive definite.

When is a 2×2 diagonal matrix positive definite? ♠

(3.41) EXERCISE.

Let A be a symmetric $d \times d$ matrix. Show that A is not positive definite if $A_{11} < 0$. ♠

3.7.2 Positive semi-definite matrices

A symmetric matrix A is called *positive semi-definite* if

$$f_A(v) \geq 0$$

for every $v \in \mathbb{R}^d$. A positive definite matrix is positive semi-definite. Probably the first example of a non positive definite, but positive semi-definite 2×2-matrix one thinks of is A being the zero matrix. Here

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0,$$

for every $x, y \in \mathbb{R}$.

(3.42) EXERCISE.

Give an example of a non-zero matrix that is positive semi-definite, but not positive definite.

When is a 2×2 diagonal matrix positive semi-definite? ♠

3.7.3 Symmetric reductions

As you probably have noticed, it is rather straightforward to see when a diagonal matrix is positive (semi)definite. For a general symmetric matrix, one needs to reduce to the case of a diagonal matrix. This is done using the following result.

(3.43) PROPOSITION.

Let A be a symmetric $n \times n$ matrix and B an invertible $n \times n$ matrix. Then A is positive (semi) definite if and only if

$$B^\top AB$$

is positive (semi) definite.

Proof. Every vector $v \in \mathbb{R}^n$ is equal to Bu for a unique $u \in \mathbb{R}^n$, since B is invertible. Why? The upshot is that the equation

$$v = Bu$$

can be solved by multiplying both sides by B^{-1} giving

$$v = Bu \iff B^{-1}v = B^{-1}(Bu) = (B^{-1}B)u = u.$$

So we get

$$v^\top Av = (Bu)^\top A(Bu) = u^\top (B^\top AB)u.$$

This computation shows that A is positive (semi) definite if $B^\top AB$ is positive semi-definite. The same reasoning with $u = B^{-1}v$ shows that $B^\top AB$ is positive (semi) definite if A is positive (semi) definite.

Notice that it is important that $Bv = 0$ only happens when $v = 0$. □

(3.44) EXERCISE.

Let

$$D = \begin{pmatrix} d & 0 \\ 0 & e \end{pmatrix}$$

be a diagonal matrix. What conditions must the diagonal entries d and e satisfy in order for D to be positive definite?

Let

$$A = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$$

denote a symmetric 2×2 matrix, where $a \neq 0$. Let

$$B = \begin{pmatrix} 1 & -\frac{c}{a} \\ 0 & 1 \end{pmatrix}.$$

Show that B is invertible and compute

$$B^\top AB.$$

Use this to show that A is positive definite if and only if $a > 0$ and $ab - c^2 > 0$.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be the function defined by

$$f(x,y) = 2x^2 + 3y^2 + 4xy.$$

Show that $f(x,y) \geq 0$ for every $x, y \in \mathbb{R}$. ♠

(3.45) QUIZ.

quiz — not displayed in the pdf version.

♠

4 WHAT IS OPTIMIZATION?

In this chapter we will denote the set of column vectors with d rows by $\mathbb{R}^d$. The arithmetic of $d \times 1$ matrices applies i.e., we may add vectors in $\mathbb{R}^d$ and multiply them by a number in $\mathbb{R}$.

In the next chapter we will introduce them as *euclidean* vector spaces. The term euclidean refers to a norm: a function measuring the size of a vector. In this chapter we only need the structure as column vectors.

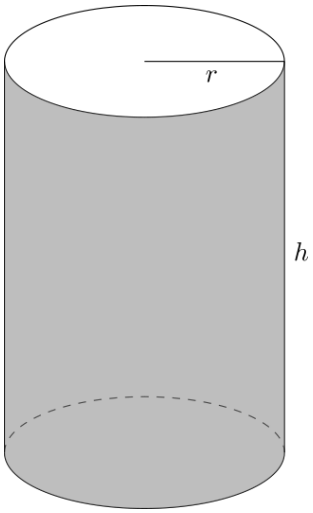
4.1 What is an optimization problem?

An optimization problem consists of maximizing or minimizing a function subject to constraints.

Below are two classical examples related to minimizing (non-linear) functions subject to (non-linear) constraints. These are actually examples of convex optimization problems. More about that later.

(4.1) EXAMPLE.

A cylindrical can is supposed to have a volume of V m^3. The material used for the top and bottom costs T DKK per m^2 and the material used for the side costs S DKK per m^2. Give the dimensions r and h of the can minimizing the price of the materials.



The cost of the top and bottom pieces is $2\pi r^2 T$. The cost of the side material is $2\pi rhS$. The constraint is that the volume must be V. This is expressed in the equation $\pi r^2 h = V$. All in all the optimization problem is

$$\min_{\substack{\pi r^2 h = V \\ r \geq 0,\ h \geq 0}} \quad 2\pi r^2 T + 2\pi rhS,$$

where V, T and S are constants.

(4.2) EXERCISE.

Can you see a way of solving this optimization problem by eliminating h in the constraint $\pi r^2 h = V$?

Hint.

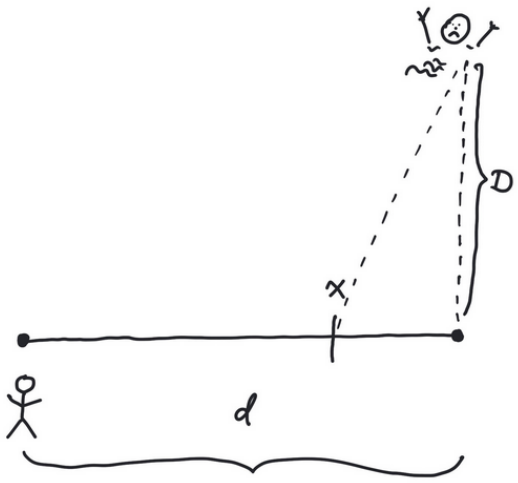
$$\pi r^2 h = V \iff h = \frac{V}{\pi r^2}$$

and h can be inserted in $2\pi r^2 T + 2\pi rhS$. Why is this helpful?

♠ ♠

(4.3) EXAMPLE.

A person is in distress D meters from the beach. The life guard spots the situation, but is d meters from where he would naturally jump in the water as indicated below. The life guard runs 8 m/s on the beach and swims 1 m/s in the water. How far (x) should he run along the beach before jumping into the water in order to minimize the time needed to reach the person in distress?



The time spent moving with a speed of v over a distance of s is

$$t = \frac{s}{v}.$$

If the life guard jumps in the water at the point x he will have to swim a distance of

$$\sqrt{D^2 + (d-x)^2}$$

using the Pythagorean theorem. Therefore the optimization problem becomes

$$\min_{x \geq 0} \frac{x}{8} + \sqrt{D^2 + (d-x)^2}$$

Strictly speaking we do not need the constraint $x \geq 0$, as the life guard is free to run in the other direction. So the optimization problem is simply to minimize

$$\frac{x}{8} + \sqrt{D^2 + (d-x)^2}$$

with no strings attached i.e., $x \in \mathbb{R}$ is just assumed to be any number.

python cell — not displayed in the pdf version.

♠

(4.4) EXERCISE.

Run the cell above and read off the (approximately) optimal jumping point x from the plot. Experiment with other values of D and d and with a faster swimmer. Finding the optimal x exactly requires the derivative — a story we return to in later chapters. ♠

(4.5) EXERCISE.

You need to build a rectangular fence in front of your house for a herb garden. Your house will make up one side of the rectangle, so you only need to build three sides. Suppose you have 10 m of wire. What is the maximum area of the herb garden you can wall in? ♠

4.2 General definition

An optimization problem consists of a subset $D \subseteq \mathbb{R}^d$ and a function $f : D \to \mathbb{R}$. We will consider optimization problems in the context of minimization. Optimize in this situation means minimize.

(4.6) DEFINITION.

In our most general setting an optimization problem looks like

$$\min_{v \in C} f(v),$$

where C and D are subsets of $\mathbb{R}^d$ with $C \subseteq D$ and $f : D \to \mathbb{R}$ is a function. The notation is read: minimize the value $f(v)$ subject to the constraint that the vector v belongs to C. The constraint is always written under the min*; several constraints may be stacked there. A solution to the optimization problem is a vector $v_0 \in C$, such that*

$$f(v_0) \leq f(v)$$

for every $v \in C$. Here v_0 is called an optimum *and $f(v_0)$ is called the* optimal value.

(4.7) REMARK.

The complexity of the problem depends very much on the nature of C and f. Also, we cannot even be certain that an optimization problem has a solution. Consider the problem

$$\min_{x \in C} x,$$

where $C = \{x \in \mathbb{R} \mid x \leq 0\}$.
Here x can be made arbitrarily small subject to the constraint $x \in C$ and the problem has no optimal solution.

(4.8) REMARK.

We have deliberately not included maximization problems in Definition 4.6. This is because a maximization problem, such as

$$\max_{v\in C} f(v) \tag{4.1}$$

can be formulated as the minimization problem

$$\min_{v\in C} -f(v).$$

A solution to (4.1) is a vector $v_0 \in C$, such that

$$f(v_0) \geq f(v)$$

for every $v \in C$. Again, v_0 is called an optimum and $f(v_0)$ the optimal value.

(4.9) EXERCISE.

Suppose that the maximization problem

$$\max_{v\in C} f(v) \tag{4.2}$$

is formulated as the minimization problem

$$\min_{v\in C} -f(v). \tag{4.3}$$

Show that $f(v_0)$ is the optimal value and v_0 the optimum for (4.2) if $-f(v_0)$ is the optimal value and v_0 the optimum for (4.3). ♠

(4.10) EXERCISE.

Suppose that $a > 0$. Solve the optimization problem

$$\min_{x\in\mathbb{R}} ax^2 + bx + c.$$

> **Hint.** Complete the square, as when deriving the formula for the roots of a quadratic polynomial in the second chapter.

♠

(4.11) QUIZ.

quiz — not displayed in the pdf version.

♠

4.3 Convex optimization

Particularly well behaved optimization problems are the convex ones. These are optimization problems, where $C \subseteq \mathbb{R}^d$ is a convex subset and $f : C \to \mathbb{R}$ a convex function in Definition 4.6.

To define these concepts we first introduce the notion of a line in $\mathbb{R}^d$.

(4.12) DEFINITION.

A line $L \subseteq \mathbb{R}^d$ is a subset of the form

$$L = \{u + tv \mid t \in \mathbb{R}\},$$

where $u, v \in \mathbb{R}^d$ with $v \neq 0$.

(4.13) EXAMPLE.

A line L in the plane $\mathbb{R}^2$ is (usually) given by its equation

$$y = ax + b. \tag{4.4}$$

This means that it consists of points $(x, y) \in \mathbb{R}^2$ satisfying $y = ax + b$. Here a can be interpreted as the slope of the line and b the intersection with the y-axis.

What about all the points (x, y) with $x = 0$? Certainly they also deserve to be called a line. However, they do not satisfy an equation like (4.4). Informally, this line has infinite slope.

Therefore we introduce the parametric representation of a line: a line is the set of points of the form

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + t \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}, \tag{4.5}$$

where $t \in \mathbb{R}$,

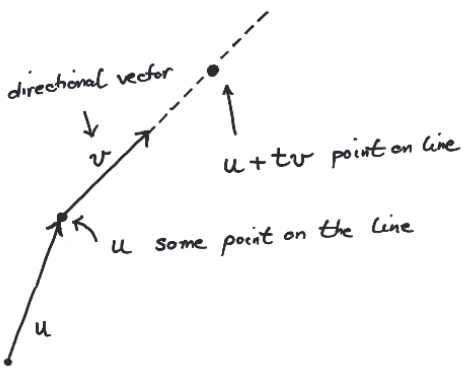
$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

is any point on the line and

$$\begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$$

is a non-zero (directional) vector.

(4.14) FIGURE.



Example of a line in $\mathbb{R}^2$ with (directional) vector $v \in \mathbb{R}^2$ through the point $u \in \mathbb{R}^2$.

Given two distinct points

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} x_1 \\ y_1 \end{pmatrix},$$

there is one and only one line passing through them. This line is given by

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + t \begin{pmatrix} x_1 - x_0 \\ y_1 - y_0 \end{pmatrix}. \tag{4.6}$$

How do we convert the line $y = ax + b$ in (4.4) to the parametric form (4.5)? Well, we know that the two distinct points

$$\begin{pmatrix} 0 \\ b \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} 1 \\ a+b \end{pmatrix}$$

are on the line. Therefore it is given by

$$\begin{pmatrix} 0 \\ b \end{pmatrix} + t \begin{pmatrix} 1 \\ a \end{pmatrix}$$

by (4.6). ♠

(4.15) EXERCISE.

Compute the parametric representation of the line L through the points $(1,1)$ and $(2,3)$. Also compute a and b in the representation $y = ax + b$ for L. ♠

(4.16) EXERCISE.

What is the parametric representation of the line consisting of the points (x,y) with $x = 0$? ♠

(4.17) EXERCISE.

Show in Definition 4.12 that if L is given by u and v, then you might as well replace v by sv, where s is a real number and $s \neq 0$. It gives the same line. ♠

(4.18) EXERCISE.

Show that there is a unique line passing through two distinct points $p, q \in \mathbb{R}^d$ and that it is given by $u = p$ and $v = q - p$ in Definition 4.12. ♠

(4.19) EXERCISE.

Do the points

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \quad \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}$$

lie on the same line in $\mathbb{R}^3$? ♠

(4.20) EXERCISE.

Show that the line through two distinct points $p, q \in \mathbb{R}^d$ is equal to the subset

$$\{(1-t)p + tq \mid t \in \mathbb{R}\} \subseteq \mathbb{R}^d.$$

♠

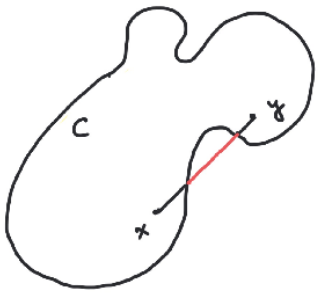
(4.21) DEFINITION.

A convex subset $C \subseteq \mathbb{R}^d$ is a subset that contains the line segment between any two of its points $x, y \in C$ i.e.,

$$(1-t)x + ty \in C$$

for every number t with $0 \leq t \leq 1$.

(4.22) FIGURE.



Example of non-convex subset of $\mathbb{R}^2$.

(4.23) QUIZ.

quiz — not displayed in the pdf version.

♠

(4.24) EXERCISE.

A closed interval in $\mathbb{R}$ is a subset of the form

$$[a, b] = \{x \mid a \leq x \leq b\}$$

for $a \leq b$. Prove that $[a, b]$ is a convex subset of $\mathbb{R}$.

Hint.

Keep cool and just apply the definitions! First of all, $x \in [a,b]$ if and only if

$$a \leq x \wedge x \leq b. \tag{4.7}$$

Now pick any $t \in [0,1]$. We must show that if $x \in [a,b]$ and $y \in [a,b]$, then

$$(1-t)x+ty \in [a,b].$$

You may also write this out as

$$a \leq x \wedge x \leq b \qquad \wedge \qquad a \leq y \wedge y \leq b$$

implies that

$$a \leq (1-t)x+ty \qquad \wedge \qquad (1-t)x+ty \leq b.$$

Hint.

$$a \leq x \implies (1-t)a \leq (1-t)x \qquad \wedge \qquad a \leq y \implies ta \leq ty$$

implies that

$$(1-t)a+ta \leq (1-t)x+ty.$$

What is $(1-t)a+ta$?

♠

(4.25) EXERCISE.

Let A and B be convex subsets of $\mathbb{R}^d$. Prove that $A \cap B$ is a convex subset of $\mathbb{R}^d$. Generalize this to show that if $A_1, \ldots, A_n$ are any number of convex subsets of $\mathbb{R}^d$, then their intersection

$$A_1 \cap \cdots \cap A_n$$

is a convex subset of $\mathbb{R}^d$. Is the union of two convex subsets necessarily convex? ♠

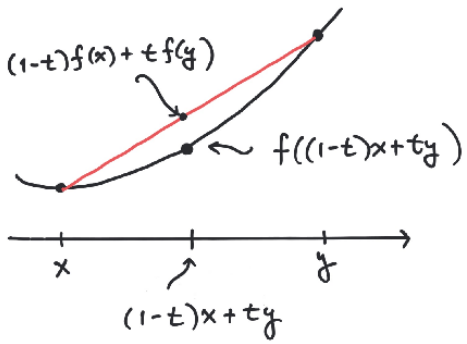
(4.26) DEFINITION.

A convex function is a function $f : C \to \mathbb{R}$ defined on a convex subset $C \subseteq \mathbb{R}^d$, such that

$$f((1-t)x+ty) \leq (1-t)f(x)+tf(y)$$

for every number t with $0 \leq t \leq 1$.

(4.27) FIGURE.



Graph of convex function. The line segment between $(x, f(x))$ and $(y, f(y))$ lies above the graph.

(4.28) EXERCISE.

(i) Let the function $f : \mathbb{R} \to \mathbb{R}$ be given by $f(x) = ax + b$, where $a, b \in \mathbb{R}$. Show that f is a convex function.

Hint. Try the case $a = 0$ first.

(ii) Can you at this point prove that $f(x) = x^2$ is a convex function?

Hint. Simplify

$$(1-t)x^2 + ty^2 - ((1-t)x + ty)^2$$

to an expression that has to be non-negative.

Hint.

python cell — not displayed in the pdf version.

(iii) Using that $f(x) = x^2$ is a convex function, prove that $g(x) = x^4$ is a convex function.

Hint. Use that $g(x) = f(x)^2$ and $a \leq b \implies a^2 \leq b^2$ if $a, b \geq 0$ (here we really need $a, b \geq 0$, since for example $-2 \leq -1$, but $4 \leq 1$ is not true) to conclude that

$$((1-t)x + ty)^4 = (((1-t)x + ty)^2)^2 \leq ((1-t)x^2 + ty^2)^2 \leq (1-t)x^4 + ty^4$$

for $x, y \in \mathbb{R}$.

(iv) It is a fact that $f(x) = x^3$ is not a convex function, but can you explain this using the definition of a convex function?

Hint. Try $x = -1, y = 0$ and $t = \frac{1}{2}$.

♠

(4.29) LEMMA.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. Then the subset

$$C = \{v \in \mathbb{R}^d \mid f(v) \leq a\}$$

is a convex subset of $\mathbb{R}^d$, where $a \in \mathbb{R}$.

Proof. Suppose that $u, v \in C$ and $t \in [0, 1]$. Looking at Definition 4.21 we must prove that

$$(1-t)u + tv \in C.$$

By the definition of f being convex (Definition 4.26), it follows that

$$f((1-t)u + tv) \leq (1-t)f(u) + tf(v).$$

But, since $f(u) \leq a$ and $f(v) \leq a$ we have

$$(1-t)f(u) \leq (1-t)a$$
$$tf(v) \leq ta$$

and therefore

$$(1-t)f(u)+tf(v) \leq (1-t)a+ta = a.$$

Therefore,

$$f((1-t)u+tv) \leq a$$

and $(1-t)u+tv \in C$. □

Why are convex optimization problems interesting? We will return to this in Chapter 6 on convex functions. As a sneak preview let me comment already now.

(4.30) REMARK.

In hunting for optimal solutions to an optimization problem one is often stuck with a point $v_0 \in \mathbb{R}^d$, which is optimal locally. This means that $f(v_0) \leq f(v)$ for every v that is sufficiently close to v_0 (we will explain what this means in the next chapter). The remarkable thing that happens in a convex optimization problem is that if v_0 is optimal locally, then it is a global optimum! It satisfies $f(v_0) \leq f(v)$ not only for v close to v_0, but for every $v \in C$.

The optimization problem in Exercise 4.10 is a very typical convex optimization problem.

(4.31) EXAMPLE.

Below you see a plot of the function (press Run)

$$f(x) = x^3+2x^2+x+1$$

restricted to the interval $[-1.5,0]$. You can see that it has a local minimum around -0.3 and also that this minimum is not a global minimum (certainly $f(-1.4)$ is smaller). So $f(x)$ is not a convex function on this interval according to Remark 4.30 (but if you look at it more locally on the interval $[-0.6,0]$ it is a convex function).

python cell — not displayed in the pdf version.

♠

(4.32) EXERCISE.

Solve the optimization problem

$$\min_{x \in C} x^3+2x^2+x+1$$

for $C = [-0.6,0]$ and $C = [-2,0]$. ♠

4.4 Linear optimization

We will start this section with a concrete example.

A company produces two products A and B. The product A is selling for 350 USD and B is selling for 300 USD. There are certain limited resources in the production of A and B. Two raw materials S_1 and S_2 are needed along with employee work time. The production of A requires 18 minutes, one unit of S_1 and six units of S_2. The production of B requires 12 minutes, one unit of S_1 and eight units of S_2. There are 3132 minutes of employee work time, 200 units of S_1 and 1440 units of S_2 available. These constraints in the production can be outlined in the diagram below

	minutes	S_1	S_2
A	18	1	6
B	12	1	8
constraint	3132	200	1440

How many units x of A and y of B should the company produce to maximize its profit?

You can rewrite this as the optimization problem

$$\begin{array}{ll} \max & 350x+300y \\ 18x+12y\leq 3132 & \\ x+y\leq 200 & \\ 6x+8y\leq 1440 & \\ x\geq 0,\ y\geq 0 & \end{array}$$

This optimization problem is a special case of linear optimization, which arguably is one of the most successful applications of mathematics (after the introduction of the simplex algorithm following World War II). We will give a taste of the mathematical setup here.

The simplest convex optimization problems are the linear ones. Recall that a linear function $f:\mathbb{R}^d\to\mathbb{R}$ has the form

$$f\begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} = c_1x_1+\cdots+c_dx_d$$

for $c_1,\ldots,c_d\in\mathbb{R}$. Usually we write this with matrix notation as

$$f(x)=c^\top x,$$

where

$$c=\begin{pmatrix} c_1 \\ \vdots \\ c_d \end{pmatrix} \qquad \text{and} \qquad x=\begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}.$$

(4.33) EXERCISE.

Show that a linear function is convex. ♠

A linear optimization problem is not about minimizing a linear function over an arbitrary convex subset. We choose the convex subset as an intersection of subsets of the form

$$\{v\in\mathbb{R}^d \mid a^\top v\leq b\},$$

where $a\in\mathbb{R}^d$ is a non-zero vector and $b\in\mathbb{R}$ a number i.e., a linear optimization problem has the form

$$\min_{v\in C} c^\top v,$$

where

$$\begin{aligned} C &= \{v \in \mathbb{R}^d \mid a_1^\top v \le b_1, \dots, a_m^\top v \le b_m\} \\ &= \{v \in \mathbb{R}^d \mid a_1^\top v \le b_1\} \cap \dots \cap \{v \in \mathbb{R}^d \mid a_m^\top v \le b_m\} \end{aligned} \tag{4.8}$$

and $c, a_1, \dots, a_m \in \mathbb{R}^d$ and $b_1, \dots, b_m \in \mathbb{R}$.

(4.34) EXERCISE.

Use a selection of previous exercises to show that the subset C defined in (4.8) is a convex subset of $\mathbb{R}^d$. ♠

Using matrix notation we write C as

$$C = \{v \in \mathbb{R}^d \mid Av \le b\},$$

where the inequality $Av \le b$ is read entry by entry, A is the $m \times d$ matrix with row vectors $a_1^\top, \dots, a_m^\top$ and

$$b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}.$$

(4.35) EXAMPLE.

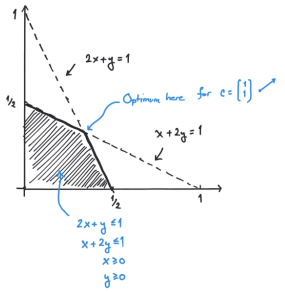
Here is a concrete example for $d = 2$. The optimization problem

$$\begin{array}{c} \max \quad x+y \\ 2x+y \le 1,\ x+2y \le 1 \\ x \ge 0,\ y \ge 0 \end{array}$$

translates into matrix notation with the matrices

$$c = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \text{and} \qquad b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

In this case it is helpful to draw the optimization problem in the plane $\mathbb{R}^2$. This is done below. The drawing suggests that the optimum sits at the corner $(\frac{1}{3}, \frac{1}{3})$, where the two lines $2x+y=1$ and $x+2y=1$ intersect, giving the optimal value $\frac{2}{3}$ — verified by the scipy computation following the picture.



Constraints pictured as shaded area above. Optimum occurs in a vertex (corner).

♠

We will give a general (but rather slow) algorithm below for solving linear optimization problems. In fact it all boils down to solving systems of linear inequalities. Sometimes linear optimization is referred to as linear programming. The basic theory of linear programming was pioneered, among others, by one of the inventors of the modern computer, John von Neumann.

John von Neumann (1903-1957). Picture from LANL.

Python's scipy library has much more advanced algorithms built in for solving linear optimization problems. I have translated the linear optimization problem in Example 4.35 into scipy code below. Notice that `linprog` always *minimizes* and only accepts *upper* bounds (`A_ub`, `b_ub`): maximizing x_1+x_2 becomes minimizing $-(x_1+x_2)$ (Remark 4.8), and a $\geq$ constraint must be multiplied by -1 to become a $\leq$ constraint.

(4.36) EXAMPLE.

python cell — not displayed in the pdf version.

♠

4.5 Fourier-Motzkin elimination

Fourier-Motzkin elimination is a classical method (dating back to 1826, and named after Joseph Fourier and Theodore Motzkin) for solving linear inequalities. It is also a key ingredient in an algorithm for solving linear optimization problems.

I am convinced that the best way to explain this method is by way of an extended example. For more formalities you may consult Chapter 1 of my book Undergraduate Convexity.

Consider the linear optimization problem

$$\max_{\substack{2x+y\leq 6,\ x+2y\leq 6,\ x+2y\geq 2\\ x\geq 0,\ y\geq 0}} x+y. \tag{4.9}$$

By adding the extra variable $z=x+y$, we might as well write this as

$$\max_{(x,y,z)\in P} z,$$

that is: find the maximal value of z, such that there exists $(x,y)\in\mathbb{R}^2$ with $(x,y,z)\in P$, where $P\subseteq\mathbb{R}^3$ is the set of solutions to the system

$$\begin{array}{rcrcl} & & z & = & x+y \\ 2x & + & y & \leq & 6 \\ x & + & 2y & \leq & 6 \\ x & + & 2y & \geq & 2 \\ x & & & \geq & 0 \\ & & y & \geq & 0 \end{array} \tag{4.10}$$

of inequalities[1].

We have the Gauss elimination method for solving systems of linear equations. How do we now solve (4.10), where we also have inequalities?

Well, at first we can actually do a Gauss elimination step by eliminating x in the equation $z = x + y$ i.e., by putting $x = z - y$. This is then inserted into the inequalities in (4.10):

$$\begin{array}{rcrcl} 2(z-y) & + & y & \leq & 6 \\ (z-y) & + & 2y & \leq & 6 \\ (z-y) & + & 2y & \geq & 2 \\ (z-y) & & & \geq & 0 \\ & & y & \geq & 0 \end{array}$$

and we get the system

$$\begin{array}{rcrcl} 2z & - & y & \leq & 6 \\ z & + & y & \leq & 6 \\ z & + & y & \geq & 2 \\ z & - & y & \geq & 0 \\ & & y & \geq & 0 \end{array}$$

of inequalities in the variables z and y. Now we only have inequalities left and we have to invent a trick for eliminating y. Let us isolate y on one side of the inequality signs $\leq$ and $\geq$:

$$\begin{array}{rcrcl} 2z & - & 6 & \leq & y \\ 6 & - & z & \geq & y \\ 2 & - & z & \leq & y \\ & & z & \geq & y \\ & & 0 & \leq & y \end{array}$$

Written a little differently this is the same as

$$\begin{array}{rcrcl} 2z & - & 6 & \leq & y \\ & & & y \leq 6 - z & \\ 2 & - & z & \leq & y \\ & & & y \leq z & \\ & & 0 & \leq & y \end{array} \tag{4.11}$$

Now the scene is set for elimination of y. Listen carefully. First the inequalities in (4.11) can be boiled down to the following two inequalities

$$\begin{array}{rcl} \max(2z-6, 2-z, 0) & \leq & y \\ & \leq & \min(6-z, z) \end{array} \tag{4.12}$$

by using (repeatedly) that $\max(a,b) \leq c \iff a \leq c \wedge b \leq c$ and $c \leq \min(a,b) \iff c \leq a \wedge c \leq b$ for three numbers $a, b, c \in \mathbb{R}$.

Then, finally comes the (Fourier-Motzkin) elimination step: The existence of a solution to (4.12) is equivalent to the single inequality

$$\max(2z-6, 2-z, 0) \leq \min(6-z, z). \tag{4.13}$$

This single inequality can be exploded or expanded (see Exercise 4.38) into the following $6 = 3 \cdot 2$ inequalities

[1] An equality $a = b$ is logically equivalent to the two inequalities $a \leq b$ and $a \geq b$ in the sense that $(a \leq b) \wedge (a \geq b) \iff a = b$.

$$\begin{aligned} 2z-6 &\leq 6-z \\ 2z-6 &\leq z \\ 2-z &\leq 6-z \\ 2-z &\leq z \\ 0 &\leq 6-z \\ 0 &\leq z. \end{aligned}$$

Similarly to (4.11) we now isolate z from the above inequalities:

$$\begin{array}{rcl} & z & \leq 4 \\ & z & \leq 6 \\ 1 \leq & z & \\ & z & \leq 6 \\ 0 \leq & z & \end{array}$$

and find that

$$\begin{array}{rcl} \max(1,0)=1 \leq & z & \\ & z & \leq \min(4,6)=4. \end{array}$$

Therefore the maximum in the optimization problem (4.9) is $z = x+y = 4$. How do we now find numbers $x, y \in \mathbb{R}$ satisfying the constraints in the optimization problem (4.9) with $z = x+y = 4$?

This is simply done inserting first $z = 4$ in (4.12). Here you get the two inequalities $2 \leq y$ and $y \leq 2$. Therefore $y = 2$. Since we had $x = z-y$ from the very beginning we therefore get $x = 2$ and we have the unique solution to the optimization problem.

(4.37) EXERCISE.

What is the solution if we replace Maximize with Minimize in the optimization problem (4.9)? ♠

(4.38) EXERCISE.

Prove the following:

Let $x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{R}$ be $m+n$ numbers. Then

$$\max(x_1, \ldots, x_m) \leq \min(y_1, \ldots, y_n)$$

if and only if the mn inequalities

$$\begin{array}{cccc} x_1 \leq y_1 & x_1 \leq y_2 & \ldots & x_1 \leq y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m \leq y_1 & x_m \leq y_2 & \ldots & x_m \leq y_n \end{array}$$

are satisfied. ♠

(4.39) EXERCISE.

The following is Exercise 1.8 from my book Undergraduate Convexity.

A vitamin pill P is produced using two ingredients M_1 and M_2. The pill needs to satisfy four constraints for the vital vitamins V_1 and V_2. It must contain at least 6 milligrams and at most 15 milligrams of V_1 and at least 5 milligrams and at most 12 milligrams of V_2. The ingredient M_1 contains 3 milligrams of V_1 and 2 milligrams of V_2 per gram. The ingredient M_2 contains 2 milligrams of V_1 and 3 milligrams of V_2 per gram:

	V_1	V_2
M_1	3	2
M_2	2	3

Let x denote the amount of M_1 and y the amount of M_2 (measured in grams) in the production of a vitamin pill. Write down a system of linear inequalities in x and y describing the constraints above.

We want a vitamin pill of minimal weight satisfying the constraints. How many grams of M_1 and M_2 should we mix?

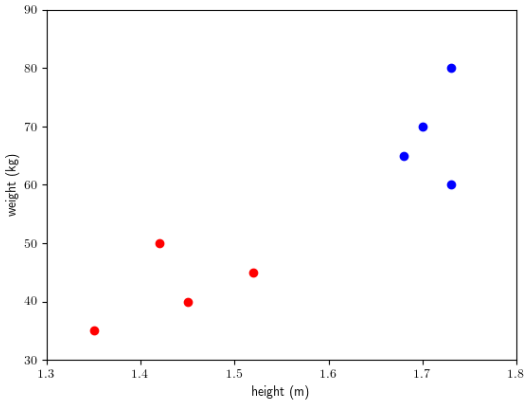
Use Fourier-Motzkin elimination to solve this problem.

Check your solution by modifying the Python code in Example 4.36. Here the objective needs no sign change, since `linprog` minimizes by default and we are minimizing the weight $x+y$ — but each $\geq$ constraint must be multiplied by -1 to fit the $\leq$ form `A_ub`, `b_ub`. ♠

4.6 Application in machine learning and data science

To start with, consider a toy example of a machine learning problem: we wish to tell the gender of a person based on a data point consisting of the height and weight of the person.

To do this we train our model by measuring the height and weight of a lot of people. Each of these measured data points are labeled female or male according to the gender of the person.



Given a new data point, we wish to tell if the person is female or male. Here we consider a very simple model for doing this. First we need to introduce some new mathematical terms. We will introduce the terms generally for data points in $\mathbb{R}^d$ and not just in $\mathbb{R}^2$ as above.

(4.40) DEFINITION.

A hyperplane in $\mathbb{R}^d$ is defined as a subset

$$H = \{v \in \mathbb{R}^d \mid a^\top v + b = 0\}$$

where $a \in \mathbb{R}^d$ is a non-zero vector (called a normal vector of H) and b is a number. A hyperplane divides $\mathbb{R}^d$ into two subsets: the points above the hyperplane satisfying $a^\top v + b > 0$ and the ones below the hyperplane satisfying $a^\top v + b < 0$.

(4.41) EXAMPLE.

In $\mathbb{R}^2$ a hyperplane is a line. The line $y = 2x + 1$ is the hyperplane

$$H = \{v \in \mathbb{R}^2 \mid a^\top v + b = 0\},$$

where

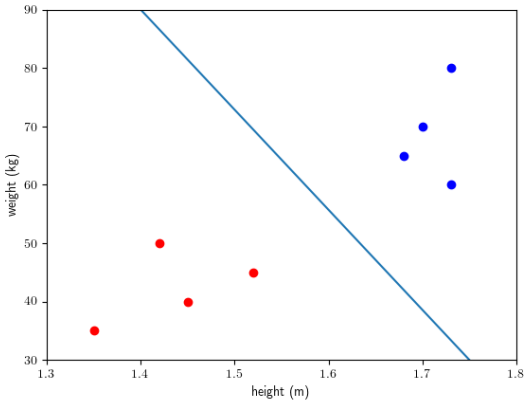
$$a = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \qquad \text{and} \qquad b = 1.$$

The points $(0,1)$ and $(1,3)$ are on the hyperplane. The point $(0,0)$ satisfies $a^\top v + b = 1 > 0$, so it is *above* the hyperplane in the sense of Definition 4.40 — even though it lies geometrically below the line $y = 2x + 1$ in a drawing! Above and below depend on the choice of a normal vector a. Had we picked

$$a = \begin{pmatrix} -2 \\ 1 \end{pmatrix} \qquad \text{and} \qquad b = -1$$

(the same hyperplane), then the point $(0,0)$ would have been below the hyperplane. ♠

Suppose we are given a data set as a finite set of points in $\mathbb{R}^d$ and that each of these points are labeled with either a blue or a red color. We wish to find a hyperplane, such that the blue points are above and the red points are below the hyperplane.



We may then use the hyperplane to predict the label of a point. This could be gender, if you win or lose money buying a stock, anything with a binary classifier.

4.6.1 Formulation as a linear optimization problem

Suppose that the points labeled blue are $x_1,\dots,x_m \in \mathbb{R}^d$ and the points labeled red are $y_1,\dots,y_n \in \mathbb{R}^d$. Then we wish to find $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$, such that

$$a^\top x_i > b$$

for $i = 1,\dots,m$ and

$$a^\top y_j < b$$

for $j = 1,\dots,n$. One can show that these strict inequalities may be solved for a and b if and only if the inequalities

$$a^\top x_i \geq b+1$$
$$a^\top y_j \leq b-1$$

are solvable for a and b, where $i = 1,\dots,m$ and $j = 1,\dots,n$.

It is, however, not realistic to expect data to behave this nicely. Instead one invents the rather ingenious linear optimization problem

$$\min_{\substack{a^\top x_i+u_i \geq b+1,\ a^\top y_j - v_j \leq b-1 \\ u_i \geq 0,\ v_j \geq 0}} \frac{1}{m}(u_1+\cdots+u_m)+\frac{1}{n}(v_1+\cdots+v_n) \tag{4.14}$$

for $i = 1,\dots,m$ and $j = 1,\dots,n$. This linear optimization problem has optimal value zero if and only if data can be separated strictly. Otherwise, it finds a hyperplane minimizing the mean errors for the points involved.

The linear optimization problem (4.14) may look unrelated to the real world, but it has been used very successfully in the diagnosis and prognosis of breast cancer. See Mangasarian et al.

In the python cell below we have implemented the solution of the linear optimization problem (4.14), where the output is a graphical illustration of the optimal line, that separates the points in `xpts` and `ypts` with the smallest mean error as defined in the function to be minimized in (4.14).

python cell — not displayed in the pdf version.

(4.42) EXERCISE.

Run the cell above and look at the printed optimal value. It is positive — so by the discussion above, the two point sets cannot be separated strictly. Find the offending points in the data (look at the red point $(3,2)$ and the blue points $(3,1)$ and $(3,5)$; why is a strict separation impossible?). Then edit the data to make the two sets separable, rerun, and check that the optimal value becomes 0. ♠

5 EUCLIDEAN VECTOR SPACES

Big data are made up of many numbers in data sets. Such data sets can be represented as vectors in a high dimensional euclidean vector space. A vector is nothing but a list of numbers, but we need to talk mathematically about the size of a vector and perform operations on vectors. The term euclidean refers to vectors with a dot product as known from the plane $\mathbb{R}^2$.

The purpose of this chapter is to set the stage for this, especially by introducing the dot product (or inner product) for general vectors. Having a dot product is immensely useful and we give several applications like linear regression and the perceptron learning algorithm.

In the last part of the chapter we will list rudimentary basics of analysis starting with bounded, open, closed and compact subsets of euclidean spaces leading to continuous functions and the socalled extreme value theorem, Theorem 5.84. This result states that a huge class of optimization problems always have a solution.

This chapter is also where machine learning begins in earnest. The recipe behind a surprising amount of modern machine learning is: turn your data into vectors, then apply geometry. Measuring distances between vectors gives the nearest neighbor algorithm, the dot product powers the perceptron — the ancestor of modern neural networks — least squares is the mathematics behind linear regression, and the angle between vectors is used to compare texts in natural language processing. All four appear below as live Python code that you are meant to run, break and modify.

5.1 Vectors in the plane

The dot product (or inner product) between two vectors $u, v \in \mathbb{R}^2$ is given by

$$u \cdot v = x_1 x_2 + y_1 y_2, \tag{5.1}$$

where

$$u = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \qquad \text{and} \qquad v = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}. \tag{5.2}$$

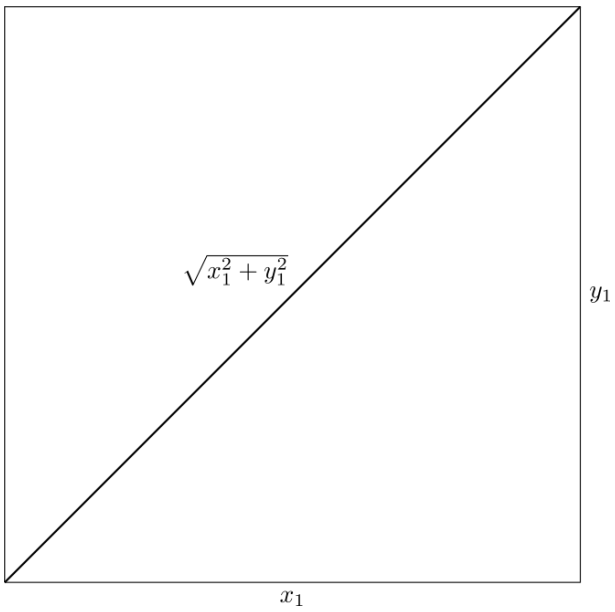
We may also interpret u and v as 2×1 matrices (or column vectors). Then the dot product in (5.1) may be realized as the matrix product:

$$u \cdot v = u^\top v.$$

The length or *norm* of the vector $u \in \mathbb{R}^2$ is given by

$$|u| = \sqrt{u \cdot u} = \sqrt{u^\top u} = \sqrt{x_1^2 + y_1^2}.$$

This follows from the Pythagorean theorem:



The distance $d(u,v)$ between the two vectors u and v is given by

$$d(u,v) = |u-v| = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

Also, the cosine of the angle θ between u and v is given by

$$\cos(\theta) = \frac{u\cdot v}{|u|\,|v|} \qquad \text{or} \qquad u\cdot v = |u|\,|v|\cos(\theta).$$

We will not go into this formula. It is a byproduct of considering the projection of a vector on another vector (see Exercise 5.6).

All of these rather natural notions in the plane $\mathbb{R}^2$ generalize naturally to $\mathbb{R}^d$ for $d > 2$.

5.2 Higher dimensions

We denote the set of column vectors with d rows by $\mathbb{R}^d$ and call it the euclidean vector space of dimension d. An element $v \in \mathbb{R}^d$ is called a vector and it has the form (column vector with d entries)

$$v = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}.$$

A vector in $\mathbb{R}^d$ is a model for a data set in real life. A collection of d numbers, which could signify d measurements. You will see an example of this below, where a vector represents a data set counting words in a string.

Being column vectors, vectors in $\mathbb{R}^d$ can be added and multiplied by numbers:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{pmatrix} = \begin{pmatrix} x_1+y_1 \\ x_2+y_2 \\ \vdots \\ x_d+y_d \end{pmatrix} \qquad \lambda \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = \begin{pmatrix} \lambda x_1 \\ \lambda x_2 \\ \vdots \\ \lambda x_d \end{pmatrix}.$$

The dot product generalizes as follows to higher dimensions.

5.2.1 Dot product, norm and cosine

(5.1) DEFINITION.

Suppose that

$$u = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \qquad \text{and} \qquad v = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{pmatrix}$$

are vectors in $\mathbb{R}^d$.

1. *The dot product between u and v is defined by*

$$u \cdot v = u^\top v = x_1 y_1 + x_2 y_2 + \cdots + x_d y_d. \tag{5.3}$$

2. *Two vectors* $u, v \in \mathbb{R}^d$ *are called* orthogonal *if* $u \cdot v = 0$. *We write this as* $u \perp v$.

3. *The norm of* $u \in \mathbb{R}^d$ *is defined by*

$$|u| = \sqrt{u \cdot u} = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}. \tag{5.4}$$

4. *The distance between the two vectors u and v is defined by*

$$d(u,v) = |u - v| = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}.$$

5. *The cosine of the angle between u and v is defined by*

$$\frac{u \cdot v}{|u||v|}$$

provided that they both are non-zero.

All of the definitions above are present in modern machine learning frameworks. Below we see their incarnations in the python library `numpy`.

(5.2) EXAMPLE.

python cell — not displayed in the pdf version.

♠

Your first machine learning algorithm

With nothing but the notion of distance we can already build a machine learning algorithm: *nearest neighbor classification*. We are given data points whose labels we know. A new point is classified by finding the labeled point closest to it and copying its label. Nothing is solved and nothing is trained — the data itself is the model. Try moving the new point w below, or add points of your own, and rerun.

python cell — not displayed in the pdf version.

Nearest neighbor classification is no toy: with enough data it is hard to beat, and it is the reason the distance $d(u,v)$ deserves its central place in this chapter. You meet it every time a streaming service suggests a film. In this setting *you* are a vector: if the catalogue holds n films, a viewer is represented as a point in $\mathbb{R}^d$ whose i-th coordinate is 1 if the viewer watched film number i to the very end and 0 otherwise. With millions of subscribers this gives millions of points in a space of enormous dimension, and two viewers lie close in the distance $d(u,v)$ precisely when they have watched largely the same films. When the service wonders whether to suggest a particular film to you, the picture becomes exactly the code above: the points are the other viewers, labeled blue if they watched that film to the very end and red if they gave up along the way or never watched it at all. *You* are the new unlabeled point w. The service finds the viewers closest to you and copies their label: if your nearest neighbors stayed to the end, the film lands on your front page. Real services refine this picture in two ways: they first compress the enormous viewing-history vectors into learned *taste vectors* with a few hundred coordinates, and they often measure closeness by the angle between vectors instead of the distance — the cosine similarity you will meet later in this chapter. But the nearest neighbor search itself remains the core, and the mathematics is exactly what you just ran; only the dimension is larger.

(5.3) EXERCISE.

Show that

$$u \perp u \iff u = 0,$$

where $u \in \mathbb{R}^d$. ♠

(5.4) EXERCISE.

Use the definition in (5.3) to show that

$$\begin{aligned} u \cdot (v+w) &= u \cdot v + u \cdot w \\ (\lambda u) \cdot v &= u \cdot (\lambda v) = \lambda (u \cdot v) \end{aligned}$$

for $u, v, w \in \mathbb{R}^d$ and $\lambda \in \mathbb{R}$. ♠

(5.5) EXERCISE.

Let $u \in \mathbb{R}^d$ be a nonzero vector and $\lambda \in \mathbb{R}$. Use the definition in (5.4) to show that $|\lambda u| = |\lambda|\,|u|$ and that

$$\frac{1}{|u|} u$$

is a unit vector.

Hint: You could perhaps use Exercise 5.4 to do this. Notice also that $|\lambda|$ is the absolute value for λ if $\lambda \in \mathbb{R}$. ♠

(5.6) EXERCISE.

Given two vectors $u, v \in \mathbb{R}^d$ with $v \neq 0$, find $\lambda \in \mathbb{R}$, such that $u - \lambda v$ and v are orthogonal, i.e.

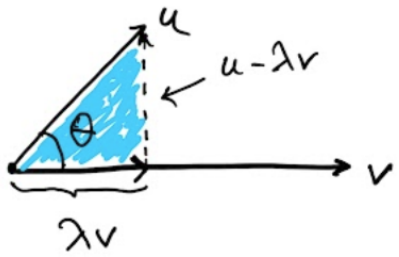
$$(u - \lambda v) \cdot v = 0.$$

Hint:

$$(u - \lambda v) \cdot v = 0 \iff (u \cdot v) - \lambda (v \cdot v) = 0.$$

This is an equation, where λ is unknown!

For $d = 2$, it is sketched below that if $u - \lambda v$ and v are orthogonal, then $u, \lambda v$ and $u - \lambda v$ are the sides in a right triangle.



In this case, if θ is the angle between u and v, show that

$$|u|\cos(\theta) = |v|\,\lambda.$$

Use this to show that

$$u \cdot v = |u|\,|v|\cos(\theta).$$

Finally show that

$$\cos(A - B) = \cos(A)\cos(B) + \sin(A)\sin(B),$$

where A and B are two angles.

Hint: In the last question, you could use that the vectors

$$\begin{pmatrix}\cos(A)\\ \sin(A)\end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix}\cos(B)\\ \sin(B)\end{pmatrix}$$

are unit vectors. ♠

(5.7) EXERCISE.

Given two vectors $u, v \in \mathbb{R}^d$, solve the minimization problem

$$\min_{\lambda \in \mathbb{R}} |u - \lambda v|.$$

Hint: First convince yourself that λ minimizes $|u - \lambda v|$ if and only if it minimizes

$$(u - \lambda v) \cdot (u - \lambda v) = |v|^2 \lambda^2 - 2(u \cdot v)\lambda + |u|^2,$$

which happens to be a quadratic polynomial in λ. ♠

5.3 The unreasonable effectiveness of the dot product

(5.8) QUIZ.

quiz — not displayed in the pdf version.

♠

5.3.1 The dist formula from high school

The infamous dist formula from high school says that the distance from the point (x_1, y_1) to the line given by $y = ax + b$ is

$$\frac{|ax_1 + b - y_1|}{\sqrt{a^2 + 1}}. \tag{5.5}$$

Where does this magical formula come from? Consider a general line L in parametrized form (see Definition 4.12)

$$L = \{u + tv \mid t \in \mathbb{R}\} \subseteq \mathbb{R}^d.$$

If $w \in \mathbb{R}^d$, then the distance from w to L is given by the solution t_0 to the optimization problem

$$\min_{t \in \mathbb{R}} (w - (u + tv)) \cdot (w - (u + tv)). \tag{5.6}$$

This looks scary, but simply boils down to finding the top of a parabola. The solution is

$$t_0 = \frac{v \cdot w - u \cdot v}{v \cdot v}$$

and the point on L closest to w is $u + t_0 v$.

Now we put (see Example 4.13)

$$u = \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad v = \begin{pmatrix} 1 \\ a \end{pmatrix} \quad \text{and} \quad w = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

in order to derive (5.5). The solution to (5.6) becomes

$$t_0 = \frac{x_1 + ay_1 - ab}{1 + a^2}.$$

We must compute the distance D from w to $u + t_0 v$ in this case. The distance squared is

$$\begin{aligned} D^2 &= |w - (u + t_0 v)|^2 = (w - (u + t_0 v)) \cdot (w - (u + t_0 v)) \\ &= (x_1 - t_0)^2 + (y_1 - b - t_0 a)^2. \end{aligned}$$

This is a mouthful and I have to admit that I used symbolic software (see below) to verify that

$$D^2 = \frac{a^2 x_1^2 + 2abx_1 - 2ax_1 y_1 + b^2 - 2by_1 + y_1^2}{1 + a^2} = \frac{(ax_1 + b - y_1)^2}{1 + a^2}.$$

python cell — not displayed in the pdf version.

5.3.2 The perceptron algorithm

Already at this point we have the necessary definitions for explaining the perceptron algorithm. This is one of the early algorithms of machine learning. It aims at finding a high dimensional line (hyperplane) that separates data organized in two clusters. In terms of the dot product, the idea of the algorithm is described below in dimension two.

A line in the plane is given by an equation

$$ax+by+c=0$$

for $a,b,c\in\mathbb{R}$, where $(a,b)\neq(0,0)$. Given finitely many points

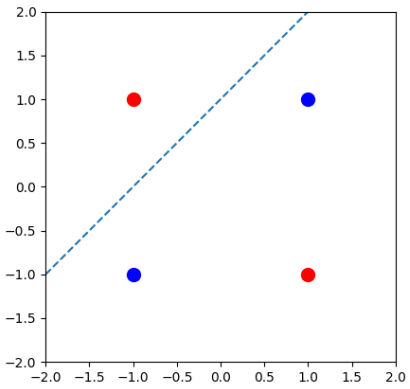
$$v_1=(x_1,y_1),v_2=(x_2,y_2),\ldots,v_n=(x_n,y_n)$$

each with a label $\ell_1,\ldots,\ell_n$ of ±1 (or blue and red for that matter), we wish to find a line (given by a,b,c), such that

$$\begin{aligned} ax_i+by_i+c>0 \quad &\text{if} \quad \ell_i=1\\ ax_i+by_i+c<0 \quad &\text{if} \quad \ell_i=-1 \end{aligned}$$

for $i=1,\ldots,n$. Such a line is called a separating line for the labeled points.

In some cases this is impossible (an example is illustrated below).



(5.9) EXERCISE.

Show that it is impossible to find a line separating the red and blue points above. The red points are $(-1,1)$ and $(1,-1)$. The blue points are $(-1,-1)$ and $(1,1)$. ♠

A clever approach to finding such a line, if it exists, is to reformulate the problem by looking at the vectors given by

$$\hat{v}_1=(\ell_1x_1,\ell_1y_1,\ell_1),\qquad \hat{v}_2=(\ell_2x_2,\ell_2y_2,\ell_2),\qquad \ldots\qquad ,\qquad \hat{v}_n=(\ell_nx_n,\ell_ny_n,\ell_n) \tag{5.7}$$

in $\mathbb{R}^3$. Then the existence of the line is equivalent to the existence of a vector $\alpha\in\mathbb{R}^3$ with $\alpha\cdot\hat{v}_i>0$ for $i=1,\ldots,n$. If $\alpha=(\alpha_1,\alpha_2,\alpha_3)$ is such a vector, then we have for $i=1,\ldots,n$,

$$\alpha_1x_i+\alpha_2y_i+\alpha_3>0 \quad \text{if} \quad \ell_i=1 \tag{5.8}$$

$$-\alpha_1x_i-\alpha_2y_i-\alpha_3>0 \quad \text{if} \quad \ell_i=-1. \tag{5.9}$$

Therefore we may take $a=\alpha_1,b=\alpha_2$ and $c=\alpha_3$ as the line.

A ridiculously simple algorithm

In view of the approach introduced in (5.8), the following general question is interesting.

Given finitely many vectors $v_1, \dots, v_m \in \mathbb{R}^d \setminus \{0\}$, can we find $\alpha \in \mathbb{R}^d$, such that

$$\alpha \cdot v_i > 0$$

for every $i = 1, \dots, m$?

(5.10) EXERCISE.

Come up with a simple example, where this problem is unsolvable i.e., come up with vectors $v_1, \dots, v_n \in \mathbb{R}^d$, where such an α does not exist.

Hint. Try out some simple examples for $d = 1$ and $d = 2$.

♠

In case α exists, the following ridiculously simple algorithm works in computing α. It is called the *perceptron (learning) algorithm*.

(i) Begin by putting $\alpha = 0$.

(ii) If there exists $v_i \in \{v_1, \dots, v_m\}$ with $\alpha \cdot v_i \leq 0$, then replace α by $\alpha + v_i$ and repeat this step. Otherwise α is the desired output vector.

(5.11) EXAMPLE.

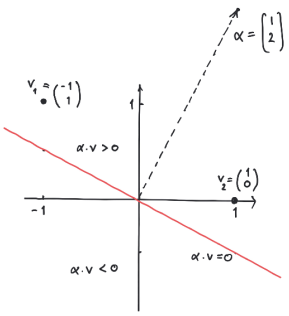
Let us try out the algorithm on the simple example of just two points in $\mathbb{R}^2$ given by

$$v_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \qquad \text{and} \qquad v_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

In this case the algorithm proceeds as pictured below.

$$\alpha = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \xrightarrow{+v_1} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \xrightarrow{+v_2} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \xrightarrow{+v_2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \xrightarrow{+v_1} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \xrightarrow{+v_2} \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

It patiently crawls its way ending with the vector $\alpha = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, which satisfies $\alpha \cdot v_1 > 0$ and $\alpha \cdot v_2 > 0$.



♠

Let us see how (5.7) works in a concrete example.

(5.12) EXAMPLE.

Consider the points

$$v_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad v_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \text{and} \qquad v_3 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

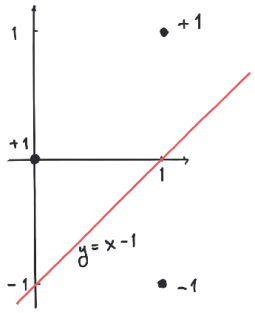
in $\mathbb{R}^2$, where v_1 and v_2 are labeled by $+1$ and v_3 is labeled by -1. Then we let

$$\hat{v}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \qquad \hat{v}_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \text{and} \qquad \hat{v}_3 = \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix}.$$

Now we run the simple algorithm above Example 5.11:

$$\hat{\alpha} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{+\hat{v}_1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \xrightarrow{+\hat{v}_3} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \xrightarrow{+\hat{v}_1} \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}.$$

From the last vector we see that $x - y - 1 = 0$ determines a line separating the labeled points.



♠

Below is an implementation of the perceptron (learning) algorithm in python (with numpy) with input from Example 5.12 (it also works in higher dimensions).

python cell — not displayed in the pdf version.

The algorithm comes alive when you see the line it finds. Below the perceptron runs on two larger clusters and the separating line $ax + by + c = 0$ it computes is drawn through the data. Move the points — or make the clusters overlap and watch what happens to the patiently crawling algorithm (the code gives up after ten thousand rounds; the theorem below explains why it would otherwise never stop).

python cell — not displayed in the pdf version.

(5.13) EXERCISE.

Consider the points

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \text{and} \qquad \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

in $\mathbb{R}^2$, where the first point is labeled with -1 and the rest by 1. Use the perceptron algorithm to compute a separating hyperplane.

What happens when you run the perceptron algorithm on the above points, but where the label of

$$\begin{pmatrix}1\\1\end{pmatrix}$$

is changed from 1 to -1? ♠

5.3.3 Why does the perceptron algorithm work?

We will assume that there exists $\alpha \in \mathbb{R}^d$, such that

$$\alpha \cdot v_i > 0$$

for every $i = 1,\dots,m$. Therefore $\mu = \min(\alpha \cdot v_1,\dots,\alpha \cdot v_m) > 0$ and if we put

$$\alpha^* = \frac{1}{\mu}\alpha, \tag{5.10}$$

then $\alpha^* \cdot v_i \geq 1$ for every $i = 1,\dots,m$.

The basic insight is the following

(5.14) PROPOSITION.

Let $r = \max\{|v_1|,\dots,|v_m|\}$. After k iterations of the perceptron algorithm, α satisfies

$$\alpha \cdot \alpha^* \geq k \qquad \text{and} \qquad kr^2 \geq |\alpha|^2,$$

where α^* is defined in (5.10).

Proof. The algorithm starts with $\alpha = 0$. In the second step we update α to $\alpha + v_i$ if $\alpha \cdot v_i \leq 0$. For such a v_i we have the following inequalities

$$(\alpha + v_i) \cdot \alpha^* = \alpha \cdot \alpha^* + v_i \cdot \alpha^* \geq \alpha \cdot \alpha^* + 1$$

and

$$(\alpha + v_i) \cdot (\alpha + v_i) = |\alpha|^2 + 2v_i \cdot \alpha + |v_i|^2 \leq |\alpha|^2 + |v_i|^2 \leq |\alpha|^2 + r^2.$$

If the second step of the algorithm is executed after k steps, then we get for the new $\alpha + v_i$ that

$$(\alpha + v_i) \cdot \alpha^* \geq \alpha \cdot \alpha^* + 1 \geq k + 1 \qquad \text{and} \qquad |\alpha + v_i|^2 \leq |\alpha|^2 + r^2 \leq kr^2 + r^2 = (k+1)r^2.$$

□

Proposition 5.14 implies that

$$k \leq |\alpha|\,|\alpha^*| \leq \sqrt{k}r\,|\alpha^*|.$$

Therefore we get $k \leq r^2\,|\alpha^*|^2$ and there is an upper bound on the number of iterations used in the second step. So after a finite number of steps, we must have $\alpha \cdot v_i > 0$ for every $i = 1,\dots,m$.

5.4 Pythagoras and the least squares method

The result below is a generalization of the theorem of Pythagoras about right triangles to higher dimensions.

(5.15) PROPOSITION.

If $u, v \in \mathbb{R}^d$ and $u \perp v$, then

$$|u+v|^2 = |u|^2 + |v|^2 .$$

Proof. This follows from

$$(u+v)\cdot(u+v) = u\cdot u + u\cdot v + v\cdot u + v\cdot v = u\cdot u + v\cdot v = |u|^2 + |v|^2 ,$$

since $u \cdot v = v \cdot u = 0$. □

The dot product and the norm have a vast number of applications. One of them is the method of least squares: suppose that you are presented with a system

$$Av = b \tag{5.11}$$

of linear equations, where A is an $m \times d$ matrix.

You may not be able to solve (5.11). There could be for example 17 equations and only 2 unknowns making it impossible for all the equations to hold. As an example, the system

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} \tag{5.12}$$

of three linear equations and two unknowns does not have any solutions.

The method of (linear) least squares seeks the best approximate solution v_0 to (5.11) as a solution to the minimization problem

$$\min_{v \in \mathbb{R}^d} |b - Av|^2 . \tag{5.13}$$

There is a surprising way of finding optimal solutions to (5.13):

(5.16) THEOREM.

If $v_0 \in \mathbb{R}^d$ is a solution to the system

$$(A^\top A)v = A^\top b \tag{5.14}$$

of d linear equations with d unknowns, then v_0 is an optimal solution to (5.13). If v_0 on the other hand is an optimal solution to (5.13), then v_0 is a solution to (5.14).

Proof. Suppose we know that $b - Av_0$ is orthogonal to Aw for every $w \in \mathbb{R}^d$. Then

$$|b - Av|^2 = |b - Av_0 + A(v_0 - v)|^2 = |b - Av_0|^2 + |A(v - v_0)|^2$$

for every $v \in \mathbb{R}^d$ by Proposition 5.15. So, in the case that $b - Av_0 \perp Aw$ for every $w \in \mathbb{R}^d$ we have

$$|b - Av|^2 \geq |b - Av_0|^2$$

for every $v \in \mathbb{R}^d$ proving that v_0 is an optimal solution to (5.13).

Now we wish to show that $b - Av_0$ is orthogonal to Aw for every $w \in \mathbb{R}^d$ if and only if $A^\top Av_0 = A^\top b$. This is a computation involving the matrix arithmetic introduced in Chapter 3:

$$\begin{aligned}(b - Av_0) \cdot Aw &= \\ (b - Av_0)^\top Aw &= \\ b^\top Aw - v_0^\top A^\top Aw &= \\ (b^\top A - v_0^\top A^\top A)w &= 0\end{aligned}$$

for every $w \in \mathbb{R}^d$ if and only if $b^\top A - v_0^\top A^\top A = 0$. But

$$(b^\top A - v_0^\top A^\top A)^\top = A^\top b - A^\top Av_0$$

so that $(A^\top A)v_0 = A^\top b$.

On the other hand, if $|b - Av_0|^2 \leq |b - Av|^2$ for every $v \in \mathbb{R}^d$, then $(b - Av_0) \cdot Aw = 0$ for every $w \in \mathbb{R}^d$: if we could find w with $(b - Av_0) \cdot Aw < 0$, then

$$|b - A(v_0 - \varepsilon w)|^2 < |b - Av_0|^2$$

for a small number $\varepsilon > 0$. This follows, since

$$|b - A(v_0 - \varepsilon w)|^2 = ((b - Av_0) + \varepsilon Aw)^2,$$

which is

$$|b - Av_0|^2 + 2\varepsilon(b - Av_0) \cdot Aw + \varepsilon^2(Aw)^2$$

By picking $\varepsilon > 0$ sufficiently small,

$$\varepsilon(2(b - Av_0) \cdot Aw + \varepsilon(Aw)^2) < 0.$$

□

In a future course on linear algebra you will see that the system of linear equations in Theorem 5.16 is always solvable i.e., an optimal solution to (5.13) can always be found in this way.

(5.17) EXERCISE.

Show that (5.12) has no solutions. Compute the best approximate solution to (5.12) using Theorem 5.16. ♠

(5.18) EXAMPLE.

The classical application of the least squares method is to find the best line $y = \alpha x + \beta$ through a given set of points

$$(x_1, y_1), \quad (x_2, y_2), \quad \ldots \quad , (x_n, y_n)$$

in the plane $\mathbb{R}^2$.

Usually we cannot find a line matching the points precisely. This corresponds to the fact that the system of equations

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

has no solutions.

Working with the least squares solution, we try to compute the best line $y = \alpha x + \beta$ in the sense that

$$(y_1 - \alpha x_1 - \beta)^2 + (y_2 - \alpha x_2 - \beta)^2 + \cdots + (y_n - \alpha x_n - \beta)^2$$

is minimized.

(5.19) FIGURE.

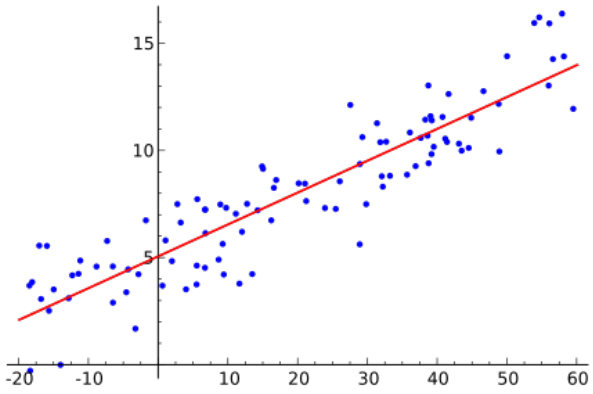


Figure 5.1: Best fit of line to random points from Wikipedia.

We might as well have asked for the best quadratic polynomial

$$y = \alpha x^2 + \beta x + \gamma$$

passing through the points

$$(x_1, y_1), \quad (x_2, y_2), \quad \ldots \quad , (x_n, y_n)$$

in $\mathbb{R}^2$.

The same method gives us the system

$$\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \\ x_n^2 & x_n & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

of linear equations.

(5.20) FIGURE.

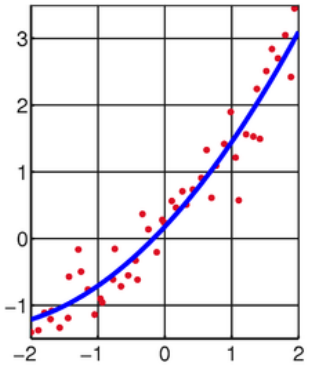


Figure 5.2: Best fit of quadratic polynomial to random points from Wikipedia.

The method generalizes naturally to finding the best polynomial of degree m

$$y = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$

through a given set of points. ♠

In machine learning the least squares method goes by the name *linear regression*, and it is the first model most data scientists reach for. The code below carries out the whole computation for the line fit: build the matrix A and the vector b, solve the normal equations $(A^\top A)x = A^\top b$ of Theorem 5.16, and plot the result. Change the points, add noise, add more points — the three lines of code in the middle stay exactly the same; only the data changes.

python cell — not displayed in the pdf version.

(5.21) EXERCISE.

Find the best line $y = \alpha x + \beta$ through the points $(1,2),(2,1)$ and $(4,3)$ and the best quadratic polynomial $y = ax^2 + bx + c$ through the points $(-2,2),(-1,1),(0,0),(1,1)$ and $(2,2)$.

It is important here, that you write down the relevant system of linear equations according to Theorem 5.16. It is however ok to solve the equations on a computer (or check your best fit on WolframAlpha).

Also, you can get a graphical illustration of your result in the python cell below.

python cell — not displayed in the pdf version.

♠

Lines can also be fitted to data they have no business fitting. The next exercise is a small experiment whose failure is the whole point — remember it when you reach Chapter 7.

(5.22) EXERCISE.

Eight students studied x hours for an exam and either failed ($y = 0$) or passed ($y = 1$):

x	0.5	1	1.5	2	3	4	5	6
y	0	0	0	0	1	1	1	1

Find the best line $y = \alpha x + \beta$ through these eight points (set up the normal equations from Theorem 5.16; solving them on a computer is fine — the cell below plots your line over the data). It is tempting to read the line's value at x as "the chance of passing after x hours of study". Compute the line's value at $x = 6$ and at $x = 0$. Why do the two numbers make that reading absurd? Something better is needed for yes/no data — that something is the sigmoid function, and it arrives in Chapter 7 where these eight students return.

python cell — not displayed in the pdf version.

♠

(5.23) EXERCISE.

A circle with center (a,b) and radius r is given by the equation

$$(x-a)^2 + (y-b)^2 = r^2. \tag{5.15}$$

1. Explain how (5.15) can be rewritten to the equation

$$2ax+2by+c=x^2+y^2, \tag{5.16}$$

where $c=r^2-a^2-b^2$.

2. Explain how fitting a circle to the points $(x_1,y_1),\dots,(x_n,y_n)$ in the least squares context using (5.16) leads to the system

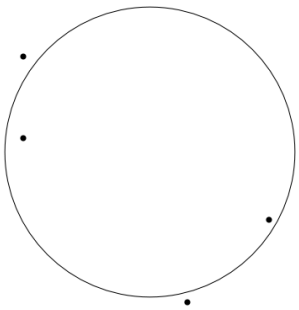
$$\begin{pmatrix} 2x_1 & 2y_1 & 1 \\ 2x_2 & 2y_2 & 1 \\ \vdots & \vdots & \vdots \\ 2x_n & 2y_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} x_1^2+y_1^2 \\ x_2^2+y_2^2 \\ \vdots \\ x_n^2+y_n^2 \end{pmatrix},$$

of linear equations.

3. Compute the best circle through the points

$$(0,2),\quad (0,3),\quad (2,0)\quad \text{and}\quad (3,1)$$

by giving the center coordinates and radius with two decimals. Use the python cell below to plot your result too see if it matches the drawing.



python cell — not displayed in the pdf version.

♠

5.5 The Cauchy-Schwarz inequality

Take another look at 5 in Definition 5.1. It is actually a small miracle that no matter which (non-zero) vectors u and v you use as input to the `cosine` function defined in Example 5.2, you always get a number between -1 and 1. The mathematics behind this is rather elegant. It is a consequence of the famous Cauchy-Schwarz inequality stated and proved below.

(5.24) THEOREM.

For two vectors $u,v\in\mathbb{R}^d$,

$$|u\cdot v|\le |u|\,|v|\,.$$

Proof. If $u=0$ both sides are 0, so we may assume $u\neq 0$. We consider the function $q:\mathbb{R}\to\mathbb{R}$ given by

$$q(x)=(xu+v)\cdot(xu+v)=|u|^2x^2+2(u\cdot v)x+|v|^2$$

Then $q(x)$ is a quadratic polynomial with $q(x) \geq 0$. Therefore its discriminant must be ≤ 0 i.e.,

$$4(u \cdot v)^2 - 4\,|u|^2\,|v|^2 \leq 0,$$

which gives the result. □

(5.25) EXERCISE.

Why are the two inequalities

$$-1 \leq \frac{u \cdot v}{|u||v|} \leq 1$$

a consequence of Theorem 5.24? ♠

(5.26) EXERCISE.

For arbitrary two numbers $x, y \in \mathbb{R}$,

$$2(x^2+y^2) \geq (x+y)^2,$$

since

$$2(x^2+y^2) - (x+y)^2 = x^2+y^2-2xy = (x-y)^2 \geq 0.$$

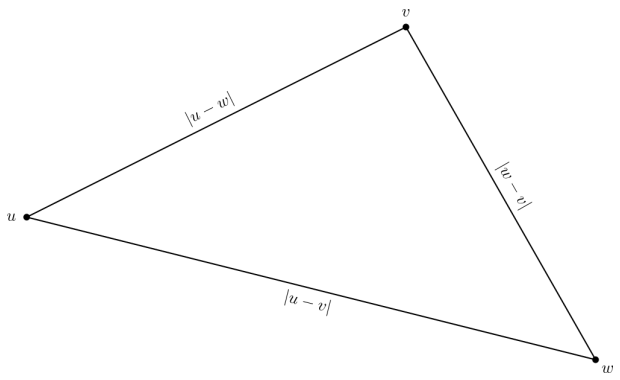
Why is

$$n(x_1^2+\cdots+x_n^2) \geq (x_1+\cdots+x_n)^2$$

for arbitrary n numbers $x_1, \ldots, x_n \in \mathbb{R}$? ♠

5.5.1 The triangle inequality

Another nice consequence of the Cauchy-Schwarz inequality is the triangle inequality.



(5.27) COROLLARY (Triangle inequality).

For three vectors $u, v, w \in \mathbb{R}^d$,

$$d(u,w) \leq d(u,v) + d(v,w).$$

Proof. From the Cauchy-Schwarz inequality (Theorem 5.24) it follows that

$$|v_1+v_2|^2 = (v_1+v_2)\cdot(v_1+v_2) = |v_1|^2 + 2v_1 \cdot v_2 + |v_2|^2 \leq |v_1|^2 + 2\,|v_1|\,|v_2| + |v_2|^2$$

for two vectors $v_1, v_2 \in \mathbb{R}^d$. Since the right hand side of this inequality is $(|v_1|+|v_2|)^2$, we have

$$|v_1+v_2| \leq |v_1|+|v_2|\,.$$

By the definition of $d(u,w)$, we then get the desired inequality as

$$d(u,w) = |u-w| = |(u-v)+(v-w)| \leq |u-v|+|v-w| = d(u,v)+d(v,w).$$

□

(5.28) EXERCISE.

Apply the triangle inequality in the form

$$|u+v| \leq |u|+|v|$$

for $u, v \in \mathbb{R}^d$ to show that

$$\begin{aligned} ||u|-|v|| &\leq |u-v| \\ ||u|-|v|| &\leq |u+v|\,. \end{aligned}$$

♠

5.5.2 Cosine similarity in machine learning

When two vectors $u, v \in \mathbb{R}^d$ are interpreted as data sets, the number in 5 of Definition 5.1 is known as the *cosine similarity*. It measures how well the two vectors u and v point in the same direction: 1 means perfectly aligned, 0 orthogonal, -1 opposite.

A very primitive way of modelling sentences in a language is the socalled *one-hot encoding* of its words. We will illustrate this by an example. Suppose that our language consists of the words

'a', 'and', 'applicable', 'are', 'fun', 'is', 'mathematics', 'matrices', 'matrix', 'useful'

Each word gets embedded into $\mathbb{R}^{10}$ with a vector associated to its row below

a	1	0	0	0	0	0	0	0	0	0
and	0	1	0	0	0	0	0	0	0	0
applicable	0	0	1	0	0	0	0	0	0	0
are	0	0	0	1	0	0	0	0	0	0
fun	0	0	0	0	1	0	0	0	0	0
is	0	0	0	0	0	1	0	0	0	0
mathematics	0	0	0	0	0	0	1	0	0	0
matrices	0	0	0	0	0	0	0	1	0	0
matrix	0	0	0	0	0	0	0	0	1	0
useful	0	0	0	0	0	0	0	0	0	1

(5.17)

Now consider the two sentences "*mathematics is fun and a matrix is useful*" and "*mathematics is fun and matrices are applicable*".

From the words in the two strings we form the following vectors in $\mathbb{R}^{10}$ using the one-hot embedding in (5.17).

mathematics is fun and a matrix is useful	1	1	0	0	1	2	1	0	1	1
mathematics is fun and matrices are applicable	0	1	1	1	1	1	1	1	0	0

Here a sentence is mapped to the vector, which is the sum of all the vectors corresponding to the words in the sentence, where each vector is multiplied by its multiplicity i.e., how many times the word occurs. The closer the cosine gets to 1 (corresponding to an angle of 0 degrees), the more similar we consider the sentences. Use the python snippet below to experiment and compute the cosine similarity in the example.

python cell — not displayed in the pdf version.

Cosine similarity is all you need to build a tiny search engine. The code below retrieves the sentences most similar to a query from a small collection of documents (run the cell above first, so that `cosinesim` is defined — cells on a page share their variables). Scaled up — with better embeddings and billions of documents — this is how retrieval works in modern search engines and in chatbots that look up sources before answering.

python cell — not displayed in the pdf version.

(5.29) EXERCISE.

Use the cell above to compute the cosine similarity between the two one-word sentences "matrix" and "matrices" — and between "fun" and "boring". Explain the outcome. Is it reasonable? ♠

The exercise exposes the fundamental weakness of one-hot encoding: every pair of different words is orthogonal, so "matrix" and "matrices" are judged exactly as unrelated as "matrix" and "banana". The encoding sees spelling, not meaning.

The bread and butter of modern language models is therefore *dense* embeddings: every word is mapped to a vector with hundreds or thousands of coordinates, *learned* from enormous amounts of text in such a way that words with similar meaning end up as vectors with high cosine similarity. The breakthrough came in 2013, when Google introduced word2vec. Famously, the learned vectors support a strange arithmetic of meaning:

$$v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}.$$

The cell below shows the mechanics on a hand-made miniature: eight words embedded in $\mathbb{R}^5$, where we have designed the coordinates ourselves to mean (royal, male, female, young, food). Real embeddings are learned, not designed, and their individual coordinates mean nothing to a human — but the geometry works the same way. Compute $v_{\text{king}} - v_{\text{man}} + v_{\text{woman}}$ by hand and check that the ranking below makes sense. Notice one deliberate oddity: apple and banana were given *identical* coordinates, so their cosine similarity is exactly 1 and the machine literally cannot tell them apart. In a designed embedding that is a decision someone made; the real, learned embedding a few cells below keeps them close but distinguishable. Then invent new words and coordinates of your own.

python cell — not displayed in the pdf version.

The miniature is a toy, but the real thing is running on this very page: the *Ask the book* feature of the search palette (press Cmd/Ctrl+K) embeds every passage of this book as a learned vector in $\mathbb{R}^{384}$, using a small language model called all-MiniLM-L6-v2, and answers a question by cosine-ranking all passages against it — exactly the tiny search engine you built above, at scale. The vectors travel next to this page in the file `search-vectors.js`, every coordinate stored as an 8-bit integer together with one scaling factor per vector, to keep the download small.

The fold below carries 18 English words embedded by that same model, stored with the same integer trick — open it and you are looking at the actual coordinates: one line per word, 384 small integers each, and a scaling factor per word turning them back into real numbers. Run the cell once; it defines a dictionary `E` holding the 18 learned vectors in $\mathbb{R}^{384}$ and prints one of them.

Python: 18 words as learned vectors in dimension 384.

python cell — not displayed in the pdf version.

Now the strange arithmetic of meaning can be checked on a real learned embedding, not a designed one. One wrinkle: in a learned embedding the vector $v_{\text{king}} - v_{\text{man}} + v_{\text{woman}}$ is still closest to v_{king} itself, so it is standard practice to leave the three words of the query out of the ranking. Nobody designed a coordinate to mean *royal* or *female* here — 384 learned coordinates, individually meaningless, and yet queen wins. Try the other analogies in the comment, or invent your own from the 18 words.

python cell — not displayed in the pdf version.

When embedding text one usually considers tokens and not words: every input to a chatbot is broken into a sequence of tokens from a vocabulary of roughly 50,000–100,000, and each token is embedded into a euclidean space of dimension well above 1000. The cosine similarity you have computed in this section is, quite literally, the operation used when a chatbot searches a document collection for the passages most relevant to your question.

5.5.3 Attention: the soft nearest neighbor

Inside the chatbot itself, the dot product runs an even bigger show. The T in GPT stands for *transformer*, and the central mechanism of a transformer is called *attention*. Here is the problem it solves. A word has one embedded vector, fixed once and for all in a table like the fold above — but meaning is not fixed: "bank" points one way in "the bank of the river" and another in "the bank raised its rates". So the transformer lets every word *update* its vector by blending in the vectors of the words around it — heavily where the connection is strong, lightly where it is weak. Blending vectors is a weighted average. And the strengths? Dot products, of course.

The recipe is short. Call the vector doing the asking the *query* q, and let $v_1, \ldots, v_n$ be the vectors it may consult. Compute the similarity scores s_i between q and each v_i — cosines, as in 5 of Definition 5.1 — and turn the scores into weights by exponentiating and normalizing:

$$w_i = \frac{e^{\beta s_i}}{e^{\beta s_1} + \cdots + e^{\beta s_n}}. \tag{5.18}$$

Every weight is positive and together they sum to 1 (make sure you see why), so

$$w_1 v_1 + \cdots + w_n v_n$$

is an honest weighted average — the updated vector. The exponential recipe (5.18) is called *softmax*, and it is one of the workhorses of machine learning. The number $\beta \geq 0$ is a dial we have added to make the recipe visible; in a real transformer its role is hidden inside learned matrices that reshape every vector before the dot products are taken, but the skeleton is exactly what you see here.

The dial is worth turning, because it explains what attention *is*. At $\beta = 0$ all the weights are $\frac{1}{n}$: the update is the plain average, attention spread evenly over everyone. As β grows, the largest score soaks up more and more of the weight, and in the limit the weighted average simply *becomes* the most similar vector — the nearest neighbor from the beginning of this chapter. Softmax attention is a dial between "average of everything" and "winner takes all": a *soft* nearest neighbor. The cell below turns the 18 embedded words loose on each other. Run it, then change the query and the dial.

python cell — not displayed in the pdf version.

Read the table: `kitten` attends to `cat`, `puppy` and `dog` and all but ignores `france` — and the updated vector has been pulled into the animal corner of $\mathbb{R}^{384}$ (in our run `cat` tops its similarity ranking). This is the whole trick. In a transformer, *every* token of your prompt runs this update against every other token, layer after layer, dozens of times, with learned matrices reshaping the vectors between rounds — and out of nothing but dot

products, exponentials and weighted averages comes a machine that answers your questions. The mathematics you need in order to *train* such machinery is the business of Chapter 7, culminating in Section 7.9 — where the softmax recipe will meet you again.

(5.30) EXERCISE.

Turn the dial in the cell above.

1. Set $\beta = 0$ and rerun. Explain every number in the table before you believe it.
2. Set $\beta = 100$. Which algorithm from the beginning of this chapter has the cell just reproduced?
3. Change the query to `denmark` and find a β where `copenhagen` gets the lion's share of the attention while `france` and `italy` still receive visibly more than `banana`.

♠

(5.31) EXERCISE.

The claims made about the dial deserve proofs. Let $s_1, \ldots, s_n$ be the scores in (5.18).

1. Show that $w_i > 0$ for all i and that $w_1 + \cdots + w_n = 1$, no matter the value of β and the signs of the scores.
2. Suppose $s_1 > s_i$ for all $i \geq 2$. Show that $w_1 \to 1$ as $\beta \to \infty$.

   Hint. For the second part, divide the numerator and the denominator of w_1 by $e^{\beta s_1}$:

$$w_1 = \frac{1}{1 + e^{-\beta(s_1 - s_2)} + \cdots + e^{-\beta(s_1 - s_n)}}.$$

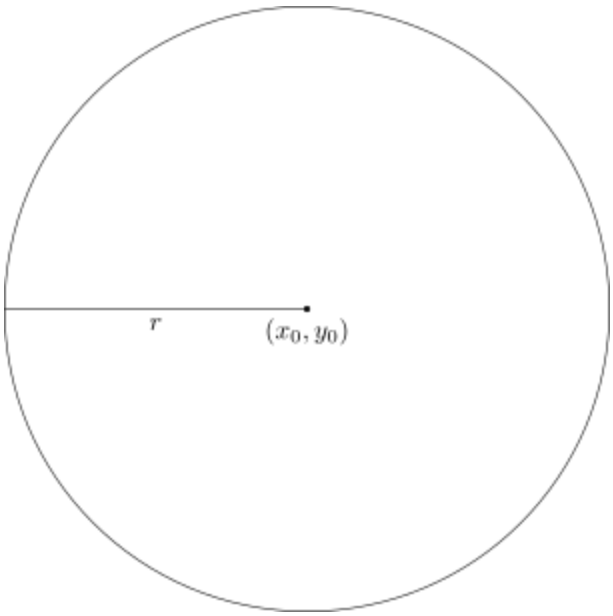
   Every exponent $-\beta(s_1 - s_i)$ tends to $-\infty$ as $\beta \to \infty$, since $s_1 - s_i > 0$.

♠

5.6 Special subsets of euclidean spaces

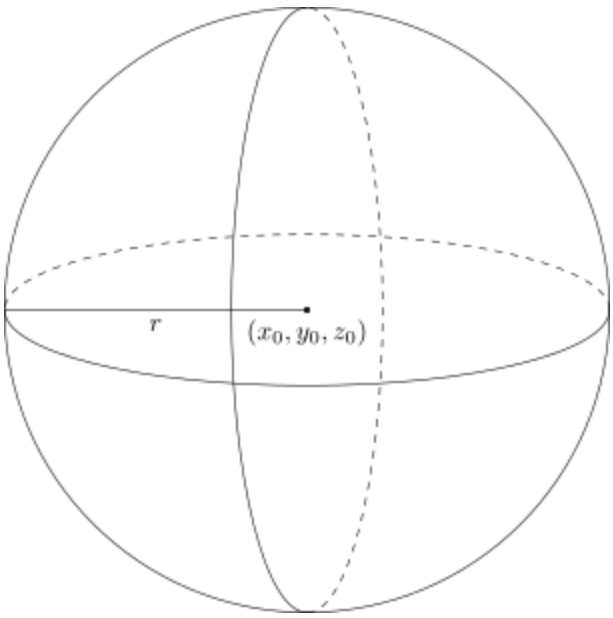
Recall that a circle (or an open disk) centered at $(x_0, y_0) \in \mathbb{R}^2$ with radius $r \in \mathbb{R}$ is defined as the subset

$$\begin{aligned}
&\left\{(x,y) \in \mathbb{R}^2 \middle| (x-x_0)^2 + (y-y_0)^2 < r^2\right\} = \\
&\left\{(x,y) \in \mathbb{R}^2 \middle| \sqrt{(x-x_0)^2 + (y-y_0)^2} < r\right\} = \\
&\left\{(x,y) \in \mathbb{R}^2 \middle| d((x_0,y_0),(x,y)) < r\right\}.
\end{aligned}$$



Similarly an open ball in $\mathbb{R}^3$ centered at $(x_0, y_0, z_0) \in \mathbb{R}^3$ with radius $r \in \mathbb{R}$ is defined as the subset

$$\begin{aligned}
&\left\{(x,y,z) \in \mathbb{R}^3 \middle| (x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 < r^2\right\} = \\
&\left\{(x,y,z) \in \mathbb{R}^3 \middle| \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} < r\right\} = \\
&\left\{(x,y,z) \in \mathbb{R}^3 \middle| d((x_0,y_0,z_0),(x,y,z)) < r\right\}.
\end{aligned}$$



The natural generalization of this definition to higher dimensions is given below.

(5.32) DEFINITION.

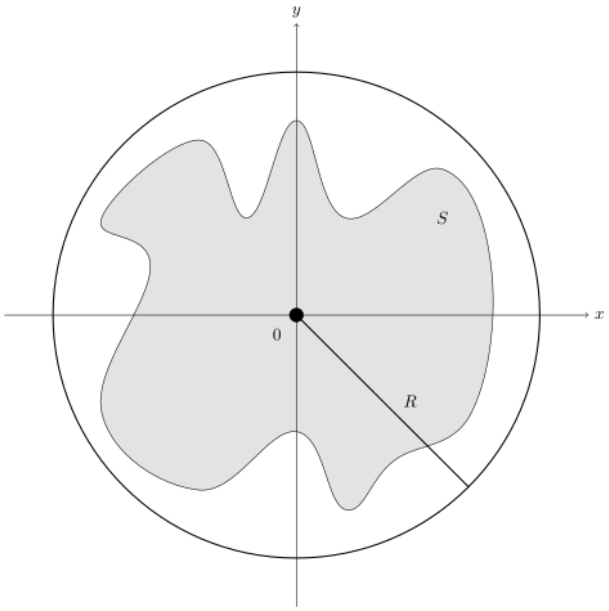
The open ball centered at $u \in \mathbb{R}^d$ with radius $r \in \mathbb{R}$ is defined as

$$B(u,r) = \{v \in \mathbb{R}^d \mid d(u,v) < r\}.$$

Why do we need special classes of subsets? The pay-off comes at the end of this chapter: the extreme value theorem (Theorem 5.84) guarantees that an optimization problem has a solution when the function is continuous and the constraint set is *compact*. This section builds the vocabulary needed to state — and appreciate — that theorem: bounded, open, closed and compact subsets. Pay close attention to the logic of the definitions; the quantifiers $\exists$ and $\forall$ from the first chapter do all the heavy lifting here.

5.6.1 Bounded subsets

A subset of $\mathbb{R}^d$ is bounded if it does not stretch infinitely far away from the origin — it can be caught inside a large enough open ball centered at 0:



(5.33) DEFINITION.

A subset $S \subseteq \mathbb{R}^d$ is called bounded if there exists $R \in \mathbb{R}$, such that

$$S \subseteq B(0,R).$$

(5.34) REMARK (The logic of boundedness).

Spelled out with quantifiers, Definition 5.33 says

$$\exists R \in \mathbb{R}\, \forall v \in S : |v| < R.$$

Read it aloud: there is *one* radius R that works for *every* point of S at the same time. This is where many stumble, because swapping the two quantifiers gives the statement

$$\forall v \in S\, \exists R \in \mathbb{R} : |v| < R,$$

which is true for every subset S whatsoever: given v, choose $R = |v| + 1$. The swapped statement says nothing — choosing R after seeing v is cheating. In the definition R must be chosen first, and then survive all $v \in S$. The order of $\exists$ and $\forall$ is everything.
For the same reason, showing that a subset S is *not* bounded amounts to proving the negation

$$\forall R \in \mathbb{R}\, \exists v \in S : |v| \geq R$$

i.e., no matter which radius is proposed, some point of S escapes the ball.

(5.35) REMARK.

It does not matter whether we demand $|v| < R$ or $|v| \leq R$ above: if $|v| \leq R$ for every $v \in S$, then also $|v| < R + 1$ for every $v \in S$, and $R + 1$ is just another radius. With this in mind, boundedness of S is equivalent to each of the following two conditions.

(i) There exists R, such that

$$x_1^2 + \cdots + x_d^2 \leq R$$

for every $(x_1, \ldots, x_d) \in S$. This says $|v|^2 \leq R$, which is the same as $|v| \leq \sqrt{R}$.

(ii) There exists R, such that

$$|x_i| \leq R$$

for $i = 1, \ldots, d$ and every $(x_1, \ldots, x_d) \in S$ i.e., no single coordinate can run off to infinity. Here $|x_i| \leq |v|$ gives one direction and $|v| \leq \sqrt{d} \cdot \max(|x_1|, \ldots, |x_d|)$ the other.

Every finite subset is bounded (why?). For $d = 1$, Definition 5.33 simply says

$$\exists R \in \mathbb{R}\, \forall x \in S : |x| \leq R$$

This implies that an interval $S = [a, b]$ is bounded by putting $R = \max(|a|, |b|)$ in Definition 5.33.

LLM prompt — not displayed in the pdf version.

(5.36) EXERCISE.

Show precisely that the subset $\mathbb{N}$ of $\mathbb{R}$ is not bounded, whereas the subset $\{1, \frac{1}{2}, \frac{1}{3}, \ldots\}$ is.

> **Hint.** Use the negation from Remark 5.34: given any proposed radius R, you must point to a natural number n with $n \geq R$ (this is the archimedean property of the real numbers from the first chapter).

♠

(5.37) QUIZ.

quiz — not displayed in the pdf version.

♠

(5.38) EXERCISE.

Sketch why

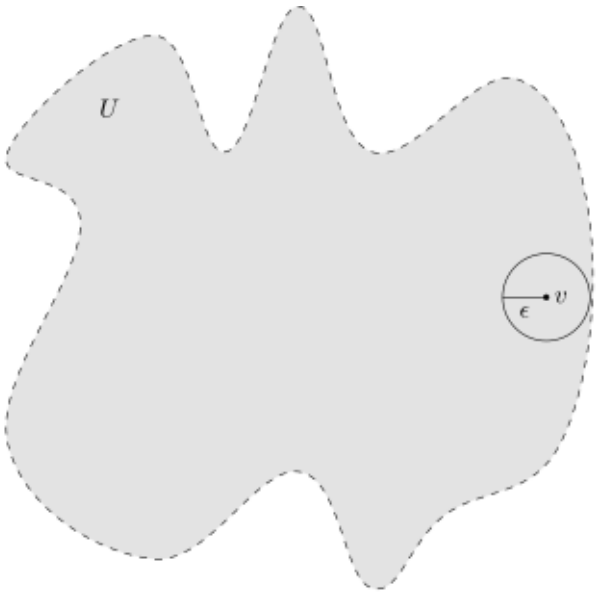
$$S = \{(x,y) \mid x \geq 0, y \geq 0, x+y \leq 1\} \subseteq \mathbb{R}^2$$

is bounded. Now use Fourier-Motzkin elimination to show the same without sketching. ♠

5.6.2 Open, closed and compact subsets and boundaries and interiors of subsets

Open subsets

An open subset of $\mathbb{R}^d$ is a subset consisting of points, that are interior in the following sense:



(5.39) DEFINITION.

A subset $U \subseteq \mathbb{R}^d$ is called open *if for every $v \in U$, there exists $\varepsilon > 0$, such that*

$$B(v,\varepsilon) \subseteq U.$$

Think of an open subset as one where every point has wiggle room: you can move a little in any direction and still stay inside. Notice the quantifiers: this time it is $\forall v \exists \varepsilon$ — the wiggle room ε is allowed to depend on the point v (points close to the edge get less of it). Compare with boundedness, where the single R had to be chosen before seeing the points.

(5.40) EXAMPLE.

The interval $(0,1)$ is an open subset of $\mathbb{R}$: for $v \in (0,1)$ the ball $B(v,\varepsilon) = (v-\varepsilon, v+\varepsilon)$ is contained in $(0,1)$ if we choose $\varepsilon = \min(v, 1-v) > 0$ — the wiggle room shrinks as v approaches 0 or 1, but it never vanishes.

The interval $[0,1]$ is *not* open: the point $v = 0$ has no wiggle room at all, since $B(0,\varepsilon) = (-\varepsilon, \varepsilon)$ contains negative numbers for every $\varepsilon > 0$. ♠

(5.41) REMARK.

The empty set $\emptyset$ and $\mathbb{R}^d$ itself are both open. For $\mathbb{R}^d$, every ε works at every point. For $\emptyset$, the condition is an all-statement over the empty set — true for the same reason that $\forall x \in \emptyset : x = 7$ was true in the quiz from the first chapter.

(5.42) EXERCISE.

Decide whether each of the subsets given below are open.

(a) $\{1\} \subseteq \mathbb{R}$

(b) $[0,1) \subseteq \mathbb{R}$

(c) $\{(x,y) \in \mathbb{R}^2 \mid y = 0\}$ (the x-axis in $\mathbb{R}^2$)

(d) $\{(x,y) \in \mathbb{R}^2 \mid y > 0\}$

♠

(5.43) EXERCISE.

Prove that an open ball given by $B(v,\varepsilon) \subseteq \mathbb{R}^d$ is an open subset.

Hint: Suppose that $u \in B(v,\varepsilon)$. Define a suitable $\varepsilon' > 0$ for $u \in B(v,\varepsilon)$ and use Corollary 5.27 to conclude that $B(u,\varepsilon') \subseteq B(v,\varepsilon)$. ♠

(5.44) EXERCISE.

Show that a finite subset of $\mathbb{R}^d$ is never open. ♠

We will need the result below.

(5.45) PROPOSITION.

If $U_1, \ldots, U_m \subseteq \mathbb{R}^d$ are open subsets, then

$$U_1 \cup \cdots \cup U_m \qquad \text{and} \qquad U_1 \cap \cdots \cap U_m$$

are open subsets.

Proof. Let $v \in U_1 \cup \cdots \cup U_m$. Then $v \in U_i$ for some i, and since U_i is open there exists $\varepsilon > 0$ with $B(v,\varepsilon) \subseteq U_i \subseteq U_1 \cup \cdots \cup U_m$.

Let $v \in U_1 \cap \cdots \cap U_m$. For each i there exists $\varepsilon_i > 0$ with $B(v,\varepsilon_i) \subseteq U_i$. Put $\varepsilon = \min(\varepsilon_1, \ldots, \varepsilon_m) > 0$. Then $B(v,\varepsilon) \subseteq B(v,\varepsilon_i) \subseteq U_i$ for every i, so $B(v,\varepsilon) \subseteq U_1 \cap \cdots \cap U_m$. □

(5.46) EXERCISE.

The proposition above concerns finitely many open subsets. Show that

$$(-1,1) \cap \left(-\tfrac{1}{2},\tfrac{1}{2}\right) \cap \left(-\tfrac{1}{3},\tfrac{1}{3}\right) \cap \cdots = \{0\}$$

and that $\{0\}$ is not an open subset of $\mathbb{R}$. Where does the proof above break down for infinitely many subsets? ♠

Closed subsets

(5.47) DEFINITION.

A subset $F \subseteq \mathbb{R}^d$ is called closed *if $\mathbb{R}^d \setminus F$ is open.*

(5.48) REMARK.

Closed is *not* the opposite of open — subsets are not doors. A subset can be both open and closed ($\emptyset$ and $\mathbb{R}^d$), and it can be neither: $[0,1)$ is not open (no wiggle room at 0) and not closed (its complement $(-\infty,0) \cup [1,\infty)$ has no wiggle room at 1).

In analogy with Proposition 5.45 we have the result below.

(5.49) PROPOSITION.

If $F_1, \ldots, F_m \subseteq \mathbb{R}^d$ are closed subsets, then

$$F_1 \cup \cdots \cup F_m \quad \text{and} \quad F_1 \cap \cdots \cap F_m$$

are closed subsets.

(5.50) EXERCISE.

Decide whether each of the subsets given below are closed.

(a) $\{1\}$

(b) $\mathbb{R}^d$

(c) $[0,1]$

(d) $[0,1)$

♠

Open intervals

(5.51) PROPOSITION.

The following subsets

$$(a,\infty) = \{x \in \mathbb{R} \mid a < x\}$$
$$(-\infty,a) = \{x \in \mathbb{R} \mid x < a\}$$
$$(a,b) = \{x \in \mathbb{R} \mid a < x < b\}$$

are open subsets of $\mathbb{R}$ *for every* $a,b \in \mathbb{R}$.

Proof. Let us prove that (a,∞) is an open subset of $\mathbb{R}$. If $x \in (a,\infty)$, then we let $\varepsilon = x - a$. Suppose that $|y - x| < \varepsilon$. If $y > x$, then $y > a$ and $y \in (a,\infty)$. If $y < x$, then $x - y < \varepsilon = x - a$ and therefore $y > a$ and $y \in (a,\infty)$. We have proved that (a,∞) is an open subset.

A similar proof shows that $(-\infty,a)$ is an open subset. If $a < b$, then

$$(a,b) = (-\infty,b) \cap (a,\infty),$$

which is an open subset by the above and Proposition 5.45. □

Closed intervals

We have a similar result for closed subsets.

(5.52) PROPOSITION.

The following subsets

$$[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$$
$$[a,\infty) = \{x \in \mathbb{R} \mid a \leq x\}$$
$$(-\infty,a] = \{x \in \mathbb{R} \mid x \leq a\}$$

are closed subsets of $\mathbb{R}$ *for every* $a,b \in \mathbb{R}$.

Proof. The proof follows from Definition 5.47 and Proposition 5.51. For example,

$$\mathbb{R} \setminus [a,\infty) = (-\infty,a).$$

□

Compact subsets

We single out the following very important class of subsets

(5.53) DEFINITION.

A subset $C \subseteq \mathbb{R}^d$ is called compact if it is bounded and closed.

Compact subsets are the heroes of optimization: a continuous function attains a minimum and a maximum on a compact subset (the extreme value theorem at the end of this chapter). Both halves of the definition are needed, as the following example shows.

(5.54) EXAMPLE.

The interval $[0,1]$ is compact: bounded and closed. The interval $(0,1)$ is bounded but not closed, and the continuous function $f(x) = x$ has no minimum on it: for every $x \in (0,1)$ there are smaller numbers in $(0,1)$, and the natural candidate 0 does not belong to the subset. The subset $\mathbb{R}$ is closed but not bounded, and $f(x) = x$ has no minimum on it either — this time it escapes to $-\infty$. ♠

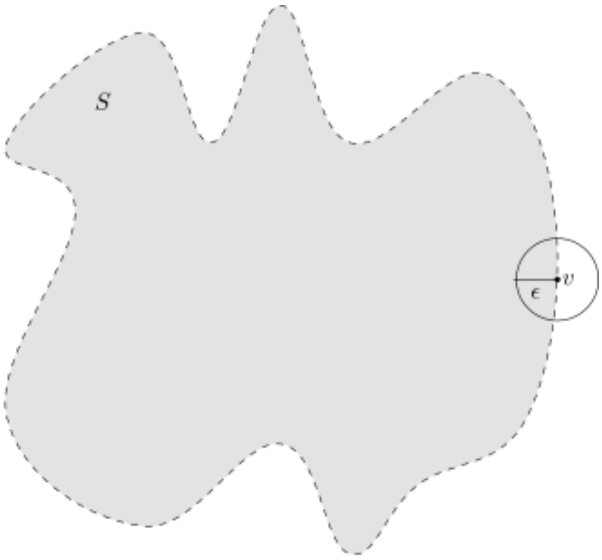
(5.55) QUIZ.

quiz — not displayed in the pdf version.

♠

The boundary of a subset

The boundary of a subset S is informally the subset of points barely touching S:



This is made precise in the following definition.

(5.56) DEFINITION.

The boundary ∂S of a subset $S \subseteq \mathbb{R}^d$ is defined as

$$\partial S = \{v \in \mathbb{R}^d \mid \forall \varepsilon > 0 : B(v,\varepsilon) \cap S \neq \emptyset \qquad \text{and} \qquad B(v,\varepsilon) \cap (\mathbb{R}^d \setminus S) \neq \emptyset\}.$$

(5.57) EXERCISE.

What is the boundary of $[0,1]$? What about $(0,1)$? ♠

The interior of a subset

The interior of a subset consists of the points, which are interior to the subset. More precisely

(5.58) DEFINITION.

The interior of a subset $S \subseteq \mathbb{R}^d$ is defined by

$$S^\circ = \{v \in S \mid \exists \varepsilon > 0 : B(v,\varepsilon) \subseteq S\}.$$

(5.59) EXERCISE.

Prove that the interior S° of any subset $S \subseteq \mathbb{R}^d$ is an open subset. Then show that S is open if and only if $S = S^\circ$. ♠

(5.60) EXERCISE.

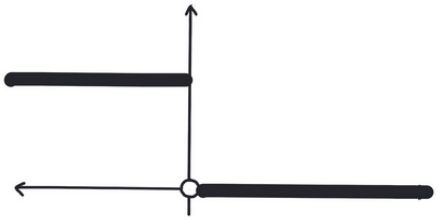
Let $S = [0,1] \subseteq \mathbb{R}$ and $S \times \{1\} = \{(x,1) \mid x \in S\} \subseteq \mathbb{R}^2$. First make a sketch of these two subsets in $\mathbb{R}$ and $\mathbb{R}^2$ respectively. Then find S° and $(S \times \{1\})^\circ$. ♠

5.7 Continuous functions

Informally, a function is continuous if small changes of the input produce small changes of the output: the graph has no sudden jumps and can be drawn without lifting the pencil. This matters for optimization. In the real world — and inside a computer — the input is almost never exact: measurements have errors and floating point numbers are rounded. A function you can trust is one where a tiny error in the input cannot cause a violent jump in the output.

The mother of all examples of a function that cannot be trusted is

$$f(x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{if } x \leq 0 \end{cases}. \tag{5.19}$$



At the point 0 something dramatic happens: $f(0) = 1$, but arbitrarily close to 0 there are inputs $x > 0$ with $f(x) = 0$. An input error of one billionth flips the output from 1 to 0. It is impossible to plot this function without lifting the pencil. Away from 0, on the other hand, the function is perfectly harmless.

How do we turn "no jumps" into precise mathematics? Through one of the most important notions in all of analysis — the *limit* of a function at a point — and a game about error tolerances.

The limit of a function at a point

We want to give precise meaning to the sentence: *the values $f(u)$ approach w as u approaches v*. Remarkably, the point v does not even have to belong to the domain of f — and this is exactly what makes the notion so useful, as the example below will show. The precise meaning is a game.

(5.61) REMARK (The ε–δ game).

Fix a function $f : S \to T$, a point v and a candidate limit w. A skeptic challenges you:

> *Skeptic*: I demand that the output stays within ε of w.
>
> *You*: Then keep the input within δ of v. I promise that suffices.

You win the round if your δ delivers: every input $u \in S$ within distance δ of v must have $f(u)$ within distance ε of w. The limit of f at v is w if you can win every round, no matter how small an ε the skeptic demands. The smaller the skeptic's ε, the smaller you will typically have to choose your δ.

The formal definition is the game in one line.

(5.62) DEFINITION.

Let $f : S \to T$ be a function, where $S \subseteq \mathbb{R}^m$ and $T \subseteq \mathbb{R}^d$, let $v \in \mathbb{R}^m$ and let $w \in \mathbb{R}^d$. We write

$$\lim_{u \to v} f(u) = w$$

and say that $f(u)$ has limit *w as u approaches v, if*

$$\forall \varepsilon > 0 \, \exists \delta > 0 \, \forall u \in S : d(u,v) < \delta \implies d(f(u),w) < \varepsilon. \tag{5.20}$$

Match the quantifiers in (5.20) with the game: $\forall \varepsilon$ — whatever the skeptic demands; $\exists \delta$ — you have an answer; $\forall u$ — and your answer works for every input within δ. Note who moves first: the skeptic picks ε before you pick δ, so your δ may (and usually does) depend on ε. Compare with the logic of boundedness in Remark 5.34: once again the order of the quantifiers is the whole story.

(5.63) EXAMPLE.

Consider

$$f(x) = \frac{x^2-1}{x-1},$$

a function defined on $S = \mathbb{R} \setminus \{1\}$: at $v = 1$ both numerator and denominator vanish and the formula breaks down. And yet f approaches a definite value as x approaches 1, because

$$f(x) = \frac{(x-1)(x+1)}{x-1} = x+1$$

for every $x \in S$. So

$$\lim_{x\to 1} f(x) = 2,$$

and the game is easy to win: if the skeptic demands ε, answer $\delta = \varepsilon$, since

$$d(f(x),2) = |x+1-2| = |x-1| = d(x,1) < \varepsilon$$

for every $x \in S$ with $d(x,1) < \delta$. Notice that the limit exists even though $f(1)$ does not: the limit only speaks about the values of f *near* 1. ♠

(5.64) EXERCISE.

What is

$$\lim_{x\to 2} \frac{x^2-4}{x-2}?$$

Does $\lim_{x\to 0} f(x)$ exist for the jump function (5.19)?

> **Hint.** For the jump function: whichever candidate limit w is proposed, the skeptic demands $\varepsilon = \frac{1}{4}$. Every δ-strip around 0 contains inputs with output 1 and inputs with output 0 — can both be within $\frac{1}{4}$ of w?

♠

Continuity

The jump function (5.19) jumps at 0; the function in Example 5.63 glides toward a definite value. The notion separating the two is *continuity*, and with limits in hand it takes one line to define: near v, the function must approach exactly the value it takes at v.

(5.65) DEFINITION.

A function $f : S \to T$, where $S \subseteq \mathbb{R}^m$ and $T \subseteq \mathbb{R}^d$, is called continuous at $v \in S$ *if*

$$\lim_{u\to v} f(u) = f(v).$$

The function f is called continuous if it is continuous at every $v \in S$. Unfolded via Definition 5.62, continuity of f at v says

$$\forall \varepsilon > 0 \exists \delta > 0 \forall u \in S : d(u,v) < \delta \implies d(f(u),f(v)) < \varepsilon \tag{5.21}$$

i.e., you can win the ε–δ game with $w = f(v)$.

LLM prompt — not displayed in the pdf version.

The definitions are short and sweet, but take some time to assimilate. You can play the continuity game (the case $w = f(v)$ of the ε–δ game) in the cell below. The green horizontal band is the skeptic's demand (ε); the blue vertical strip is your answer (δ). You win if the graph inside the strip stays inside the band; red dots are inputs that break your promise. For the jump function (5.19) at $v = 0$ you lose for every δ as soon as $\varepsilon < 1$ — try a few! Then change f to `x**2`, where the game can always be won, and play with `v`, `eps` and `delta`.

python cell — not displayed in the pdf version.

Let us return to the jump function (5.19) and see, formally, how Definition 5.65 kills any hope of it being continuous at $v = 0$. To prove this we must prove that the negation of the proposition in (5.21) is true. This reads

$$\exists \varepsilon > 0 \forall \delta > 0 \exists u \in S : d(u,0) < \delta \wedge d(f(u),1) \geq \varepsilon$$

for the function defined in (5.19). In the language of the game you now play the skeptic: demand $\varepsilon = \frac{1}{2}$. Whatever $\delta > 0$ is offered in reply, the input $u = \frac{\delta}{2}$ breaks the promise:

$$d(u,0) = \frac{\delta}{2} < \delta \qquad \text{and} \qquad d(f(u),1) = d(0,1) = 1 \geq \frac{1}{2}.$$

Almost all functions we encounter will be continuous. The function (5.19) is an anomaly — but notice that it is only discontinuous at the single point 0. At every $v \neq 0$ it is continuous (can you see how to win the game there?).

Let us stop briefly once more and see Definition 5.65 in action.

(5.66) EXAMPLE.

Let $S = T = \mathbb{R}$ in Definition 5.65. We consider the two functions

$$f(x) = x$$
$$g(x) = c,$$

where $c \in \mathbb{R}$ i.e., f is the identity function and g is a constant function given by the real number c. Both of these functions are continuous. Let us see why.

For the function f, (5.21) reads

$$\forall \varepsilon > 0 \exists \delta > 0 \forall y \in \mathbb{R} : d(y,x) < \delta \implies d(f(y),f(x)) = d(y,x) < \varepsilon.$$

This is certainly true if we pick $\delta = \varepsilon$: you win the game by echoing the skeptic's demand.

For the function g, (5.21) reads

$$\forall \varepsilon > 0 \exists \delta > 0 \forall y \in \mathbb{R} : d(y,x) < \delta \implies d(g(y),g(x)) = d(c,c) = 0 < \varepsilon.$$

Here $\delta > 0$ can be picked arbitrarily, since $0 = d(c,c) < \varepsilon$ is always true — against a constant function the skeptic never stood a chance. ♠

(5.67) QUIZ.

quiz — not displayed in the pdf version.

♠

5.7.1 An elegant way of characterizing a continuous function

Recall the definition of the preimage from Definition 1.115 and the definition of an open subset from Definition 5.39. The following characterization of continuous functions came rather late in the history of mathematics.

(5.68) PROPOSITION.

Let $f : \mathbb{R}^m \to \mathbb{R}^d$ be a function. Then f is continuous if and only if $f^{-1}(U)$ is open in $\mathbb{R}^m$ for every open subset $U \subseteq \mathbb{R}^d$.

Proof. Let $U \subseteq \mathbb{R}^d$ be an open subset. Assume first that f is continuous. We wish to prove that $f^{-1}(U)$ is open. Pick $v \in f^{-1}(U)$ and $\varepsilon > 0$ so that $B(f(v),\varepsilon) \subseteq U$. Now use the continuity of f to pick $\delta > 0$ so that (5.21) is satisfied i.e.,

$$u \in B(v,\delta) \implies f(u) \in B(f(v),\varepsilon) \tag{5.22}$$

Since $B(f(v),\varepsilon) \subseteq U$, (5.22) says that

$$B(v,\delta) \subseteq f^{-1}(U)$$

showing that $f^{-1}(U)$ is an open subset.

Now suppose that $f^{-1}(U)$ is open whenever $U \subseteq \mathbb{R}^d$ is open. For $v \in \mathbb{R}^m$ and $\varepsilon > 0$ we put $V = B(f(v),\varepsilon)$. Since V is an open subset, $f^{-1}(V)$ is open and $v \in f^{-1}(V)$. So we may find $\delta > 0$ so that $B(v,\delta) \subseteq f^{-1}(V)$. But this is exactly the statement that

$$u \in B(v,\delta) \implies f(u) \in B(f(v),\varepsilon)$$

showing that f is continuous. □

The following result is often a very useful tool in showing that a subset is closed.

(5.69) PROPOSITION.

If $F \subseteq \mathbb{R}^d$ is a closed subset and $f : \mathbb{R}^m \to \mathbb{R}^d$ a continuous function, then the preimage

$$f^{-1}(F) = \{v \in \mathbb{R}^m \mid f(v) \in F\}$$

is a closed subset of $\mathbb{R}^m$.

Proof. If $F \subseteq \mathbb{R}^d$ is closed, then $\mathbb{R}^d \setminus F$ is open. Therefore

$$f^{-1}(\mathbb{R}^d \setminus F) = f^{-1}(\mathbb{R}^d) \setminus f^{-1}(F) = \mathbb{R}^m \setminus f^{-1}(F)$$

is open by Proposition 5.68. This implies that $f^{-1}(F)$ is closed. □

(5.70) EXAMPLE.

Let us assume for now that $f : \mathbb{R}^2 \to \mathbb{R}$ given by $f(x,y) = x^2 + y^2$ is continuous (see Exercise 5.76). Then Proposition 5.69 shows that the subset

$$\begin{aligned}
&\{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \geq 1\} = \\
&\{(x,y) \in \mathbb{R}^2 \mid f(x,y) \geq 1\} = \\
&\{(x,y) \in \mathbb{R}^2 \mid f(x,y) \in [1,\infty)\} = \\
&f^{-1}([1,\infty))
\end{aligned}$$

of $\mathbb{R}^2$ is closed, since $[1,\infty)$ is a closed subset of $\mathbb{R}$ by Proposition 5.52. ♠

(5.71) EXERCISE.

Show formally that the subset

$$\{(x,y) \in \mathbb{R}^2 \mid x^2+y^2 > 1\}$$

is an open subset of $\mathbb{R}^2$. ♠

5.7.2 Working with continuous functions

We give now three important results, which can be used in concrete situations to verify that a given function is continuous. They can be proved without too much hassle. The first result below basically follows from the definition of the norm of a vector (see (5.4)).

(5.72) LEMMA.

The projection functions $\pi_i : \mathbb{R}^d \to \mathbb{R}$ defined in Definition 1.104 are continuous. In general a function $f : S \to T$ is continuous if and only if $\pi_j \circ f : S \to \mathbb{R}$ is continuous for every $j = 1,\ldots,d$, where $S \subseteq \mathbb{R}^m$ and $T \subseteq \mathbb{R}^d$.

(5.73) EXAMPLE.

Lemma 5.72 shows for example that the functions $f(x,y) = x$ and $g(x,y) = y$ are continuous functions from $\mathbb{R}^2$ to $\mathbb{R}$.

Consider the vector function $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$f(x,y) = \begin{pmatrix} x^2+y^2 \\ \sin(xy) \end{pmatrix}$$

as an example. To prove that f is continuous, Lemma 5.72 tells us that it is enough to prove that its coordinate functions $f_1, f_2 : \mathbb{R}^2 \to \mathbb{R}$

$$f_1(x,y) = x^2+y^2$$
$$f_2(x,y) = \sin(xy)$$

are continuous. ♠

Definition 5.65 also behaves nicely when continuous functions are composed. This is the content of the following

(5.74) PROPOSITION.

Suppose that $g : S \to T$ and $f : T \to R$ are continuous functions, where $S \subseteq \mathbb{R}^d, T \subseteq \mathbb{R}^e$ and $R \subseteq \mathbb{R}^f$. Then the composition

$$(f \circ g) : S \to R$$

is continuous.

To get continuous functions from functions already known to be continuous using arithmetic operations, the result below is useful.

(5.75) PROPOSITION.

Let $f, g : U \to \mathbb{R}$ be functions defined on a subset $U \subseteq \mathbb{R}^d$. If f and g are continuous, then the functions

$$\begin{aligned} (f+g) : U \to \mathbb{R} \quad & \text{given by} \quad (f+g)(x) = f(x)+g(x) \\ (fg) : U \to \mathbb{R} \quad & \text{given by} \quad (fg)(x) = f(x)g(x) \\ (f/g) : V \to \mathbb{R} \quad & \text{given by} \quad (f/g)(x) = f(x)/g(x) \end{aligned}$$

are continuous functions, where $V = \{x \in U \mid g(x) \neq 0\}$ (the last function is defined only if $g(x) \neq 0$).

Proof. This result is a consequence of the definition of continuity and Proposition 5.74. □

(5.76) EXERCISE.

Show in detail that the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x,y) = x^2 + y^2$$

is continuous by using Proposition 5.75 combined with Lemma 5.72. ♠

(5.77) REMARK.

By combining Example 5.66 with Proposition 5.75, one finds that every polynomial is a continuous function and that

$$h(x) = \frac{f(x)}{g(x)}$$

is continuous for $g(x) \neq 0$, where $f, g \in \mathbb{R}[x]$.

(5.78) EXERCISE.

Verify the claim in Remark 5.77. ♠

(5.79) REMARK.

More advanced (transcendental) functions like $\sin(x)$ and e^x also turn out to be continuous. We will return to this in the next chapter, where differentiable functions are defined.

(5.80) EXERCISE.

Show from scratch (without using Remark 5.77) that

$$g(x) = \frac{a(x)}{b(x)}$$

is a continuous function $g : V \to \mathbb{R}$, where $a(x) = x^2 - 3x + 2$ and $b(x) = x^2 - 4x + 3$ and

$$V = \mathbb{R} \setminus \{1, 3\}.$$

Use Proposition 5.52 and Proposition 5.69 to show that

$$\{x \in \mathbb{R} \mid a(x) \leq 17\}$$

is a closed subset of $\mathbb{R}$.

Hint. Write

$$\{x \in \mathbb{R} \mid a(x) \leq 17\} = a^{-1}(S),$$

where $S \subseteq \mathbb{R}$ is a suitable (closed) interval.

Does

$$\lim_{x \to 3} g(x)$$

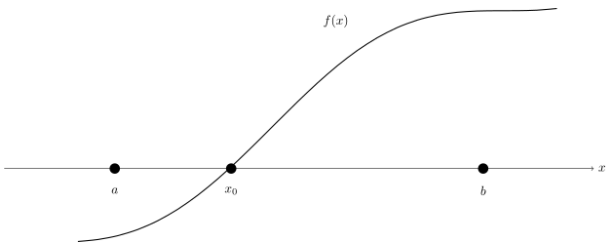
exist? What about

$$\lim_{x \to 1} g(x)?$$

♠

5.8 Important and special results for continuous functions

Below we quote a famous and very intuitive result from 1817 due to Bolzano. This result is also known as the intermediate value theorem.



(5.81) THEOREM.

Let $f : [a, b] \to \mathbb{R}$ be a continuous function, where $a < b$. If $f(a) < 0$ and $f(b) > 0$, then there exists x_0 with $a < x_0 < b$, such that $f(x_0) = 0$.

Python: Bolzano's theorem as an algorithm. The idea behind Bolzano's theorem is not just theory — it is the *bisection algorithm* for solving equations: cut the interval in half, keep the half where the sign changes, repeat. Below it hunts down a root of $f(x) = x^3 - 2$, i.e., the cube root of 2. Fifty halvings of $[0, 2]$ trap the root in an interval of length 2^{-49}.

python cell — not displayed in the pdf version.

Polynomials are continuous functions. Bolzano's result fits perfectly in the proof of the result below. This result is wrong for polynomials in $\mathbb{Q}[x]$ as witnessed by $f(x) = x^3 - 2$, which does not have a rational root.

(5.82) EXERCISE.

Use the methods of Example 1.83 to show that there is no $\xi \in \mathbb{Q}$ with $f(\xi) = 0$, where $f(x) = x^3 - 2$. ♠

(5.83) PROPOSITION.

Let

$$f(x) = a_n x^n + \cdots + a_1 x + a_0 \in \mathbb{R}[x]$$

be a polynomial of odd degree, i.e. n is odd and $a_n \neq 0$. Then f has a root, i.e. there exists $x_0 \in \mathbb{R}$, such that $f(x_0) = 0$.

Proof. We will assume that $a_n > 0$ (if not, just multiply f by -1). Consider $f(x)$ written as

$$f(x) = x^n \left(a_n + \frac{a_{n-1}}{x} + \cdots + \frac{a_1}{x^{n-1}} + \frac{a_0}{x^n} \right).$$

By choosing c negative with $|c|$ extremely big, we have $f(c) < 0$, since c^n is negative and

$$a_n + \frac{a_{n-1}}{c} + \cdots + \frac{a_1}{c^{n-1}} + \frac{a_0}{c^n} > 0$$

as a_n is positive. Notice here that the terms

$$\frac{a_{n-1}}{c} + \cdots + \frac{a_1}{c^{n-1}} + \frac{a_0}{c^n}$$

are extremely small, when $|c|$ is extremely big.

Similarly by choosing d positive and tremendously big, we have $f(d) > 0$. By Theorem 5.81, there exists x_0 with $c < x_0 < d$ with $f(x_0) = 0$. □

We end this section with a result that might be coined the mathematical cornerstone of optimization (also due to Bolzano, at least for $d = 1$). The result below is called *the extreme value theorem*.

(5.84) THEOREM.

Let C be a compact subset of $\mathbb{R}^d$ and $f : C \to \mathbb{R}$ a continuous function. Then there exists $v_{\min}, v_{\max} \in C$, such that

$$f(v_{\min}) \leq f(v) \qquad \text{and} \qquad f(v) \leq f(v_{\max})$$

for every $v \in C$.

This is a rather stunning result! You are guaranteed solutions to optimization problems of the type

$$\min_{v \in C} f(v),$$

where C is a compact subset and $f : C \to \mathbb{R}$ a continuous function. Finding the optimal solutions in this setting is another story. It can be extremely hard. For the rest of these notes we will actually dive into methods for computing optimal solutions of optimization problems such as the one above.

(5.85) EXERCISE.

Give two examples, where Theorem 5.84 fails for $d = 1$ if we relax the conditions on C. One, where C is open and another one where C is not bounded. ♠

6 CONVEX FUNCTIONS

In this chapter we will dive deeper into convex functions. The main focus will be on (differentiable) convex functions defined on intervals (convex subsets) of the real numbers i.e. (differentiable) convex functions in just one variable. Along the way, differentiability is formally introduced. I will assume that you are familiar with differentiation in an operational manner.

6.1 Strictly convex functions

Below we strengthen Definition 4.26 of a convex function.

(6.1) DEFINITION.

Let $C \subseteq \mathbb{R}^d$ be a convex subset. A strictly convex function *is a convex function $f : C \to \mathbb{R}$, such that*

$$f((1-t)u+tv) < (1-t)f(u)+tf(v) \tag{6.1}$$

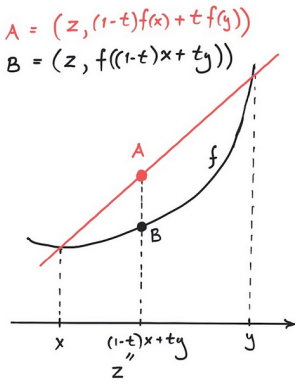
for every number t with $0 < t < 1$ and every $u, v \in C$ with $u \neq v$.

(6.2) REMARK.

The strict inequality in (6.1) collapses to an equality if $u = v, t = 0$ or $t = 1$. For example, if $u = v$, then the left hand side of (6.1) is $f((1-t)u+tu) = f(u)$ and the right hand side is $(1-t)f(u)+tf(u) = f(u)$.

(6.3) FIGURE.

Definition 6.1 is illustrated below for a function $f : \mathbb{R} \to \mathbb{R}$. Here both $u = x$ and $v = y$ are real numbers (that is, $d = 1$ in $\mathbb{R}^d$ in Definition 6.1). The (red) line segment between $(x, f(x))$ and $(y, f(y))$ lies strictly *($<$) above the (black) graph of f:*



LLM prompt — not displayed in the pdf version.

(6.4) EXAMPLE.

Consider the line (function) $f : \mathbb{R} \to \mathbb{R}$ given by

$$f(x) = ax + b$$

for $a, b \in \mathbb{R}$. This function is convex, since we can formally write for every $t \in \mathbb{R}$:

$$\begin{aligned} f((1-t)x+ty) &= a((1-t)x+ty)+b \\ &= a((1-t)x)+(1-t)b+a(ty)+tb \\ &= (1-t)(ax+b)+t(ay+b) \\ &= (1-t)f(x)+tf(y). \end{aligned} \tag{6.2}$$

However, the computation in (6.2) also shows why there is no chance that $f(x)$ is strictly convex. Intuitively, the graph of convex functions need to bend and curve a bit to be strictly convex. No lines should occur in their graphs. ♠

(6.5) EXERCISE.

Let f be a convex function. Show f is strictly convex if and only if

$$f((1-t)x+ty) = (1-t)f(x)+tf(y)$$

for $0 < t < 1$ implies that $x = y$. ♠

(6.6) EXERCISE.

Give an example of a non-constant convex function $f : \mathbb{R} \to \mathbb{R}$, which is not strictly convex. Show in details that $f(x) = x^2$ is a strictly convex function.

> **Hint.** Look back to the relevant part of Exercise 4.28 for dealing with $f(x) = x^2$.

♠

> **Python: the chord test.** Convexity is something you can *see*: the chord between two points on the graph must never dip below the graph. The cell below draws a function together with the chord between $(u, f(u))$ and $(v, f(v))$ and measures the smallest gap. Mind the logic (the quantifiers from the first chapter again!): convexity is a statement about *all* chords, so one chord above the graph proves nothing — but a single chord dipping below *disproves* convexity. Try `f = lambda x: x**4 - x**2` with `u, v = -0.5, 0.5`.
>
> *python cell — not displayed in the pdf version.*

6.2 Why are convex functions interesting?

We begin this section by giving the following result without proof.

(6.7) THEOREM.

A convex function defined on an open convex subset is continuous.

(6.8) EXERCISE.

Give an example showing that Theorem 6.7 is not true if the convex function is defined on a closed convex subset.

Hint. Try to come up with an example like $f : [0,1] \to \mathbb{R}$. Look at the end point 0.

Hint. Well, try out

$$f(x) = \begin{cases} 1 & \text{if } x = 0 \\ x & \text{if } x > 0 \end{cases}$$

♠

Let us now define precisely what is meant by a local vs a global minimum for a function.

(6.9) DEFINITION.

Let $f : S \to \mathbb{R}$ be a function, where $S \subseteq \mathbb{R}^d$ is an arbitrary subset (not necessarily convex, open or closed). Then $x_0 \in S$ is called a local minimum *for f if*

$$f(x_0) \leq f(x)$$

for every $x \in S$, which is sufficiently close to x_0. Being sufficiently close to means that $x \in S$ satisfies

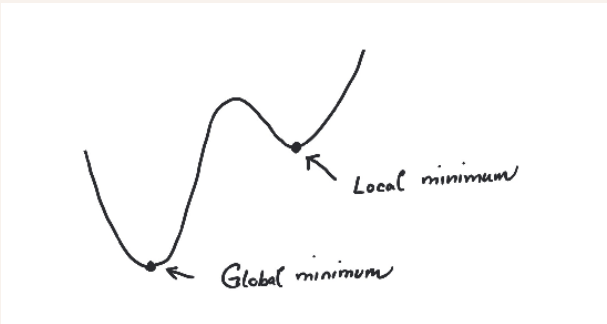
$$|x - x_0| < \varepsilon$$

for some fixed $\varepsilon > 0$.
In a much stronger notion, $x_0 \in S$ is called a global minimum *if*

$$f(x_0) \leq f(x)$$

for every $x \in S$ (not just locally).

(6.10) FIGURE.



Graph of function defined on an interval. This function has a local minimum, which is not a global minimum.

(6.11) EXERCISE.

Give an example of a local minimum that is not a global minimum for a precisely specified function. Also give an example of a global minimum, which is not uniquely defined (again for a precisely specified function). Uniquely defined means that there is precisely one x_0, such that $f(x_0)$ is minimal. ♠

We might as well have talked about maximum instead of minimum above.

(6.12) EXERCISE.

Reformulate Definition 6.9 in order to define a local and a global maximum. ♠

A *local extremum* is a point $x_0 \in S$, which is either a local minimum or a local maximum.

Convex functions $f : C \to \mathbb{R}$ are interesting, because of the local nature of the minimization problem

$$\min_{x \in C} f(x) \tag{6.3}$$

If you run into a local minimum in (6.3), then you are sure that it also is a global minimum! This is the content of the result below.

(6.13) THEOREM.

Let $f : C \to \mathbb{R}$ be a convex function defined on a convex subset $C \subseteq \mathbb{R}^d$. If $v_0 \in C$ is a local minimum, then v_0 is a global minimum. If f is strictly convex, then a global minimum for f is unique.

Proof. By the definition of local minimum in Definition 6.9, there exists $\varepsilon > 0$, such that $f(v_0) \leq f(v)$, when $v \in C$ and $|v - v_0| < \varepsilon$. Suppose that v_0 is not a global minimum. Then there exists $v_1 \in C$ with $f(v_1) < f(v_0)$. Consider the point

$$v_t = (1-t)v_0 + tv_1 \in C,$$

where $0 < t < 1$. Then

$$f(v_t) \leq (1-t)f(v_0) + tf(v_1) < (1-t)f(v_0) + tf(v_0) = f(v_0).$$

Since $|v_t - v_0| = t\,|v_1 - v_0|$, we can choose $t > 0$ sufficiently small such that $|v_t - v_0| < \varepsilon$ implying $f(v_0) \leq f(v_t)$, since v_0 is a local minimum. This contradicts that $f(v_t) < f(v_0)$ for every $0 < t < 1$. Let f be strictly convex and let v_0 be a global minimum for f. If $v_1 \in C$, $v_1 \neq v_0$ and $f(v_1) = f(v_0)$, then

$$f((1-\lambda)v_0 + \lambda v_1) < (1-\lambda)f(v_0) + \lambda f(v_1) = f(v_0)$$

for $0 < \lambda < 1$. This would contradict the global minimality of v_0, since $v_0 \neq (1-\lambda)v_0 + \lambda v_1 \in C$ for $0 < \lambda < 1$. □

(6.14) QUIZ.

quiz — not displayed in the pdf version.

♠

The following little result turns out to be very useful and also very intuitive and drawable! It is a key component in characterizing convex differentiable functions $f(x)$ in terms of $f''(x)$. We will not give the proof here.

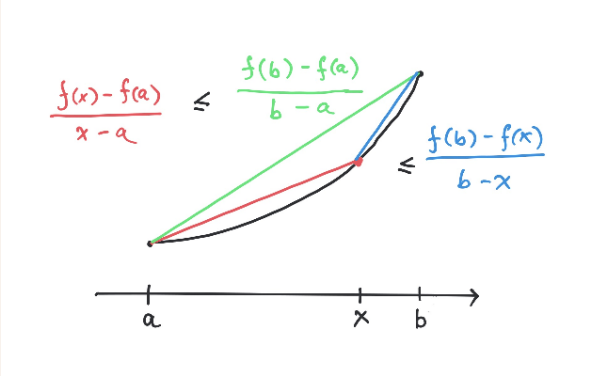
(6.15) LEMMA.

Let $f : [a,b] \to \mathbb{R}$ be a convex function. Then

$$\frac{f(x)-f(a)}{x-a} \leq \frac{f(b)-f(a)}{b-a} \leq \frac{f(b)-f(x)}{b-x}$$

for $a < x < b$.

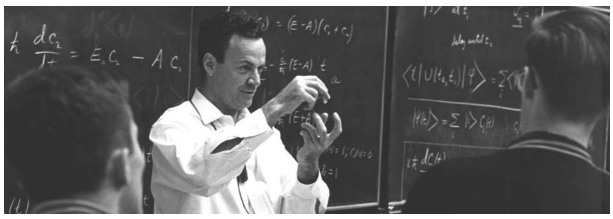
(6.16) FIGURE.



The result in Lemma 6.15 is depicted above. A formal proof can be given from first principles only using Definition 4.26.

6.3 Differentiable functions

To appreciate the depth of the notion of differentiability, you should read the story (joke, actually) in the second paragraph of section 8-2 in volume I of the famous Feynman Lectures on Physics. Below is a photograph of the master explainer in action.



6.3.1 Definition

Let $f : (a,b) \to \mathbb{R}$ be a function defined on the open interval $(a,b) \subset \mathbb{R}$. What should it mean that f is *differentiable* at a point $x_0 \in (a,b)$? The whole idea fits in one sentence.

Zoom in on the graph at the point $(x_0, f(x_0))$ and it eventually looks like a line. The slope of that line is the derivative $f'(x_0)$.

Try it yourself, right now. The cell below draws the graph of $f(x) = x^3$ together with a dashed line through the point $(1,1)$. Run it, then zoom in on the black point with the scroll wheel (or a trackpad pinch), or by dragging a small rectangle around it — keep zooming and watch the curve straighten into the dashed line. Double-click the plot to zoom all the way out again. (The first run downloads the plotting library, so it may take a little while.)

python cell — not displayed in the pdf version.

However far you zoom in, the graph of x^3 never stops looking like the dashed line through $(1,1)$ — and the slope of that line is $f'(1) = 3$. Zoom in around any *other* point and the curve straightens into a *different* line: the derivative depends on the point, which is why f' is itself a function. The rest of this subsection turns the experience you just had into a precise definition.

Notice what did *not* happen in your zooming: nothing about the graph was changed — we only looked closer.

Now try the same experiment on a function with a *corner*. The cell below draws $f(x) = |x-1|$, which has a corner at the point $(1,0)$. Zoom in on the corner, as deep as you like. It never straightens: every zoom level shows the same V shape. There is no line that this graph eventually looks like, and f is *not* differentiable at 1.

python cell — not displayed in the pdf version.

So the intuition works, and it even detects corners. Let us now turn it into a precise definition — in three small steps.

Step 1: say what a zoom is. A zoom is a *square* window centered at the point $(x_0, f(x_0))$: both axes show an interval of the same small half-width w. Every zoom you performed above was such a window, with smaller and smaller half-width w. The window must be a square because zooming has to treat both axes equally — stretching only one axis distorts the picture instead of enlarging it (more on this in the remark further below).

Step 2: blow the window up to standard size. Tiny windows are hard to compare, so we blow each one up, like enlarging a photograph. Blowing up divides every offset from the center by w: a point on the graph with horizontal offset tw (where $t \in [-1,1]$ is the rescaled coordinate) and vertical offset $f(x_0+tw) - f(x_0)$ lands at coordinates t and

$$g_w(t) = \frac{f(x_0+tw) - f(x_0)}{w}.$$

In other words: the blown-up window shows the graph of the function g_w. The cell below draws these blown-up windows for three zoom levels, all in *one* frame, so that they can be compared directly. Watch the curves settle onto a line as w shrinks. And try $f(x) = |x|$ at $x_0 = 0$: the blow-ups are all identical — a corner looks like the same corner at every zoom level and never becomes a line.

python cell — not displayed in the pdf version.

Step 3: say what must happen. "The graph becomes a line" now has an exact meaning: there is a slope c, such that the curves g_w settle onto the line ct as $w \to 0$. How far are we from the line? Writing $k = tw$ for the horizontal offset, a one-line computation gives, for $t \neq 0$,

$$g_w(t) - ct = t\left(\frac{f(x_0+k) - f(x_0)}{k} - c\right).$$

Since $|t| \leq 1$, everything hinges on the parenthesis. Give it a name:

$$\varepsilon(k) = \frac{f(x_0+k) - f(x_0)}{k} - c.$$

The curves settle onto the line exactly when $\varepsilon(k) \to 0$ as $k \to 0$ — that is, when ε, extended by $\varepsilon(0) = 0$, is continuous at 0 in the sense of the previous chapter. Finally, multiply the definition of ε by k and rearrange:

$$f(x_0+k) = f(x_0) + ck + \varepsilon(k)\,k.$$

This single equation is the definition we were after — reached from the zoom picture, not decreed from above. The ε-function in it is nothing mysterious: it is the *zoom error*, recording how far the graph deviates from the line, relative to the zoom level.

(6.17) DEFINITION.

The function $f:(a,b)\to\mathbb{R}$ is differentiable at $x_0\in(a,b)$ if there exists

(i) $c\in\mathbb{R}$

(ii) $\delta>0$ *with* $x_0-\delta, x_0+\delta\in(a,b)$ *i.e.,* $a+\delta<x_0$ *and* $x_0<b-\delta$.

(iii) A function $\varepsilon:(-\delta,\delta)\to\mathbb{R}$ *continuous at* 0 *with* $\varepsilon(0)=0$,

such that

$$f(x_0+h)-f(x_0)=ch+\varepsilon(h)h \tag{6.4}$$

for every $h\in(-\delta,\delta)$.
The number c is denoted $f'(x_0)$ and called the derivative *of f at x_0; f is called* differentiable *if it is differentiable at every $x_0\in(a,b)$.*

LLM prompt — not displayed in the pdf version.

Definition 6.17 is the zoom picture written out. It also contains the classical description of the derivative: dividing (6.4) by h gives

$$\frac{f(x_0+h)-f(x_0)}{h}=c+\varepsilon(h),$$

and the left hand side is the slope of the line through $(x_0,f(x_0))$ and the nearby point $(x_0+h,f(x_0+h))$ on the graph. So $\varepsilon(h)\to 0$ says exactly that these slopes tend to c i.e.,

$$\lim_{h\to 0}\frac{f(x_0+h)-f(x_0)}{h}=f'(x_0).$$

This is how the derivative is often introduced — Newton would say that $f'(x_0)$ is the slope just before x becomes x_0 — and you may use the limit description and the zoom picture interchangeably.

(6.18) REMARK.

Two fine points about the zoom intuition. First, the window must be a *square*: plotting software often stretches the two axes independently, and with enough one-sided stretching almost any graph can be made to look flat. Honest zooming treats both axes equally. Second, "looks like a line" must mean a line with a slope $c\in\mathbb{R}$: zooming in on $f(x)=\sqrt[3]{x}$ at $x_0=0$ produces something that looks more and more like a *vertical* line — no real number describes that slope, and the function is not differentiable at 0, even though the zoomed picture straightens. Try `f = lambda x: np.cbrt(x)` with `x0 = 0` in the cells above and watch the graph escape through the top and bottom of the window.

(6.19) REMARK.

If a function $f:(a,b)\to\mathbb{R}$ is differentiable, we get a new function $f':(a,b)\to\mathbb{R}$ giving the (first) derivative at a point as output. We may ask again if this function is differentiable. If this is so, we may define a function $f'':(a,b)\to\mathbb{R}$ given by $f''(x)=(f')'(x)$ called the second derivative. This procedure may be continued. We use the notation $f^{(n)}$ for the n-th derivative.

(6.20) EXAMPLE.

Let us apply Definition 6.17 to the function $f(x) = x^2$ at the point x_0. Here

$$f(x_0+h) - f(x_0) = (x_0+h)^2 - x_0^2 = 2x_0h + h^2.$$

Here you immediately see that $c = f'(x_0) = 2x_0$ with $\varepsilon(h) = h$ (and $\delta = \infty$) in Definition 6.17. ♠

(6.21) EXERCISE.

Use Definition 6.17 to formally show that $f'(x) = 3x^2$ if $f(x) = x^3$. ♠

A differentiable function is continuous as is shown in the following result.

(6.22) PROPOSITION.

If the function $f : (a,b) \to \mathbb{R}$ is differentiable at $x_0 \in (a,b)$, then it is continuous at x_0.

Proof. That f is continuous at x_0 means (recall Definition 5.65) that to every $\varepsilon > 0$, we may find $\delta > 0$ so that

$$|x - x_0| < \delta \implies |f(x) - f(x_0)| < \varepsilon. \tag{6.5}$$

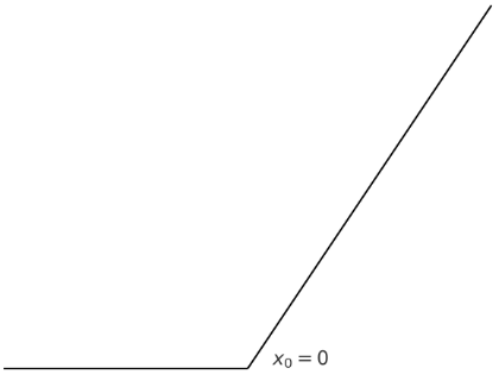
We are assuming that f is differentiable at x_0, so according to Definition 6.17, there exists a number c so that (with $h = x - x_0$)

$$|f(x) - f(x_0)| = |(c + \varepsilon(x - x_0))(x - x_0)|.$$

I will not write every detail out here, but you can see from the formula above that $|f(x) - f(x_0)| < M|x - x_0|$ for some number M, when $|x - x_0|$ is sufficiently small. This gives a δ that can be used in (6.5). □

(6.23) EXAMPLE.

The ReLU function $f(x) = \max(0,x)$ is an example of a function, which is continuous, but not differentiable at $x_0 = 0$. This is much related to its sharp corner there.



As mentioned in these notes, the ReLU function plays a prominent role as an activation function in neural networks.

♠

(6.24) EXERCISE.

Show precisely that the ReLU function is not differentiable at 0. ♠

6.3.2 Formulas

In operating with differentiable functions you are supposed to draw on your previous knowledge. I have summarized some of this knowledge below (even though we will give hints below as how to prove some of the rules).

1. If $f(x) = ag(x)$, where $a \in \mathbb{R}$, then
$$f'(x) = ag'(x).$$
2. If $f(x) = x^n$, where $n \in \mathbb{N}$, then
$$f'(x) = nx^{n-1}.$$
3. If $f(x) = e^x$, then
$$f'(x) = f(x) = e^x.$$
4. If $f(x) = \log(x)$, then
$$f'(x) = 1/x.$$
Here $\log(x)$ denotes the logarithm with base e.
5. If $f(x) = \sin(x)$, then
$$f'(x) = \cos(x).$$
6. If $f(x) = \cos(x)$, then
$$f'(x) = -\sin(x).$$
7. If $f(x)$ and $g(x)$ are differentiable functions, then the derivative of their product is
$$(fg)'(x) = f'(x)g(x) + f(x)g'(x).$$
8. If $f(x)$ and $g(x)$ are differentiable functions, then the derivative of their quotient is
$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}.$$
9. If $f(x)$ and $g(x)$ are composable differentiable functions, then the derivative of their composite is
$$(f \circ g)'(x) = f'(g(x))g'(x).$$

(6.25) EXERCISE.

Suppose that $f(x) = \sin(x)$. What is
$$f^{(17)}(x)?$$

♠

6.3.3 The derivative of a product

From high school you know that the derivative of a product of two functions f and g is given by the formula

$$(fg)'(x) = f'(x)g(x) + f(x)g'(x). \tag{6.6}$$

We can use the ε-definition (6.4) to derive the product rule in (6.6). The computation below is a bit cumbersome, but actually quite doable. We assume to begin with that f and g are differentiable at x_0 according to (6.4) i.e.,

$$\begin{aligned} f(x_0+h) &= f(x_0) + f'(x_0)h + \varepsilon_f(h)h \\ g(x_0+h) &= g(x_0) + g'(x_0)h + \varepsilon_g(h)h. \end{aligned}$$

Then we start the computation:

$$\begin{aligned} &(fg)(x_0+h) = f(x_0+h)g(x_0+h) = \\ &(f(x_0) + f'(x_0)h + \varepsilon_f(h)h)\ (g(x_0) + g'(x_0)h + \varepsilon_g(h)h) = \\ &f(x_0)g(x_0) + (f'(x_0)g(x_0) + f(x_0)g'(x_0))h + \varepsilon(h)h, \end{aligned} \tag{6.7}$$

where the function

$$\varepsilon(h) = f(x_0)\varepsilon_g(h) + f'(x_0)g'(x_0)h + f'(x_0)\varepsilon_g(h)h + \varepsilon_f(h)g(x_0) + \varepsilon_f(h)g'(x_0)h + \varepsilon_f(h)\varepsilon_g(h)h \tag{6.8}$$

is seen to be continuous at $h = 0$ with $\varepsilon(0) = 0$. The end result of this computation shows that fg is differentiable at x_0 with

$$(fg)'(x_0) = f'(x_0)g(x_0) + f(x_0)g'(x_0) \tag{6.9}$$

again according to (6.4).

(6.26) EXERCISE.

Show that the ε function defined in (6.8) satisfies the relevant conditions in Definition 6.17. ♠

The formula for the derivative of a fraction i.e.,

$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$$

can be derived using a neat little trick. This is the topic of the following exercise.

(6.27) EXERCISE.

Show how the product rule may be used to derive the rule for finding the derivative of a fraction:

$$\left(\frac{f}{g}\right)'(x_0) = \frac{f'(x_0)g(x_0) - f(x_0)g'(x_0)}{g(x_0)^2}.$$

Hint.

$$f'(x) = \left(g(x)\left(\frac{f(x)}{g(x)}\right)\right)'.$$

♠

6.3.4 The one variable chain rule

The formula for the derivative of a composite function is given by

$$(f\circ g)'(x_0) = f'(g(x_0))g'(x_0),$$

where $g(x_0)$ is in the domain of f. Let us see how (6.4) applies in showing this.

Suppose that f is differentiable at $g(x_0)$ and g is differentiable at x_0, then we can mess around a bit with the ε-functions for f and g for the composite function $f(g(x))$ around x_0:

$$\begin{aligned} f(g(x_0+h)) &= f(g(x_0)+g'(x_0)h+h\varepsilon_g(h)) \\ &= f(g(x_0))+f'(g(x_0))g'(x_0)h+\varepsilon(h)h, \end{aligned}$$

where (take a deep breath)

$$\varepsilon(h) = f'(g(x_0))\varepsilon_g(h)+\varepsilon_f\big(g'(x_0)h+\varepsilon_g(h)h\big)\big(g'(x_0)+\varepsilon_g(h)\big).$$

Here ε is seen to be continuous at 0 with $\varepsilon(0)=0$ i.e., the composition $f(g(x))$ is differentiable at x_0 with derivative

$$(f\circ g)'(x_0) = f'(g(x_0))g'(x_0). \tag{6.10}$$

The formula (6.10) is extremely important and useful. We give some applications in the exercises below.

(6.28) EXERCISE.

For the function $f(x)=x^n$ for $n\in\mathbb{N}$, you already know that $f'(x)=nx^{n-1}$. Show that if you define the function $g:\{x\in\mathbb{R}\mid x>0\}\to\mathbb{R}$ by

$$g(x) = x^a := e^{\log(x)a},$$

for an arbitrary number $a\in\mathbb{R}$, then $g'(x)=ax^{a-1}$. ♠

(6.29) EXERCISE.

Compute the derivative of the function $f:(0,\pi)\to\mathbb{R}$ given by

$$f(x) = \frac{1}{\sqrt{\sin(x)}}$$

using only paper and pencil! You can check your result afterwards using a computer. ♠

(6.30) EXERCISE.

Suppose that g and f are inverse functions i.e.,

$$f(g(x)) = x \qquad \text{and} \qquad g(f(x)) = x.$$

If you know the derivative of f, how can you use the chain rule to get the derivative of g? Illustrate with examples like $f(x)=x^2$ and $g(x)=\sqrt{x}$, $f(x)=e^x$ and $g(x)=\log(x)$. ♠

(6.31) EXERCISE.

Suppose that $f:\mathbb{R}\to\mathbb{R}$ is a convex function. We know that f is continuous, but is f differentiable at every point $x_0\in\mathbb{R}$?

> **Hint.** Nope. This is wrong. Come up with a convex function f and a point x_0, such that f is not differentiable at x_0.

♠

6.3.5 The Newton-Raphson method for finding roots

We begin this section with a surprising example.

(6.32) EXAMPLE.

Suppose that $a > 0$ and we wish to compute $\sqrt{a}$. To do this we may focus on the quadratic equation $f(x) = x^2 - a = 0$ and attempt to compute an approximate value $x_0 \geq 0$, such that $f(x_0)$ is close to 0. Let me at this point disclose that there is a very effective iterative scheme for doing this. You start by putting $x_0 = a$ and then iterate using the formula

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{a}{x_i}\right) \tag{6.11}$$

to get better and better approximations $x_0, x_1, x_2, \ldots$ to $\sqrt{a}$.

The formula in (6.11) is derived from

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

where $f(x) = x^2 - a$.

You can try out (6.11) below.

python cell — not displayed in the pdf version.

♠

I have been in complete awe of the Newton-Raphson method since my early youth. It is an algorithm, where the notion of differentiability really shines.

The method comes from Definition 6.17 with $h = x - x_0$: we are assuming that x_0 is very close to x, where $f(x) = 0$. Then

$$f(x) - f(x_0) = f'(x_0)(x - x_0) + \text{a very small number.}$$

Ignoring the very small number and solving this equation for x we get

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

In the python cell below, I have entered the algorithm starting in $x_0 = 0$ running ten iterations for finding a zero for $f(x) = \cos(x) - x$.

Graph.

python cell — not displayed in the pdf version.

python cell — not displayed in the pdf version.

(6.33) EXERCISE.

Give an example, where the Newton-Raphson method cycles between points and never finds the desired zero. Perhaps a drawing will help here. ♠

The Newton-Raphson converges rapidly in most cases. Of course, it breaks down violently if it runs into a *critical point* i.e., a point x, such that $f'(x) = 0$.

Below is some python code for experimenting with Newton's method.

python cell — not displayed in the pdf version.

(6.34) EXAMPLE.

The formula (see button in Example 1.87) for the (monthly) payment Y on a (car) loan over N payments with a down payment of P and an interest rate of r (per payment or term) is given by the formula

$$Y = \frac{rP}{1-\left(\frac{1}{1+r}\right)^N}.$$

There is no explicit formula for calculating r given Y, P and N. Here the Newton-Raphson method is invaluable for estimating r by approximating a zero for the function

$$r(x) = Y - \frac{xP}{1-\left(\frac{1}{1+x}\right)^N}.$$

♠

(6.35) EXERCISE.

Your bank promises you a loan of 1.000.000 DKK with yearly payments of 45.000 DKK over 30 years. At the same time it claims that its interest rate is very favorable at only 1.0%. Here the bank is wrong! What is the real interest rate? How much money do you save (compared to the original offer from the bank) if you insist that the bank offers you the promised interest rate of 1.0%?

Python: check your answer.

python cell — not displayed in the pdf version.

♠

6.3.6 Critical points and extrema

(6.36) DEFINITION.

A critical point for a differentiable function $f : (a,b) \to \mathbb{R}$ is a point $x_0 \in (a,b)$ with

$$f'(x_0) = 0.$$

The crucial result here is the following. It seems to date back to Fermat (see Fermat's theorem).

(6.37) LEMMA.

Let $f : (a,b) \to \mathbb{R}$ be a differentiable function. If x_0 is a local extremum for f, then x_0 is a critical point i.e., $f'(x_0) = 0$.

Proof. Suppose that ξ is a local maximum and that

$$f(\xi+h)-f(\xi)=f'(\xi)h+\varepsilon(h)h$$

according to (6.4). If $f'(\xi)>0$, then we can choose $\delta>0$ sufficiently small, such that $|\varepsilon(h)|<f'(\xi)$ if $0\leq h<\delta$, since $\varepsilon(0)=0$ and ε is continuous in 0. Therefore

$$f(\xi+h)-f(\xi)=(f'(\xi)+\varepsilon(h))h>0,$$

contradicting that ξ is a local maximum. The proof is similar for $f'(\xi)<0$ and if ξ is a local minimum. □

(6.38) EXERCISE.

Is the converse of the above lemma true i.e., if $f'(x_0)=0$ is x_0 a local extremum? ♠

(6.39) QUIZ.

quiz — not displayed in the pdf version.

♠

Theorem 6.40 below is called the mean value theorem. It is a consequence of Lemma 6.37 and the extremely important Theorem 5.84 about continuous functions on compact subsets attaining their maxima and minima!

(6.40) THEOREM.

Let $f:[a,b]\to\mathbb{R}$ be continuous and differentiable on (a,b). Then there exists $x_0\in(a,b)$ such that

$$f'(x_0)=\frac{f(b)-f(a)}{b-a}.$$

6.3.7 Increasing functions

The definition below is much simpler than the definition of differentiability.

(6.41) DEFINITION.

A function $f:S\to\mathbb{R}$ with $S\subseteq\mathbb{R}$ is called increasing *if*

$$x\leq y\Rightarrow f(x)\leq f(y)$$

and strictly increasing *if*

$$x<y\Rightarrow f(x)<f(y)$$

for $x,y\in S$.

LLM prompt — not displayed in the pdf version.

(6.42) EXERCISE.

Give an example of an increasing function. Give an example of an increasing function that is not strictly increasing. ♠

The following very important result is a consequence of Theorem 6.40. You probably already know this result from your previous (danish) education (monotoniforhold!).

(6.43) PROPOSITION.

Let $f : (a,b) \to \mathbb{R}$ be a differentiable function. Then f is increasing if and only if $f'(x) \geq 0$ for every $x \in (a,b)$. If $f'(x) > 0$ for every $x \in (a,b)$, then f is strictly increasing.

(6.44) QUIZ.

quiz — not displayed in the pdf version.

♠

(6.45) EXERCISE.

Show that $f(x) = x^3$ is strictly increasing i.e.,

$$x < y \implies x^3 < y^3.$$

Hint.

$$y^3 - x^3 = (y-x)(y^2 + xy + x^2),$$

but why is $y^2 + xy + x^2$ always > 0 except when $x = y = 0$?

♠

(6.46) EXERCISE.

Suppose that $f : [a,b] \to \mathbb{R}$ is a continuous function, such that f is differentiable on the open interval (a,b). Is f increasing on $[a,b]$ if $f'(x) \geq 0$ for every $x \in (a,b)$? ♠

(6.47) EXERCISE.

Is it possible for a strictly increasing function $f : \mathbb{R} \to \mathbb{R}$ to be bounded i.e., does there exist a (positive) number M, such that $|f(x)| \leq M$ for every $x \in \mathbb{R}$?

Hint. Have a look at

$$f(x) = \frac{1}{1+e^{-x}}.$$

♠

6.4 Taylor polynomials

If x_0 is a critical point for f we cannot conclude that x_0 is a local extremum. We know that $f'(x_0) = 0$ and we can get more information out of f by exploring the signs of

$$f''(x_0), f'''(x_0), \dots$$

Suppose that

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

is a polynomial, then

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n. \tag{6.12}$$

For nice functions like $f(x) = e^x$ we can play this game ad infinitum. In fact in this way we get the beautiful infinite series

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots + \frac{x^n}{n!} + \cdots.$$

If f is an n times differentiable function defined at 0, we call the polynomial in (6.12) the Taylor polynomial about the point 0 of degree n associated with f. Similarly, one may also define the Taylor polynomial of order n about a point a by

$$f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n.$$

Taylor polynomials can be used to approximate more complicated functions such as $\cos(x)$ and $\sin(x)$ with a well defined error term. This is cool classical mathematics. Unfortunately we do not have time to go deeper into Taylor's theorem, which states this in precise terms.

You can watch the approximation happen in the cell below: the Taylor polynomials of $\sin(x)$ hug the graph on a wider and wider interval as the degree grows. Change f, the degrees, or the plotting window — and use the cell to check your answer to the exercise below.

python cell — not displayed in the pdf version.

(6.48) EXERCISE.

Compute the Taylor polynomial for $f(x) = \cos(x)$ up to degree 10. ♠

(6.49) EXERCISE.

Suppose you have a number i that satisfies

$$i^2 = -1.$$

Can you make sense of the formula

$$e^{ix} = \cos(x) + i\sin(x)$$

using Taylor polynomials? ♠

In the context of optimization, the following result becomes important. We will not give the proof, but only notice that Theorem 6.40 also here plays an important role.

(6.50) THEOREM.

Let x_0 be a critical point of an $n+1$ times differentiable function $f:(a,b)\to\mathbb{R}$, such that $f^{(n+1)}$ is a continuous function,

$$\begin{aligned} f''(x_0) &= 0\\ f'''(x_0) &= 0\\ &\vdots\\ f^{(n-1)}(x_0) &= 0 \end{aligned}$$

and $f^{(n)}(x_0)\neq 0$. If n is even, then x_0 is a local minimum if $f^{(n)}(x_0)>0$ and a local maximum if $f^{(n)}(x_0)<0$. If n is odd, then x_0 is not a local extremum.

(6.51) EXAMPLE.

Let us apply Theorem 6.50 to the function

$$f(x)=ax^2+bx+c,$$

where $a\neq 0$. Here $f'(x)=2ax+b$ and

$$x_0=-\frac{b}{2a}$$

is a critical point (why?). Since

$$f''(x_0)=2a,$$

we see that x_0 is a local minimum if $a>0$ and a local maximum if $a<0$. ♠

(6.52) EXERCISE.

Have you seen Example 6.51 elsewhere, perhaps in a more geometric setting? What type of curve is the graph of $f(x)$? Here you may consult your previous mathematical knowledge.

What is the outcome, when you apply Theorem 6.50 to the function $f(x)=x^3$ at $x_0=0$? ♠

(6.53) EXERCISE.

Show that $x_0=0$ is a critical point of the function $f:(-\frac{1}{2},\infty)\to\mathbb{R}$ defined by

$$f(x)=e^x+\log(1+2x)-3x.$$

Use Theorem 6.50 in deciding if it is a local maximum or minimum or neither.

♠

6.5 Differentiable convex functions

The following theorem is proved using Lemma 6.15 and Theorem 6.40. It immediately implies Corollary 6.55, which is the result mostly used.

(6.54) THEOREM.

Let $f : (a,b) \to \mathbb{R}$ be a differentiable function. Then f is convex if and only if f' is increasing. If f' is strictly increasing, then f is strictly convex.

Theorem 6.54 leads to the following all important result.

(6.55) COROLLARY.

Let $f : (a,b) \to \mathbb{R}$ be a twice differentiable function. Then f is convex if and only if $f''(x) \geq 0$ for every $x \in (a,b)$. If $f''(x) > 0$ for every $x \in (a,b)$, then f is strictly convex.

(6.56) REMARK.

Wait! Stop! Why did I not write $f''(x) > 0$ if and only if f is strictly convex?

(6.57) QUIZ.

quiz — not displayed in the pdf version.

♠

(6.58) EXERCISE.

You cannot deduce from Corollary 6.55 that the function $g : \mathbb{R} \to \mathbb{R}$ given by $g(x) = x^4$ is a strictly convex function. Why not?

You can deduce from Corollary 6.55 that $f(x) = x^2$ is a strictly convex function. How can $g(x) = f(x)^2$ be used to prove that $g(x)$ is a strictly convex function? ♠

(6.59) EXERCISE.

Show that $f(x) = e^x$ is a strictly convex function $f : \mathbb{R} \to \mathbb{R}$.

Show that $f(x) = -\log(x)$ is a strictly convex function $f : (0,\infty) \to \mathbb{R}$. ♠

(6.60) EXERCISE.

Show that $f : \{x \in \mathbb{R} \mid x \geq 0\} \to \mathbb{R}$ given by

$$f(x) = -\sqrt{x}$$

is a strictly convex function. ♠

Another nice application of Lemma 6.15 (and Theorem 6.54) is the following.

(6.61) THEOREM.

Let $f : (a,b) \to \mathbb{R}$ be a differentiable function. Then f is convex if and only if

$$f(y) \geq f(x) + f'(x)(y-x)$$

for every $x, y \in (a,b)$.

Python: the graph sits above its tangents. Theorem 6.61 says that a differentiable function is convex exactly when its graph lies above every one of its tangent lines. The cell below draws a function with four of its tangents. For a convex function no tangent ever crosses the graph. Try `f = lambda x: x**3` and watch a tangent cut through.

python cell — not displayed in the pdf version.

(6.62) EXERCISE.

Suppose that $f : (a,b) \to \mathbb{R}$ is a differentiable convex function and $x_0 \in (a,b)$ is a critical point for f. What can you say about x_0 using Theorem 6.61? ♠

7 SEVERAL VARIABLES

A function of several variables usually refers to a function

$$f : \mathbb{R}^d \to \mathbb{R}, \tag{7.1}$$

where $d > 1$ is a natural number. We have already seen functions of several variables with $d > 1$. In particular, in Chapter 4, we saw linear functions (in connection with linear programming) like

$$f(x_1, x_2) = 3x_1 + 2x_2. \tag{7.2}$$

This is a rather simple function of several variables with $d = 2$ in (7.1). In general functions as in (7.1) can be wildly complicated. One of the main purposes of this chapter is to zero in on the class of differentiable functions in (7.1). In Chapter 6 we defined what it means for a function of one variable to be differentiable. This was inspired by zooming in at a point on the graph of the function. In several variables (for $n > 1$) one has to be a bit clever in the definition of differentiability. The upshot is that the derivative at a point now is a row vector (or more generally a matrix) instead of being a single number. As an example, using notation that we introduce in this chapter, the derivative of the function in (7.2) at $(0,0)$ is

$$\begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 3 & 2 \end{pmatrix}.$$

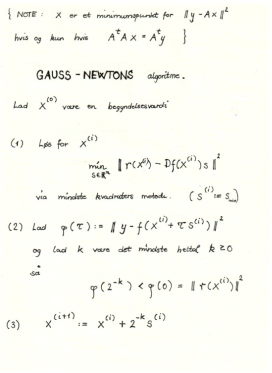
This notation means that partial differentiation with respect to a variable occurs i.e., one fixes the variable and computes the derivative with respect to this variable viewing all the other variables as constants.

First some treasured memories from the author's past.

7.1 Introduction

Many years ago (1986-89), I had a job as a financial analyst in a bank (now a hotel!) working (often late at night) with a spectacular view of Copenhagen from the office circled below.

This was long before a financial analyst became a quant and machine learning became a buzz word. Digging through my old notes from that time, I found the outlines below.


NOTE: x er et minimumspunkt for $\|y - Ax\|^2$ hvis og kun hvis $A^tAx = A^ty$

GAUSS - NEWTONS algoritme.

Lad $x^{(0)}$ være en begyndelsesværdi

(1) Løs for $x^{(i)}$

$$\min_{s \in \mathbb{R}^n} \| r(x^{(i)}) - Df(x^{(i)})s \|^2$$

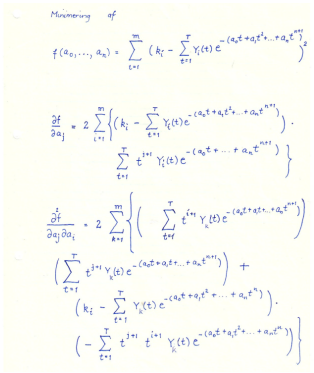
via mindste kvadraters metode. ($s^{(i)} := s_{min}$)

(2) Lad $\varphi(\tau) := \|y - f(x^{(i)} + \tau s^{(i)})\|^2$

og lad k være det mindste heltal $k \geq 0$ så

$$\varphi(2^{-k}) < \varphi(0) = \|r(x^{(i)})\|^2$$

(3) $x^{(i+1)} := x^{(i)} + 2^{-k} s^{(i)}$


Minimering af

$$f(a_0, \ldots, a_n) = \sum_{i=1}^{m} \left(k_i - \sum_{t=1}^{T} Y_i(t) e^{-(a_0 t + a_1 t^2 + \ldots + a_n t^{n+1})} \right)^2$$

$$\frac{\partial f}{\partial a_j} = 2 \sum_{i=1}^{m} \left\{ \left(k_i - \sum_{t=1}^{T} Y_i(t) e^{-(a_0 t + a_1 t^2 + \ldots + a_n t^{n+1})} \right) \cdot \sum_{t=1}^{T} t^{j+1} Y_i(t) e^{-(a_0 t + \ldots + a_n t^{n+1})} \right\}$$

$$\frac{\partial^2 f}{\partial a_j \partial a_k} = 2 \sum_{i=1}^{m} \left\{ \left(\sum_{t=1}^{T} t^{j+1} Y_i(t) e^{-(a_0 t + a_1 t^2 + \ldots + a_n t^{n+1})} \right) \left(\sum_{t=1}^{T} t^{k+1} Y_i(t) e^{-(a_0 t + a_1 t^2 + \ldots + a_n t^{n+1})} \right) + \left(k_i - \sum_{t=1}^{T} Y_i(t) e^{-(a_0 t + a_1 t^2 + \ldots + a_n t^n)} \right) \cdot \left(- \sum_{t=1}^{T} t^{j+1} t^{k+1} Y_i(t) e^{-(a_0 t + a_1 t^2 + \ldots + a_n t^n)} \right) \right\}$$

These were notes I made in connection with modelling the yield curve for zero coupon bonds. I had to fit a very non-linear function in *several variables* to financial data and had to use effective numerical tools (and programming them[1] in APL). Tools that are also used today in machine learning and data science.

Ultimately we are interested in solving optimization problems like

$$\min_{(x_1,\ldots,x_d) \in C} \quad f(x_1, \ldots, x_d),$$

where $C \subseteq \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}$ is a differentiable (read nice for now) function.

Training neural networks is a fancy name for solving an optimization problem, where usually $C = \mathbb{R}^d$ and f is built just like in the least squares method from some data points. The difference is that in neural networks, f is an incredibly complicated (differentiable) function composed of several intermediate functions. We do not, as in the method of least squares, have an explicit formula for finding a minimum. We have to rely on iterative methods. One such method is called *gradient descent*. By the end of this chapter, in Section 7.9, you will use it to train genuine neural networks — from a single neuron to a network reading handwritten digits and, at the very end, one that writes — live in these notes.

Recall the definition of a function $f : \mathbb{R} \to \mathbb{R}$ being differentiable at a point $x_0 \in \mathbb{R}$ with derivative $c = f'(x_0)$. Here we measured the change $f(x_0 + h) - f(x_0)$ of f in terms of the change h (in x). It had to have the form

$$f(x_0 + h) - f(x_0) = ch + \varepsilon(h)h, \tag{7.3}$$

where $\varepsilon : (-\delta, \delta) \to \mathbb{R}$ is a function continuous in 0 with $\varepsilon(0) = 0$ and $\delta > 0$ small. If you divide both sides of (7.3) by h you recover the usual more geometric definition of differentiability as a limiting slope:

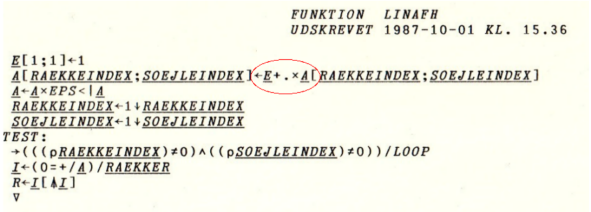
$$\lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} = c = f'(x_0). \tag{7.4}$$

We wish to define differentiability at $v_0 \in \mathbb{R}^d$ for a function $f : \mathbb{R}^d \to \mathbb{R}^m$. In this setting the quotient

$$\frac{f(v_0 + h) - f(v_0)}{h}$$

1

Programmering i APL


```
                                FUNKTION  LINAFH
                                UDSKREVET 1987-10-01 KL. 15.36
E[1;1]←1
A[RAEKKEINDEX;SOEJLEINDEX]←E+.×A[RAEKKEINDEX;SOEJLEINDEX]
A←A×EPS<|A
RAEKKEINDEX←1↓RAEKKEINDEX
SOEJLEINDEX←1↓SOEJLEINDEX
TEST:
→(((⍴RAEKKEINDEX)≠0)∧((⍴SOEJLEINDEX)≠0))/LOOP
I←(0=+/A)/RAEKKER
R←I[⍋I]
∇
```

in (7.4) does not make any sense. There is no way we can divide a vector $f(v_0+h)-f(v_0)\in\mathbb{R}^m$ by a vector $h\in\mathbb{R}^d$, unless of course $m=d=1$ as in (7.4), where we faced usual numbers.

The natural thing here is to generalize the definition in (7.3). First let us recall what functions $f:\mathbb{R}^d\to\mathbb{R}^m$ look like.

7.2 Vector functions

We will flesh out the general Definition 1.104 in a special case below.

A function $f:\mathbb{R}^d\to\mathbb{R}^m$ takes a vector $(x_1,\ldots,x_d)\in\mathbb{R}^d$ as input and gives a vector $(y_1,\ldots,y_m)\in\mathbb{R}^m$ as output. This means that every coordinate $y_1,\ldots,y_m$ in the output must be a function of $x_1,\ldots,x_d$ i.e.,

$$y_i=f_i(x_1,\ldots,x_d)$$

for $i=1,\ldots,m$. So in total, we may write f as

$$f(x_1,\ldots,x_d)=\begin{pmatrix}f_1(x_1,\ldots,x_d)\\ \vdots\\ f_m(x_1,\ldots,x_d)\end{pmatrix}. \tag{7.5}$$

Each of the (coordinate) functions f_i are functions from $\mathbb{R}^d$ to $\mathbb{R}$.

(7.1) EXERCISE.

Look back at Exercise 1.125. Write down precisely the vector function $h:\mathbb{R}^2\to\mathbb{R}^2$ occurring there. ♠

(7.2) EXERCISE.

The function $f:\mathbb{R}^2\to\mathbb{R}^2$ is rotating a vector 90 degrees counter clockwise. What are f_1 and f_2 in

$$f(x,y)=\begin{pmatrix}f_1(x,y)\\ f_2(x,y)\end{pmatrix}?$$

Hint. Try rotating some specific vectors like $(1,0),(0,1),(1,1)$ 90 degrees. Do you see a pattern?

♠

7.3 Differentiability

The definition of differentiability for a function $f:\mathbb{R}^d\to\mathbb{R}^m$ mimics (7.3), except that $\varepsilon(h)h$ is replaced by $\varepsilon(h)|h|$. Also the open interval (a,b) is replaced by an open subset U and the (open) interval $(-\delta,\delta)$ is replaced by an open subset O containing 0.

Notice, however, that now the derivative is a matrix!

(7.3) DEFINITION.

Let $f : U \to \mathbb{R}^m$ be a function with $U \subseteq \mathbb{R}^d$ an open subset. Then f is differentiable *at $v_0 \in U$ if there exists*

(i) an $m \times d$ matrix C,

(ii) an open subset $O \subseteq \mathbb{R}^d$ with $0 \in O$, such that $v_0 + h \in U$ for every $h \in O$,

(iii) a function $\varepsilon : O \to \mathbb{R}^m$ continuous at 0 with $\varepsilon(0) = 0$,

such that

$$f(v_0 + h) - f(v_0) = Ch + \varepsilon(h)\,|h|\,,$$

In this case, the $m \times d$ matrix C is called the (matrix) derivative of f at v_0 and denoted by $f'(v_0)$.
The function f is called differentiable if it is differentiable at every $v \in U$.

For one variable we saw that differentiability at x_0 means that the graph looks like a line when you zoom in around $(x_0, f(x_0))$. Definition 7.3 says the same thing one dimension up: for a function $f : \mathbb{R}^2 \to \mathbb{R}$ the graph $z = f(x,y)$ is a *surface* in $\mathbb{R}^3$, the derivative is a 1×2 matrix $C = (a\ b)$, and $z = f(v_0) + Ch$ describes a *plane* — the *tangent plane* at $v_0 = (x_0, y_0)$. Differentiability at v_0 means: zoom in on the point and the surface becomes indistinguishable from this plane.

You can watch this happen in the live cell below. The graph of $f(x,y) = x^3 - 3xy^2$ is drawn in blue together with its tangent plane at the black point, shown as a red grid. Run the cell and drag the slider to shrink the window around the point — the curved blue surface flattens into the red grid. Rotate the picture with the mouse to see it from all sides.

python cell — not displayed in the pdf version.

For a taste of what non-differentiable looks like, try the ice cream cone $f(x,y) = \sqrt{x^2 + y^2}$ at the point $(0,0)$: the cone looks exactly the same at every zoom level — its tip never flattens into a plane, no matter how far you drag the slider.

How do we compute the matrix derivative C in the above definition? We need to look at the representation of f in (7.5) and introduce the partial derivatives.

7.3.1 Partial derivatives

A function of one variable x has a derivative with respect to x. For a function of several variables $x_1, \ldots, x_d$ we have a well defined derivative with respect to each of these variables. These are called the partial derivatives (if they exist) and they are defined below.

(7.4) DEFINITION.

Let $f : U \to \mathbb{R}$ be a function, where U is an open subset of $\mathbb{R}^d$. Fix a point $v = (a_1, a_2, \ldots, a_d) \in U$ and let

$$p_i = f(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_d)$$

for $i = 1, \ldots, d$. If p_i is differentiable at $x = a_i$ according to Definition 6.17 (recalled in (7.3)), then we say that the partial derivative *of f with respect to x_i exists at $v \in U$ and use the notation*

$$\frac{\partial f}{\partial x_i}(v) := p_i'(a_i).$$

(7.5) REMARK.

The partial derivative with respect to a specific variable is computed by letting all the other variables appear as constants.

To get a feeling for the definition and computation of partial derivatives, take a look at the example below, where we compute using the classical (geometric) definition of the one variable derivative.

(7.6) EXAMPLE.

Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x_1, x_2) = x_1 x_2^2 + x_1.$$

Then

$$\begin{aligned}\frac{\partial f}{\partial x_2}(v) &= \lim_{\delta \to 0} \frac{f(x_1, x_2+\delta) - f(x_1, x_2)}{\delta} \\ &= \lim_{\delta \to 0} \frac{x_1(x_2+\delta)^2 + x_1 - (x_1 x_2^2 + x_1)}{\delta} \\ &= x_1 \lim_{\delta \to 0} \frac{(x_2+\delta)^2 - x_2^2}{\delta} = x_1 \lim_{\delta \to 0} (2x_2 + \delta) = 2x_1 x_2,\end{aligned}$$

where $v = (x_1, x_2)$. This example illustrates that $\frac{\partial f}{\partial x_i}$ can be computed just like in the one variable case, when the other variables ($\neq x_i$) are treated as constants. Notice that

$$\frac{\partial}{\partial x_1}\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}\frac{\partial f}{\partial x_1} = 2x_2.$$

♠

Partial derivatives behave almost like the usual derivatives of one variable functions. You simply fix one variable that you consider the "real" variable and treat the other variables as constants.

(7.7) EXAMPLE.

$$\frac{\partial}{\partial x}\left(\sin(xy) + x^2 y^2 + y\right) = y\cos(xy) + 2xy^2.$$

♠

Below are examples of python code computing partial derivatives. Notice that the variables must be declared first.

python cell — not displayed in the pdf version.

The computations above point to a really surprising result. It seems that it makes no difference if you compute the partial derivative with respect to x_1 and then with respect to x_2 or the other way around. You could, just for fun, try this out on the more complicated function

$$f(x_1, x_2) = x_1 x_2^2 + \cos(\sin(x_1 x_2) + \log(x_1^{17} x_2)).$$

This result is formulated in Theorem 7.11 below.

(7.8) EXERCISE.

Use the python cell above to verify the computation of the partial derivative in Example 7.7. ♠

The following result tells us how to compute the matrix derivative.

(7.9) PROPOSITION.

Let $f : U \to \mathbb{R}^m$ be a function with $U \subseteq \mathbb{R}^d$ an open subset. If f is differentiable at $v_0 \in U$, then the partial derivatives

$$\frac{\partial f_i}{\partial x_j}(v_0)$$

exist for $i = 1, \ldots, m$ and $j = 1, \ldots, d$ and the matrix C in Definition 7.3 is

$$C = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(v_0) & \cdots & \frac{\partial f_1}{\partial x_d}(v_0) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(v_0) & \cdots & \frac{\partial f_m}{\partial x_d}(v_0) \end{pmatrix}.$$

Proof. The j-th column in C is Ce_j. Putting $h = \delta e_j$ for $\delta \in \mathbb{R}$ in Definition 7.3 gives

$$f(v_0 + \delta e_j) - f(v_0) = \delta C e_j + \varepsilon(\delta e_j)\,|\delta|\,.$$

The i-th coordinate of this identity of m-dimensional vectors can be written

$$f_i(v_0 + \delta e_j) - f_i(v_0) = \delta C_{ij} + \tilde{\varepsilon}_i(\delta)\delta \tag{7.6}$$

where

$$\tilde{\varepsilon}_i(\delta) = \begin{cases} \varepsilon_i(\delta e_j)\dfrac{|\delta|}{\delta} & \text{if } \delta \neq 0 \\ 0 & \text{if } \delta = 0 \end{cases}$$

and (7.6) shows that $C_{ij} = \frac{\partial f_i}{\partial x_j}(v_0)$. □

(7.10) EXERCISE.

Compute the matrix derivative of the vector function in Exercise 7.2. ♠

For a function $f : U \to \mathbb{R}$ with $U \subseteq \mathbb{R}^d$ an open subset, the partial derivative, if it exists for every $x \in U$, is a new function

$$\frac{\partial f}{\partial x_j} : U \to \mathbb{R}.$$

We will use the notation

$$\frac{\partial^2 f}{\partial x_i \partial x_j} := \frac{\partial}{\partial x_i}\frac{\partial f}{\partial x_j}$$

for the *iterated (second order) partial derivative*.

The first part of following result is a converse to Proposition 7.9. The second part contains the surprising *symmetry of the second order partial derivatives* under rather mild conditions. We will not go into the proof of this result, which is known as Clairaut's theorem.

(7.11) THEOREM.

Let $f : U \to \mathbb{R}^m$ be a function with $U \subseteq \mathbb{R}^d$ an open subset. If the partial derivatives for f exist at every $x \in U$ with

$$\frac{\partial f_i}{\partial x_j}$$

continuous (for $i = 1, \ldots, m$ and $j = 1, \ldots, d$), then f is differentiable. If the second order partial derivatives exist for a function $f : U \to \mathbb{R}$ and are continuous functions, then

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$$

for $i, j = 1, \ldots, d$.

(7.12) EXERCISE.

Verify (by hand!) the symmetry of the second order partial derivatives for the function f in Example 7.7 i.e., show that

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}.$$

♠

(7.13) EXERCISE.

Verify that $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x,y) = \frac{x^2 y}{1+y^2}$$

is a differentiable function by computing

$$\frac{\partial f}{\partial x} \qquad \text{and} \qquad \frac{\partial f}{\partial y}$$

and applying Theorem 7.11. Check also that

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}.$$

♠

7.4 Newton-Raphson in several variables!

There is a beautiful generalization of the Newton-Raphson method to several variable functions $f : \mathbb{R}^d \to \mathbb{R}^d$. Consider first that you would like to solve the system

$$y^2 - x^3 + x = 0 \tag{7.7}$$
$$y^3 - x^2 = 0 \tag{7.8}$$

of non-linear equations in the two variables x and y. Notice that we are talking *non-linear* here. This is so much more difficult than the systems of linear equations that you encountered in a previous chapter.

However, just like we used Newton's method in one variable for solving a non-linear equation, Newton's method for finding a zero for a function $f : \mathbb{R}^d \to \mathbb{R}^d$ generalized to the iterative scheme

$$v_{i+1} = v_i - f'(v_i)^{-1} f(v_i) \tag{7.9}$$

provided that the $d \times d$ matrix derivative $f'(v_i)$ is invertible.

The reason that (7.9) works comes again from the powerful definition of differentiability in Definition 7.3 using that

$$f(v) - f(v_0) \qquad \text{is close to} \qquad f'(v_0)(v - v_0) \tag{7.10}$$

provided that $h = v - v_0$ is small. In fact, you get (again) (7.9) from (7.10) by putting $f(v)$ to 0, replacing *is close to* by $=$ and then isolating v.

For the equations in (7.7), the iteration scheme (7.9) becomes

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} -3x_i^2 + 1 & 2y_i \\ -2x_i & 3y_i^2 \end{pmatrix}^{-1} \begin{pmatrix} y_i^2 - x_i^3 + x_i \\ y_i^3 - x_i^2 \end{pmatrix}. \tag{7.11}$$

(7.14) EXERCISE.

Verify the claim in (7.11) by applying (7.9) to

$$f(x,y) = \begin{pmatrix} y^2 - x^3 + x \\ y^3 - x^2 \end{pmatrix}.$$

Carry out sufficiently many iterations starting with the vector $(1,1)$ in (7.11) to see the iteration stabilize. You should do this using a computer, for example by modifying the python code in the last half of Example 7.16. ♠

7.5 Local extrema in several variables

For a function $f : U \to \mathbb{R}$, where $U \subseteq \mathbb{R}^d$, the derivative $f'(v)$ at $v \in U$ is called *the gradient* for f at v. Classically, it is denoted $\nabla f(v)$ i.e.,

$$\nabla f(v) = \left(\frac{\partial f}{\partial x_1}(v) \dots \frac{\partial f}{\partial x_d}(v) \right).$$

The definition below is inspired by the one variable case (see Definition 6.36).

(7.15) DEFINITION.

Let $f : U \to \mathbb{R}$ be a function, where $U \subseteq \mathbb{R}^d$ is an open subset. Suppose that the partial derivatives exist at $v_0 \in U$. Then v_0 is called a critical point *for f if $\nabla f(v_0) = 0$.*

(7.16) EXAMPLE.

Consider the function $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2x + 3\log(y) \\ 3x/y - 3y^2 \end{pmatrix}$$

corresponding to finding critical points for the function

$$g(x,y) = x^2 + 3x\log(y) - y^3. \tag{7.12}$$

You can left click and hold the graph computed below (after it has rendered) and rotate the surface to get a feeling for what (7.12) looks like. Zooming in is also possible.

python cell — not displayed in the pdf version.

Here

$$f' = \begin{pmatrix} 2 & 3/y \\ 3/y & -3x/y^2 - 6y \end{pmatrix}.$$

In the python code below, Newton's method is started at $(1,1)$ and iterated four times.

python cell — not displayed in the pdf version.

♠

If v_0 is not a critical point for f we can use the gradient vector to move in a direction making f strictly smaller/larger. This is very important for optimization problems.

(7.17) LEMMA.

Let $f : U \to \mathbb{R}$ be a differentiable function, where $U \subseteq \mathbb{R}^d$ is an open subset. Suppose that $u \in \mathbb{R}^d$ and $\nabla f(v_0)u < 0$ for $v_0 \in U$. Then

$$f(v_0 + \lambda u) < f(v_0)$$

for $\lambda > 0$ small.

Proof. By the differentiability of f,

$$f(v_0 + u) - f(v_0) = \nabla f(v_0)u + \varepsilon(u)|u|,$$

where $\varepsilon : \mathbb{R}^d \to \mathbb{R}$ is a function satisfying $\varepsilon(h) \to 0$ for $h \to 0$. For $\lambda > 0$ with $v_0 + \lambda u \in U$ we have

$$f(v_0 + \lambda u) - f(v_0) = \lambda(\nabla f(v_0)u + \varepsilon(\lambda u)|u|).$$

When λ tends to zero from the right, it follows that $f(v_0 + \lambda u) - f(v_0) < 0$ for small $\lambda > 0$. □

Lemma 7.17 looks innocent, but it is the bread and butter in the training of neural networks. In mathematical terms, training means minimizing a function. In machine learning terms, λ above is called the *learning rate*. One iteration (why do I choose $u = -\nabla f(x)$?)

$$x - \lambda \nabla f(x)$$

of Lemma 7.17 is the central ingredient in an *epoch* in training a neural network.

(7.18) EXAMPLE.

Let us briefly pause and see Lemma 7.17 in action. Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x,y) = x^2 + y^2$$

and $v_0 = \begin{pmatrix}1\\1\end{pmatrix}$ with $u = \begin{pmatrix}-1\\0\end{pmatrix}$. In this case $\nabla f(v_0) = \begin{pmatrix}2 & 2\end{pmatrix}$ and $\nabla f(v_0)u = -2 < 0$. Therefore we may find a small $\lambda > 0$, such that $f(v_0 + \lambda u) < f(v_0)$. How do we choose λ optimally? If λ is too big we fail and land up in a worse point than v_0. Here

$$f(v_0 + \lambda u) = (1-\lambda)^2 + 1$$

This is a quadratic polynomial, which is minimal for $\lambda = 1$. Therefore the minimal value reached in the direction of u is 1. The process now continues replacing v_0 by $v_0 + 1 \cdot u$. ♠

7.5.1 Gradient descent

Lemma 7.17 turns into an algorithm the moment we choose the direction $u = -\nabla f(v)^\top$: then $\nabla f(v)\, u = -|\nabla f(v)|^2 < 0$, so the lemma applies at every non-critical point. The result is the single most important algorithm in machine learning.

Gradient descent. Choose a starting point $v \in \mathbb{R}^d$ and a learning rate $\lambda > 0$. Repeat

$$v := v - \lambda\, \nabla f(v)^\top$$

until $|\nabla f(v)|$ is sufficiently small.

Two things are yours to choose: the starting point and the learning rate. The learning rate is a genuine dilemma. Chosen too small, the algorithm crawls; chosen too big, the steps overshoot the valley and the algorithm can even explode. You can experience all three regimes in the cell below, which runs gradient descent on

$$f(x,y) = x^2 + 10y^2$$

— a stretched bowl with its minimum at $(0,0)$ — and draws the path on top of the level curves of f. With $\lambda = 0.09$ the path zigzags across the narrow valley before settling at the bottom. Try $\lambda = 0.01$ (patient crawling) and $\lambda = 0.105$ (disaster).

python cell — not displayed in the pdf version.

(7.19) QUIZ.

quiz — not displayed in the pdf version.

♠

The result below is the multi variable generalization of looking for local extrema by putting $f' = 0$ in the one variable case.

(7.20) PROPOSITION.

Let $f : U \to \mathbb{R}$ be a differentiable function, where $U \subseteq \mathbb{R}^d$ is an open subset. If $v_0 \in U$ is a local extremum, then v_0 is a critical point for f.

Proof. Suppose that $\nabla f(v_0) \neq 0$. If v_0 is a local minimum, then we may use $u = -\nabla f(v_0)$ in Lemma 7.17 to deduce that $f(v_0 + \lambda u) < f(v_0)$ for $\lambda > 0$ small. This contradicts the local minimality of v_0. If v_0 is a local

maximum we can apply Lemma 7.17 with $-f$ and $u = \nabla f(v_0)$ to reach a similar contradiction. Therefore $\nabla f(v_0) = 0$ and v_0 is a critical point for f. □

(7.21) EXERCISE.

Compute the critical points of

$$f(x,y) = x^3 + xy + y^3.$$

Is $(0,0)$ a local maximum or a local minimum for f?

Hint. Look at

$$f_1(t) = f(t,t)$$
$$f_2(t) = f(t,-t)$$

and $f_1''(0)$ and $f_2''(0)$ (along with Theorem 6.50).

♠

(7.22) EXERCISE.

We will prove later that a differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$ is strictly convex if the socalled Hessian matrix given by

$$\begin{pmatrix} \dfrac{\partial^2 f}{\partial x^2}(v) & \dfrac{\partial^2 f}{\partial x \partial y}(v) \\ \dfrac{\partial^2 f}{\partial y \partial x}(v) & \dfrac{\partial^2 f}{\partial y^2}(v) \end{pmatrix}$$

is positive definite for every $v \in \mathbb{R}^2$. This is a multivariable generalization of the fact that $g : \mathbb{R} \to \mathbb{R}$ is strictly convex if $g''(x) > 0$ for every $x \in \mathbb{R}$.

Now let

$$f(x,y) = x^2 + y^2 - \cos(x) - \sin(y). \tag{7.13}$$

3D graph. You can left click the surface computed below after it has rendered and rotate or zoom in.

python cell — not displayed in the pdf version.

(a) Show that f is strictly convex.

(b) Compute the critical point(s) of f.

Hint. This is a numerical computation! Modify the relevant python cell for Newton's method in the previous chapter to do it.

(c) For a differentiable convex function $f : \mathbb{R}^2 \to \mathbb{R}$ we have in general that

$$f(v) \geq f(u) + \nabla f(u)(v-u) \tag{7.14}$$

for every $u, v \in \mathbb{R}^2$. This is a multivariable generalization of Theorem 6.61.

Explain how one can use (7.14) to find a global minimum for the function f in (7.13). Is this minimum unique? Is $f(x,y) \geq -1$ for every $x, y \in \mathbb{R}$?

♠

7.6 How a nudge propagates

Before we state the chain rule in several variables, let us understand it the physicist's way. Forget graphs and slopes for a moment and reread the definition of the derivative in (7.3) as a statement about cause and effect:

if you nudge the input of f by a tiny amount δ, the output moves by $f'(x_0)\,\delta$ — plus junk that is tiny compared to δ.

In symbols,

$$f(x_0+\delta)-f(x_0)=\underbrace{f'(x_0)\,\delta}_{\text{the response}}+\underbrace{\varepsilon(\delta)\,\delta}_{\text{the junk}},$$

where the junk has the redeeming feature that $\varepsilon(\delta)\to 0$: divide it by δ and it still vanishes. So the derivative is an *exchange rate* between nudges: wiggle in, wiggle out, at the rate $f'(x_0)$.

Machines in series

Now put two machines in a row: x goes into g, and what comes out goes into f,

$$x \longrightarrow \boxed{g} \longrightarrow \boxed{f} \longrightarrow z.$$

Nudge x by δ. The first machine responds with a nudge of $g'(x_0)\,\delta$ on its output — but that *is* a nudge on the input of f, which therefore responds with

$$f'(g(x_0))\cdot\big(g'(x_0)\,\delta\big).$$

The total exchange rate from x to z is the *product* of the two rates. That is the chain rule. There is nothing more to it: exchange rates in series multiply — just as two gearboxes coupled in a row multiply their gear ratios.

You do not have to take this on faith. The cell below measures all three rates by actually nudging.

python cell — not displayed in the pdf version.

Where did the junk go? The nudge arriving at f is not exactly $g'(x_0)\,\delta$ — it carries junk. But feeding junk through f multiplies it by f' and adds junk of its own, and junk times a constant, or junk of junk, is still tiny compared to δ. The honest bookkeeping of exactly this is the entire proof of the chain rule; the *idea* is nothing more than rates in series multiply.

Wires in parallel

Several variables add one — and only one — new phenomenon. Take a function $F(x,y)$ of two variables and nudge *both* inputs at once, x by δ_1 and y by δ_2. Do it in two steps: first nudge x, then y. The first step responds with $\frac{\partial F}{\partial x}\,\delta_1$. The second step responds with $\frac{\partial F}{\partial y}\,\delta_2$ — evaluated at the slightly nudged point, but being off by a tiny amount in *where* we read the rate changes the response only by junk. Altogether,

$$F(x_0+\delta_1,y_0+\delta_2)-F(x_0,y_0)=\frac{\partial F}{\partial x}\,\delta_1+\frac{\partial F}{\partial y}\,\delta_2+\text{junk}.$$

Nudges arriving on different wires simply add.

Series and parallel is all there is. Suppose x feeds two intermediate quantities u and v, which both feed z:

$$x \longrightarrow \begin{array}{|c|}\hline u \\ \hline v \\ \hline\end \longrightarrow z.$$

Nudge x by δ. Along the upper route, u moves by $\frac{\partial u}{\partial x}\delta$, and z responds to that with $\frac{\partial z}{\partial u}\frac{\partial u}{\partial x}\delta$ — rates in series multiply. The lower route contributes $\frac{\partial z}{\partial v}\frac{\partial v}{\partial x}\delta$. The two nudges arrive at z on different wires, so they add:

$$\frac{dz}{dx}=\frac{\partial z}{\partial u}\frac{\partial u}{\partial x}+\frac{\partial z}{\partial v}\frac{\partial v}{\partial x}.$$

Multiply the rates along each route, then add over the routes.

This slogan is the entire chain rule in several variables. The next section dresses it in matrices — and the fit is perfect, because the entry of a matrix product is precisely a sum over intermediate wires of products of rates. Matrix multiplication could have been invented for this purpose.

Let us measure a two-route example. The function

$$F(x,y) = \sin(xy) + x^2y^2 + y$$

will follow us for the rest of this chapter. Nudging x at the point $(1,1)$ and reading off the exchange rate gives a number you should remember — you will meet it again twice, computed by two entirely different methods.

python cell — not displayed in the pdf version.

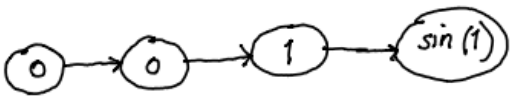
Why machine learning cares

A neural network is nothing but millions of these little machines wired in series and in parallel, and *training* it means nudging. Tweak one weight w buried deep inside the network by δ, and the tweak propagates: multiplied by a local rate at every machine it passes through, added up wherever routes merge, until it arrives at the loss L as the response $\frac{\partial L}{\partial w}\,\delta$. A parameter with a large exchange rate is worth nudging; one with rate zero is not — and gradient descent (Lemma 7.17) nudges every parameter against its own rate, all at once. The backward pass you will implement in Section 7.7 and use for real in Section 7.9 is just the systematic collection of all these exchange rates in a single sweep.

7.7 The chain rule

Suppose you want to compute the value of the function $h(x) = \sin(\cos(2x))$ for $x = 0$. Then you would start by evaluating the inner function $2x$, then applying cos and finally sin. This computation can be illustrated in the (computational) graph

where you plug $x = 0$ into the leftmost node and fill in each node taking input from its left neighbor



Suppose we want to compute $h'(x)$ for $x = 0$. Can we use the computational graph for this?

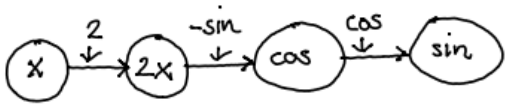
Recall the chain rule for functions of one variable. Here we have functions $f : (a,b) \to \mathbb{R}$ and $g : (c,d) \to \mathbb{R}$, such that $g(x) \in (a,b)$ for $x \in (c,d)$. If g is differentiable at $x_0 \in (c,d)$ and f is differentiable at $g(x_0) \in (a,b)$, the chain rule says that $f \circ g$ is differentiable at x_0 with

$$(f \circ g)'(x_0) = f'(g(x_0))g'(x_0).$$

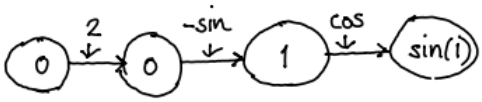
The chain rule tells us that

$$h'(x) = \cos(\cos(2x))(-\sin(2x))2.$$

This expression involves three derivatives corresponding to the three edges in the computational graph. We can illustrate the chain rule by labeling each edge with the derivative of its end node:



Then the derivative $h'(0)$ can be computed by as the product of the labels evaluated on their left nodes in the filled in computational graph:



i.e., $h'(0) = \cos(1)(-\sin(0))2$. This observation is the basis of the famous backpropagation rule used in training neural networks.

The chain rule for functions of one variable generalizes verbatim to functions of several variables:

$$(f \circ g)'(x_0) = f'(g(x_0))g'(x_0)$$

for compatible multivariate functions f and g when you replace usual multiplication by matrix multiplication.

(7.23) THEOREM.

Let $f : U \to \mathbb{R}^m$ and $g : V \to \mathbb{R}^d$ with $U \subseteq \mathbb{R}^d$, $V \subseteq \mathbb{R}^l$ open subsets and $g(V) \subseteq U$. If g is differentiable at $v_0 \in V$ and f is differentiable at $g(v_0) \in U$, then $f \circ g$ is differentiable at v_0 with

$$(f \circ g)'(v_0) = f'(g(v_0))g'(v_0). \tag{7.15}$$

The proof of the chain rule in this general setting uses Definition 7.3 just as in the one variable case. It is not conceptually difficult, but severely cumbersome. We will not give it here.

7.7.1 Matrix multiplication graphically

To really understand the chain rule, it pays to view the matrix multiplication in (7.15) in a new light (inspired by computer science and neural networks).

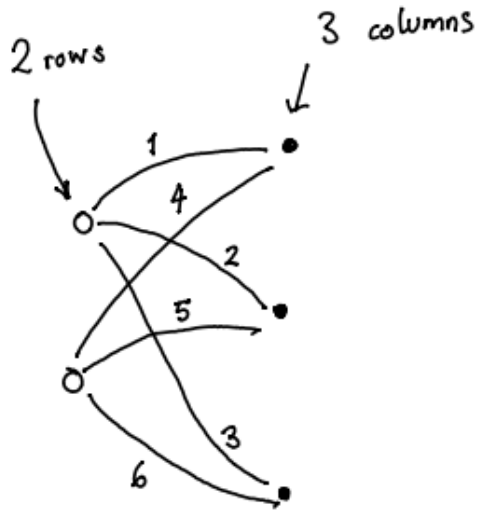
An $m \times n$ matrix (a_{ij}) is a rectangular table with m rows and n columns containing mn numbers. We may also view it as a (bipartite) graph with m left nodes, n right nodes and an edge from the left node i to the right node j with weight a_{ij}. This is best illustrated by an example, which also tells you how matrix multiplication is (beautifully) interpreted in this setting.

(7.24) EXAMPLE.

The 2×3 matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

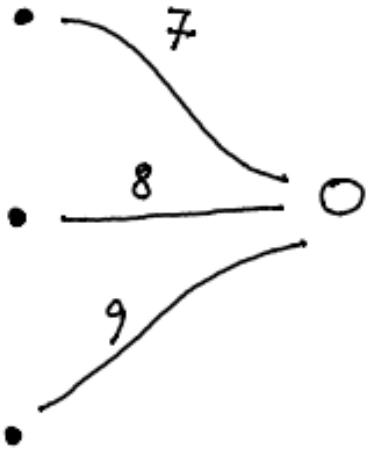
is represented below as a graph



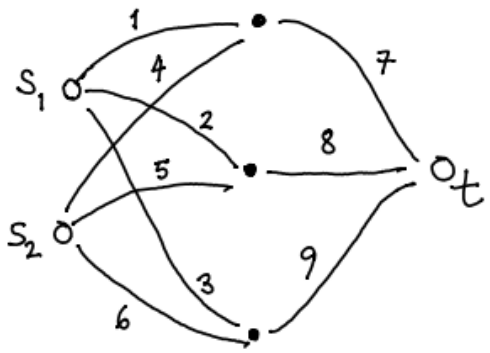
Similarly the 3×1 matrix

$$B = \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}$$

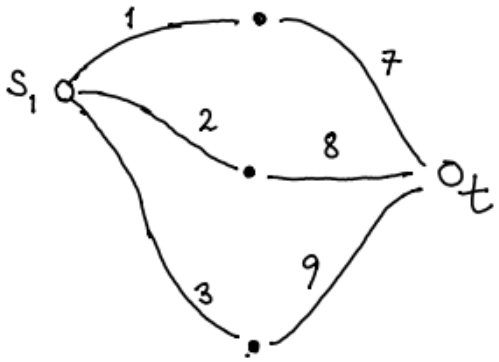
is represented as



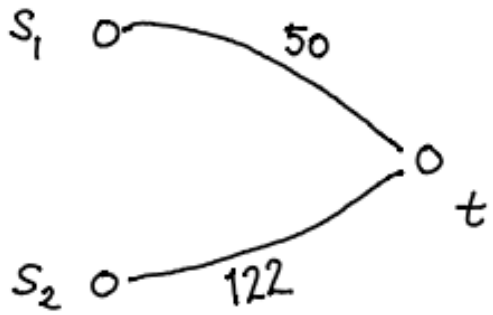
You know that the matrix product AB is a 2×1 matrix. Let us line A and B up graphically:



There are three paths from s_1 to t and three paths from s_2 to t. Here are the three paths from s_1 to t:



Finally, the matrix product AB is represented by the graph



The number 50 on the edge from s_1 to t is gotten by adding the products of the weights on each of the three paths from s_1 to t i.e., $1 \cdot 7 + 2 \cdot 8 + 3 \cdot 9$. This is the graphical interpretation of matrix multiplication!

♠

7.7.2 Unpacking the chain rule

The matrix multiplication in (7.15) looks deceivingly simple. Let us write it out. Assume for simplicity that $g : \mathbb{R}^l \to \mathbb{R}^n$ is a function in the variables $x_1, \ldots, x_l$ and that $f : \mathbb{R}^n \to \mathbb{R}^m$ is a function in the variables $y_1, \ldots, y_n$:

$$g(x_1,\ldots,x_l) = \begin{pmatrix} g_1(x_1,\ldots,x_l) \\ \vdots \\ g_n(x_1,\ldots,x_l) \end{pmatrix} \qquad \text{and} \qquad f(y_1,\ldots,y_n) = \begin{pmatrix} f_1(y_1,\ldots,y_n) \\ \vdots \\ f_m(y_1,\ldots,y_n) \end{pmatrix}.$$

Then $h = (f \circ g) : \mathbb{R}^l \to \mathbb{R}^m$ is a function in the variables $x_1, \ldots, x_l$:

$$h(x_1,\ldots,x_l) = \begin{pmatrix} h_1(x_1,\ldots,x_l) \\ \vdots \\ h_m(x_1,\ldots,x_l) \end{pmatrix}$$

and we wish to compute $h'(v_0)$, which is an $m \times l$ matrix with entries

$$\frac{\partial h_i}{\partial x_j}(v_0),$$

where $i = 1, \ldots, m$ represent the rows and $j = 1, \ldots, l$ the columns. Here (7.15) says that

$$\frac{\partial h_i}{\partial x_j}(v_0) = \frac{\partial f_i}{\partial y_1}(g(v_0))\frac{\partial g_1}{\partial x_j}(v_0) + \cdots + \frac{\partial f_i}{\partial y_n}(g(v_0))\frac{\partial g_n}{\partial x_j}(v_0).$$

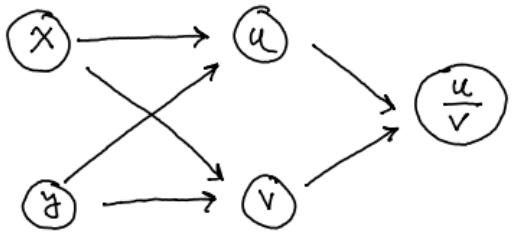
When using the chain rule in computations it pays to use the graphical interpretation of matrix multiplication in subsection 7.7.1 with edges labeled by the derivatives in a computational graph. We illustrate this below.

(7.25) EXAMPLE.

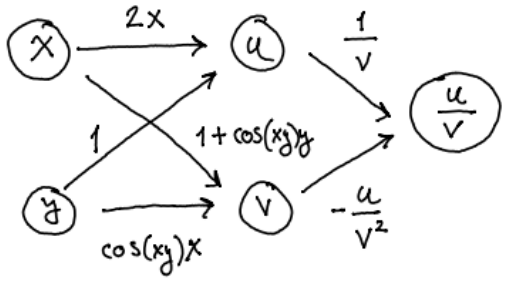
The function $h : \mathbb{R}^2 \to \mathbb{R}$ given by

$$h(x,y) = \frac{x^2 + y}{x + \sin(xy)}$$

may be evaluated using the computational graph



where u is the function $u(x,y) = x^2 + y$ and v is the function $v(x,y) = x + \sin(xy)$. Similar to the one variable case discussed in the beginning of section 7.7, we label each edge, but now by the partial derivative of the function in its ending node with respect to the variable in its beginning node:



From the graphical interpretation of the matrix product and the chain rule you follow the two paths from the input node x to the output node u/v and conclude that

$$\frac{\partial h}{\partial x} = \frac{2x}{x+\sin(xy)} - \frac{x^2+y}{(x+\sin(xy))^2}(1+\cos(xy)y).$$

♠

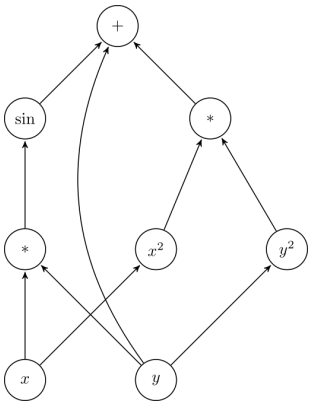
Here is another example of the chain rule in action through a computational graph. In the end you will see an implementation in a famous python library.

(7.26) EXAMPLE.

Consider the example

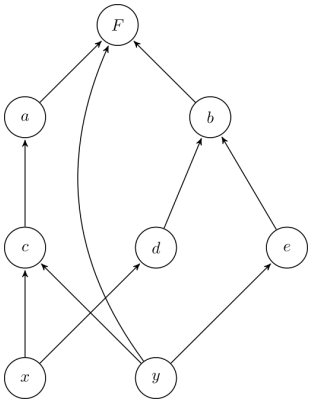
$$f(x,y) = \sin(xy) + x^2y^2 + y$$

from Example 7.7. Even though $f(x,y)$ superficially looks rather simple, it is composed of several smaller functions as displayed in the computational graph



Every node in the above graph, except the input nodes (with no ingoing arrows), represents some function $f : \mathbb{R}^d \to \mathbb{R}^m$. For example the node sin represents a function $f : \mathbb{R} \to \mathbb{R}$ and $*$ represents a function $f : \mathbb{R}^2 \to \mathbb{R}$.

To emphasize that the non-input nodes really are functions we replace them by letters:



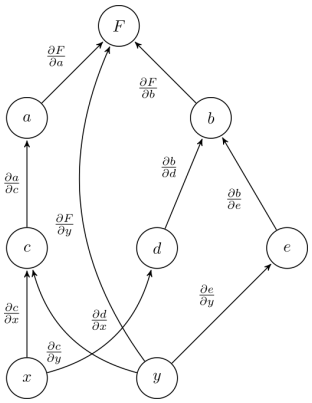
Here we see that

$$f(x,y) = F(a(c(x,y)),y,b(d(x),e(y))),$$

where

$$\begin{aligned} F(a,y,b) &= a+y+b \\ a(c) &= \sin(c) \\ c(x,y) &= xy \\ b(d,e) &= de \\ d(x) &= x^2 \\ e(y) &= y^2 \end{aligned}$$

The gradient is then available from the decorated graph below



by multiplying the decorations on each path from the top to the input variable and the summing up. For example,

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial a}\frac{\partial a}{\partial c}\frac{\partial c}{\partial x} + \frac{\partial F}{\partial b}\frac{\partial b}{\partial d}\frac{\partial d}{\partial x}.$$

Computational graphs and the chain rule are important components in machine learning libraries. Below is an example of the computation of $\frac{\partial F}{\partial x}$ in the computational graph above using the `sympy` library.

python cell — not displayed in the pdf version.

♠

7.7.3 Backpropagation from scratch

There is no magic in how `sympy` — or `pytorch` — computes these derivatives. The complete mechanism is the recipe you just used by hand: label every edge of the computational graph with the derivative of its end node with respect to its beginning node, multiply the labels along each path from the output down to an input, and add up the products. The program below implements exactly this and nothing more, in about twenty five lines.

A `Node` holds a number together with the edges of the computational graph: which nodes it was computed from, and the label on each incoming edge. Each operation writes down its own edge labels — for instance $d(uv)/du = v$ for a product. The function `backprop` then walks every path backwards from the output, multiplying the labels as it goes and adding the product into `grad` when it reaches a node.

python cell — not displayed in the pdf version.

Compare with the `sympy` window above: the same numbers come out. Two details are worth savoring. First, $d = x \cdot x$ records *two* edges into x, each with label x — so the two paths contribute $x + x = 2x$, and the power rule appears all by itself. Second, `backprop` is the graphical matrix multiplication of subsection 7.7.1 in executable form: every path is followed, the labels along it are multiplied, and parallel paths are added.

When you train neural networks in Section 7.9, the backward pass is this walk written out with matrices. And `pytorch`'s celebrated *autograd* is again this walk, organized so that each node is visited only once (in reverse order of computation), which is what lets it scale to graphs with billions of nodes.

(7.27) EXERCISE.

Construct a computational graph for

$$f(x,y) = x^3 + xy + y^3$$

and detail the computation of the gradient ∇f in this context.

Compute the gradient of f at $(x,y) = (1,1)$ using the `sympy` window above (or `pytorch` in Google Colab). ♠

(7.28) EXERCISE.

Consider $f : \mathbb{R} \to \mathbb{R}^3$ and $g : \mathbb{R}^3 \to \mathbb{R}$ given by

$$f(t) = \begin{pmatrix} t \\ t^2 \\ t^3 \end{pmatrix} \qquad \text{and} \qquad g\begin{pmatrix} x \\ y \\ z \end{pmatrix} = x^2 + 3y^6 + 2z^5.$$

Compute $(g \circ f)'(t)$ using the chain rule and check the result with an explicit computation of the derivative of $g \circ f : \mathbb{R} \to \mathbb{R}$. ♠

(7.29) EXERCISE.

We wish to show that the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x,y) = x^2 + y^2$$

is convex. This means that we need to prove that

$$f((1-t)x_0 + tx_1, (1-t)y_0 + ty_1) \leq (1-t)f(x_0,y_0) + tf(x_1,y_1)$$

for every $(x_0.y_0), (x_1,y_1) \in \mathbb{R}^2$ and every t with $0 \leq t \leq 1$. This can be accomplished from the one variable case in the following way. Define

$$g(t) = f((1-t)x_0 + tx_1, (1-t)y_0 + ty_1)$$

and show that g is convex by using the chain rule to show that $g''(t) \geq 0$. Show how the convexity of f follows from this by using that

$$g(t) = g((1-t)\cdot 0 + t \cdot 1).$$

♠

7.8 Logistic regression

The beauty of the sigmoid function is that it takes any value $x \in \mathbb{R}$ and turns it into a probability $0 < \sigma(x) < 1$ by

$$\sigma(x) = \frac{1}{1+e^{-x}},$$

i.e., $\sigma(-\infty) = 0$ and $\sigma(\infty) = 1$.

Graph of the sigmoid function.

python cell — not displayed in the pdf version.

(7.30) EXERCISE.

Prove that

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

and

$$\log \frac{\sigma(x)}{1-\sigma(x)} = x.$$

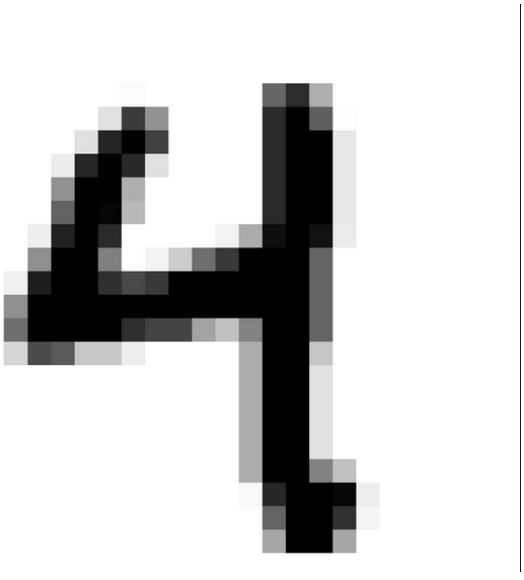
♠

We will not go into all the details (some of which can be traced to introductory probability and statistics), but suppose that we have an outcome E, which may or may not happen.

We have an idea, that the probability of E is dependent on certain parameters $w_0, w_1, \ldots, w_n$ and observations $x_1, \ldots, x_n$ that fit into the sigmoid function as

$$p(x_1, \ldots, x_n) = \sigma(w_0 + w_1 x_1 + \cdots + w_n x_n) = \frac{1}{1+e^{-w_0 - w_1 x_1 - \cdots - w_n x_n}}. \tag{7.16}$$

An example of this could be where $x_1, \ldots, x_{784}$ denote the gray scale of each pixel in a 28×28 image. The event E is whether the image contains the digit 4:



Here $p(x_1, \ldots, x_{784})$ would be the probability that the image contains the digit 4.

7.8.1 Estimating the parameters

Suppose also that we have a table of observations (data set)

$$\begin{array}{cccc} x_{11} & \cdots & x_{1n} & E_1 \\ x_{21} & \cdots & x_{2n} & E_2 \\ \vdots & \ddots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mn} & E_m, \end{array} \tag{7.17}$$

where each row has observations $x_{i1},\dots,x_{in}$ along with a binary variable E_i, which is 1 if E was observed to occur and 0 if not.

Assuming that (7.16) holds, the probability of observing the m observations in (7.17) is

$$\prod_{i=1}^{m} p(x_{i1},\dots,x_{in})^{E_i}(1-p(x_{i1},\dots,x_{in}))^{1-E_i}. \tag{7.18}$$

Notice that (7.18) is a function $L(w_0,\dots,w_n)$ of the parameters $w_0,w_1,\dots,w_n$ for fixed observations x_{ij}.

We wish to choose the parameters so that $L(w_0,w_1,\dots,w_n)$ is maximized (this is called maximum likelihood estimation). So we are in fact here, dealing with an optimization problem, which is usually solved by gradient descent (for $-L$) or solving the equations

$$\nabla L(w_0,w_1,\dots,w_n)=0.$$

Instead of maximizing $L(w_0,\dots,w_n)$ one usually maximizes the logarithm

$$\begin{aligned}\ell(w_0,w_1,\dots,w_n) &= \log L(w_0,w_1,\dots,w_n)\\ &= \sum_{i=1}^{m} E_i \log p(x_{i1},\dots,x_{in}) + (1-E_i)\log(1-p(x_{i1},\dots,x_{in}))\\ &= \sum_{i=1}^{m} E_i(w_0+w_1x_{i1}+\cdots+w_nx_{in}) - \log(1+e^{w_0+w_1x_{i1}+\cdots+w_nx_{in}}).\end{aligned}$$

Notice that we have used Exercise 7.30 and the logarithm rules $\log(ab)=\log(a)+\log(b)$ and $\log(a/b)=\log(a)-\log(b)$ in the computation above.

(7.31) EXAMPLE.

Suppose that the event E is assumed to be dependent on only one observation x i.e., $n=1$ above. For example, E could be the event of not showing up on a Monday paired with the amount of sleep x in the weekend.

Here

$$p(x)=\sigma(\alpha+\beta x)$$

and

$$\begin{aligned}\ell(\alpha,\beta) &= \sum_{i=1}^{m} E_i \log p(x_i) + (1-E_i)\log(1-p(x_i))\\ &= \sum_{i=1}^{m} E_i(\alpha+\beta x_i) - \log(1+e^{\alpha+\beta x_i}).\end{aligned}$$

♠

(7.32) EXERCISE.

Explain how the end result of the computation of $\ell(\alpha,\beta)$ in Example 7.31 is obtained and compute $\nabla\ell(\alpha,\beta)$.

♠

(7.33) EXAMPLE.

I remember exactly where I was when first hearing about the Challenger[2] disaster in 1986.

[2] See byuistats.github.io for more details on this example

VIDEO: https://youtu.be/fSTrmJtHLFU

This dreadful event was caused by failure of a socalled O-ring. The O-rings had been tested before the launch for failure (=1 below) at different temperatures (in F) resulting in the (partial) table below.

$$\begin{array}{cc} 53.0 & 1 \\ 56.0 & 1 \\ 57.0 & 1 \\ 63.0 & 0 \\ \vdots & \vdots \\ 70.0 & 0 \\ 70.0 & 1 \\ \vdots & \vdots \\ 79.0 & 0 \end{array}$$

At the morning of the launch the outside temperature was (uncharacteristically low for Florida) 31 degrees Fahrenheit. We wish to use logistic regression to compute the probability that the O-ring fails.

The model $p(x) = \sigma(\alpha+\beta x)$ is exactly a *single neuron* — the one you will train again in Section 7.9 — and we can fit it right here with the tools of this chapter, no library and no black box: maximize $\ell(\alpha,\beta)$ from Example 7.31 by gradient *ascent*, i.e., repeat

$$(\alpha,\beta) := (\alpha,\beta) + \lambda\,\nabla\ell(\alpha,\beta).$$

We walk *with* the gradient, since we are maximizing.

One preparation first — a step used everywhere in machine learning. We *standardize* the input:

$$z = \frac{x-\bar{x}}{s},$$

where

$$\bar{x} = \frac{x_1+\cdots+x_m}{m} \qquad \text{and} \qquad s^2 = \frac{(x_1-\bar{x})^2+\cdots+(x_m-\bar{x})^2}{m}$$

are the mean and the variance of the $m = 24$ temperatures (s is the standard deviation). The standardized inputs z have mean 0 and standard deviation 1 — small, comparable numbers around 0, no matter how the raw data are scaled. Why we bother will become clear in a moment, when we try skipping this step. We train the neuron $\sigma(a+bz)$ on the standardized input and translate back afterwards. Nothing is lost in the translation, because

$$a+bz = a+b\,\frac{x-\bar{x}}{s} = \left(a-\frac{b\bar{x}}{s}\right)+\frac{b}{s}x,$$

so the trained neuron $\sigma(a+bz)$ is the same function of the original temperature x as $\sigma(\alpha+\beta x)$ with

$$\beta = \frac{b}{s} \qquad \text{and} \qquad \alpha = a-\frac{b\bar{x}}{s}.$$

python cell — not displayed in the pdf version.

The mathematics was there before the launch: at 31 degrees Fahrenheit the trained neuron puts the probability of O-ring failure at 0.997.

So why did we standardize? Try the ascent directly on the raw temperatures and watch it fail, miserably — seeing *how* it fails teaches more than the success did. The cell below tries: 2000 steps with $\lambda = 0.1$, starting from $(\alpha,\beta) = (0,0)$. The two gradient lines in the loop are precisely $\nabla\ell$ from the exercise above — the only change from the successful cell is the missing standardization.

python cell — not displayed in the pdf version.

At a maximum the gradient is $(0,0)$. The cell reports the gradient $(6,381)$ — after 2000 steps the ascent is nowhere near the top. The culprit is the factor $x_i \approx 70$ in the second coordinate of the gradient. The very first step is

$$\beta := 0 + 0.1 \sum_{i=1}^{24} \left(E_i - \tfrac{1}{2}\right) x_i \approx -45.8,$$

which makes $\alpha + \beta x_i \approx -3200$. Then e^{3200} overflows, every probability $p(x_i)$ snaps to 0 or 1, and from there the ascent bounces back and forth without ever settling down. Shrinking the learning rate does not save us: with $\lambda = 0.00001$ the β-updates calm down, but now α — which has to climb above 11 — moves in steps of size about 10^{-5}. Rerun the cell with this λ and watch α barely leave 0; even two million steps would only push it to about 0.9. One fixed learning rate cannot serve two parameters living on such different scales — standardization put them on the same scale, and the ascent sailed home.

♠

(7.34) EXERCISE.

Professional libraries fit logistic regression in a few lines. Run the `scikit-learn` code below and compare α, β and the failure probability with the neuron trained in Example 7.33.

python cell — not displayed in the pdf version.

The option `solver='lbfgs'` chooses an algorithm for maximizing $\ell(\alpha,\beta)$ and `C=25` controls a socalled regularization. Try removing `C=25` first and then `solver='lbfgs'`. What happens?

LLM prompt — not displayed in the pdf version.

In the button below is a naive implementation of gradient ascent for the Challenger data set — a variant of the training in Example 7.33 that adjusts the step size with successive negative powers of 2, as in the introduction to this chapter, instead of standardizing the input. Run experiments with different initial values and number of iterations, and compare the quality of the solutions in terms of the gradient (which is available in the output from the Naive code).

Yes, you are allowed (and encouraged) to use generative AI tools here!

Naive code.

python cell — not displayed in the pdf version.

♠

7.9 Training neural networks

Every ingredient is now on the table: partial derivatives, the gradient, the chain rule on computational graphs, gradient descent (Section 7.5.1) and the activation functions — sigmoid and ReLU — from the very first chapter. In this section we put them to work and train genuine neural networks — first one so small that every number can be inspected by hand, then one that reads handwritten digits, and in the end one that *writes*. Everything runs live in these notes, and no mathematics beyond this chapter is involved.

7.9.1 A single neuron

The smallest neural network is a single *neuron* with one input: it takes a number x, forms $b+wx$ and applies the sigmoid function,

$$y = \sigma(b+wx) = \frac{1}{1+e^{-b-wx}}.$$

The numbers w and b are called the *weight* and the *bias*. They are the parameters the network is allowed to change when it learns. You have seen this function before: it is exactly the probability (7.16) of logistic regression — a single neuron *is* logistic regression.

Run the cell below to see how w and b bend and shift the sigmoid, then change the values and rerun.

python cell — not displayed in the pdf version.

7.9.2 Training a single neuron

Training means: choose w and b so that the neuron reproduces known data as well as possible. Suppose eight students report how many hours x_i they studied and whether they passed ($E_i = 1$) or failed ($E_i = 0$) an exam. (You met these eight students in Exercise 5.22, where the best straight line through the data predicted "probabilities" beyond 1 — here comes the better idea it promised.) A neuron with sigmoid activation *is* the logistic model of Section 7.8, so we already know what "as well as possible" should mean: maximize the log-likelihood. Machine learning prefers to minimize, so we flip the sign and call the result the *cross-entropy loss*: writing $y_i = \sigma(b+wx_i)$,

$$L(w,b) = -\sum_{i=1}^{8}\Big(E_i \ln y_i + (1-E_i)\ln(1-y_i)\Big),$$

which we minimize by gradient descent — the same computation as gradient ascent on ℓ, in new clothes. The gradient is a small miracle of cancellation. By the chain rule and the formula $\sigma'(z) = \sigma(z)(1-\sigma(z))$ from Exercise 7.30, the i-th term of L responds to the sum $z_i = b+wx_i$ at the rate

$$-\left(\frac{E_i}{y_i} - \frac{1-E_i}{1-y_i}\right) y_i(1-y_i) = y_i - E_i$$

— the derivative of the sigmoid cancels completely, leaving

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{8}(y_i - E_i)\,x_i \qquad \text{and} \qquad \frac{\partial L}{\partial b} = \sum_{i=1}^{8}(y_i - E_i).$$

Prediction too high: push down. Prediction too low: push up. Every data point pushes with a force equal to its error — few formulas in machine learning are used more often than this one. It is all a neural network library ever does, just for many more parameters. The training loop below is four lines.

python cell — not displayed in the pdf version.

The right hand plot is the famous *loss curve* — the first thing every machine learning engineer looks at during training. The left hand plot shows what the neuron has learned: a threshold around 2.5 hours of study. One thing to notice: because the data are perfectly separated at 2.5 hours, longer training keeps steepening the threshold — the weights grow forever, chasing probabilities 0 and 1 they can never quite reach. Real data are rarely this clean. Play with the data and watch the neuron adapt.

7.9.3 A genuine network: solving XOR

A single neuron has a hard limit. In Exercise 5.9 you showed that no line separates the four points $(0,0),(1,1)$ (labeled blue) and $(0,1),(1,0)$ (labeled red) — and a single neuron can only draw a line. This little data set, known as *XOR*, famously stalled neural network research for years.

The way out is to compose neurons into a network with a *hidden layer*: two neurons look at the input, and a third neuron looks at what they found,

$$\begin{aligned} h_1 &= \sigma(b_1 + w_{11}x_1 + w_{21}x_2) \\ h_2 &= \sigma(b_2 + w_{12}x_1 + w_{22}x_2) \\ y &= \sigma(c + v_1h_1 + v_2h_2). \end{aligned}$$

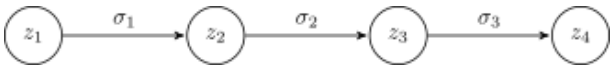
The composed function is no longer a bent line — and thanks to the chain rule we can still compute the gradient of the cross-entropy loss with respect to all nine parameters; the miraculous $y - E$ cancellation from the single neuron applies to the output neuron unchanged. The code below does exactly that in vectorized form: the *forward pass* evaluates the network, the *backward pass* walks the computational graph in reverse, exactly as in Section 7.7. This is backpropagation.

python cell — not displayed in the pdf version.

Look at the picture: the network has invented a diagonal band — a shape no single line can produce. Each hidden neuron contributes one line, and the output neuron combines them. Deep learning is this phenomenon repeated at scale.

(7.35) EXERCISE.

Backpropagation by hand. Consider the simple chain network



where

$$\begin{aligned} z_2 &= \sigma_1(z_1) = \sigma(a + bz_1) \\ z_3 &= \sigma_2(z_2) = \sigma(c + dz_2) \\ z_4 &= \sigma_3(z_3) = \sigma(e + fz_3), \end{aligned}$$

and σ is the sigmoid function. This neural network has input z_1 and output z_4. Let C be a function of the output z_4. For fixed z_1, we consider C as a function of a, b, c, d, e, f via

$$F\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = C(\sigma_3(\sigma_2(\sigma_1(z_1)))).$$

Backpropagation for training neural networks is using the chain rule for computing the gradient

$$\nabla F = \left(\frac{\partial F}{\partial a}, \frac{\partial F}{\partial b}, \frac{\partial F}{\partial c}, \frac{\partial F}{\partial d}, \frac{\partial F}{\partial e}, \frac{\partial F}{\partial f}\right).$$

Explain how to do this — it is exactly what the backward pass in the code above does. ♠

7.9.4 Reading handwritten digits

We end with real data: images of handwritten digits. The `sklearn` library ships 1797 small 8×8 gray scale images. Before training anything, look at the data — the cell below shows the first image in the collection together with the 8×8 matrix of gray values behind it.

python cell — not displayed in the pdf version.

An image is nothing but an 8×8 matrix of numbers. Flattening the matrix row by row turns it into a vector in $\mathbb{R}^{64}$ — and the machinery of this chapter applies. The network we now train is a composition

$$\mathbb{R}^{64} \longrightarrow \mathbb{R}^{32} \longrightarrow \mathbb{R}^{10},$$

exactly as promised in the very first chapter. Written out: an image, stored as a row vector $x \in \mathbb{R}^{64}$, is processed in two layers

$$\begin{aligned} h &= \mathrm{act}(xW_1 + b_1) \\ y &= \sigma(hW_2 + b_2), \end{aligned}$$

where W_1 is a 64×32 matrix of weights, $b_1 \in \mathbb{R}^{32}$ holds the biases of the 32 hidden neurons, W_2 is a 32×10 matrix and $b_2 \in \mathbb{R}^{10}$. Output number d of $y \in \mathbb{R}^{10}$ is a score between 0 and 1 for "this image shows the digit d" — ten separate yes/no neurons, so the scores need not sum to 1 — and the network's guess is the output with the largest score. Counting as in the first chapter, the network has

$$64 \cdot 32 + 32 + 32 \cdot 10 + 10 = 2410$$

parameters — all of them trained by gradient descent, in your browser.

The hidden activation function act is *your* choice in the code: the sigmoid or the ReLU from the first chapter. The output layer always uses the sigmoid, so that its ten scores land between 0 and 1. Backpropagation only needs the derivative of the activation: $\sigma'(z) = \sigma(z)(1-\sigma(z))$ for the sigmoid, while the ReLU has derivative 1 for $z > 0$ and 0 for $z < 0$ — this is the function `dact` in the code.

Before you run it, read the code with this dictionary. The 1500 training images are stacked as the rows of the matrix `Xtr`, so `Xtr @ W1 + b1` computes all 1500 hidden vectors in one matrix multiplication. The line `T = np.eye(10)[labels]` turns the digit d into the target vector with 1 in entry d and 0 elsewhere (one-hot encoding — the same trick as for words in Chapter 5). The loss is the cross-entropy, summed over the ten outputs — exactly as for the single neuron, ten times.

So how does the code find all 2410 partial derivatives? This is *backpropagation*, and it is nothing but the chain rule of Section 7.7 — the nudge bookkeeping of Section 7.6 — organized cleverly. Let us walk through it for a single image x with target vector t. Write the two layers with their intermediate sums,

$$\begin{aligned} z^{(1)} &= xW_1 + b_1, & h &= \mathrm{act}(z^{(1)}), \\ z^{(2)} &= hW_2 + b_2, & y &= \sigma(z^{(2)}), \end{aligned}$$

and let

$$L = -\sum_{d=1}^{10} \Big(t_d \ln y_d + (1-t_d)\ln(1-y_d) \Big)$$

be the cross-entropy loss.

Start at the loss and move backwards. How much does L respond to a nudge of the sum $z_d^{(2)}$ entering output neuron d? The nudge passes through the sigmoid and into the d-th term of the loss — and the single-neuron cancellation strikes again:

$$\delta_d^{(2)} = \frac{\partial L}{\partial z_d^{(2)}} = y_d - t_d.$$

This is the line `dy` in the code. With $\delta^{(2)}$ in hand, the gradients of the output layer's parameters are immediate: the weight $(W_2)_{jd}$ enters the sum $z_d^{(2)}$ multiplied by h_j, so

$$\frac{\partial L}{\partial (W_2)_{jd}} = h_j \, \delta_d^{(2)} \qquad \text{and} \qquad \frac{\partial L}{\partial (b_2)_d} = \delta_d^{(2)}$$

— the lines `dW2` and `db2`.

Push the nudge one layer further back. The hidden value h_j influences the loss through *all ten* outputs, one path per output neuron, so its exchange rate is a sum over the ten paths — exactly as in the computational graphs of Section 7.7.1:

$$\frac{\partial L}{\partial h_j} = \sum_{d=1}^{10} \delta_d^{(2)} \, (W_2)_{jd}.$$

Passing the nudge through the activation of hidden neuron j then gives

$$\delta_j^{(1)} = \frac{\partial L}{\partial z_j^{(1)}} = \left(\sum_{d=1}^{10} \delta_d^{(2)} \, (W_2)_{jd} \right) \mathrm{act}'\left(z_j^{(1)}\right)$$

— the line `dh`, where the sum over d is the matrix product `dy @ W2.T` and act$'$ is the function `dact`. The first layer's gradients now follow exactly as before:

$$\frac{\partial L}{\partial (W_1)_{ij}} = x_i \, \delta_j^{(1)} \qquad \text{and} \qquad \frac{\partial L}{\partial (b_1)_j} = \delta_j^{(1)}$$

— the lines `dW1` and `db1`.

Notice the economy of the scheme. One forward pass stores h and y; one backward pass computes $\delta^{(2)}$ and then $\delta^{(1)}$; and out come all 2410 partial derivatives. The price of the whole gradient is roughly *two* evaluations of the network, no matter how many parameters there are — this is why the loss nudges are propagated *backwards*, layer by layer, and why the method is called backpropagation. Finally, the code does all of this for the 1500 training images at once: stacking the images as rows turns the products $h_j \, \delta_d^{(2)}$ into the matrix product `h.T @ dy`, which sums over the images automatically, and the division by `len(Xtr)` makes L the mean loss over the data. Training takes a few seconds.

python cell — not displayed in the pdf version.

Now switch `activation` to `"relu"` and rerun. Which activation reads more of the unseen images correctly here? Notice that the ReLU asks for a smaller learning rate — its unbounded output makes big steps dangerous.

Around nine out of ten unseen images are read correctly — by a network you trained from scratch and whose every line of mathematics you have now met. Scaling this recipe up *is* modern AI: more pixels, more layers, and libraries such as `pytorch` that apply the chain rule automatically to computational graphs with billions of nodes — exactly the graphs of Section 7.7. Such libraries do not run inside a web page, but you can use them for free in Google Colab.

7.9.5 A network that writes

The networks above *read*: they take an image and answer a question about it. But the neural network you use every day — the chatbot from Section 1.1 — *writes*. That looks like a completely different ability. It is not. Writing is reading in disguise: a network writes by answering, over and over again, the classification question

"given the text so far, which symbol comes next?"

picking a symbol according to the answer, appending it to the text and asking again. In this final section you will train a genuine writing network. A chatbot learns from trillions of words; we scale the idea down to a miniature that learns from a couple of hundred Danish first names — and then invents new ones.

Here is the plan. Our alphabet consists of the letters that occur in the names plus a special symbol "." marking the end of a name — 27 symbols in all. The network gets a *context* of $C = 3$ consecutive symbols and must predict the symbol that follows. Every name in the list becomes a handful of training examples; padding with dots supplies the start, and the final dot teaches the network how names *end*. The name `anna` alone contributes five:

$$\text{"..."} \mapsto \text{"a"}, \quad \text{"..a"} \mapsto \text{"n"}, \quad \text{".an"} \mapsto \text{"n"}, \quad \text{"ann"} \mapsto \text{"a"}, \quad \text{"nna"} \mapsto \text{"."}$$

Symbols become vectors by one-hot encoding, exactly as in (5.17): each of the three context symbols becomes a one-hot vector in $\mathbb{R}^{27}$, and gluing the three together gives an input vector $x \in \mathbb{R}^{81}$. The target is one of 27 classes. Structurally we are back at the digits network — reading handwritten digits was "which of 10 classes?", writing names is "which of 27 classes?", asked repeatedly.

One ingredient needs an upgrade. For writing, the 27 output numbers must form a *probability distribution*: they must be positive and sum to 1, because we are going to roll a die weighted by them. The digits network used ten separate sigmoids, which obey no such constraint. The standard fix is the softmax function, which turns any vector $z \in \mathbb{R}^V$ of scores into probabilities:

$$\operatorname{softmax}(z)_j = \frac{e^{z_j}}{e^{z_1} + \cdots + e^{z_V}}. \tag{7.19}$$

The exponentials make every output positive, and dividing by their sum makes the outputs sum to 1. You have met this recipe before: it is exactly (5.18), which computed the attention weights in Section 5.5.3 — there the outputs weighted a blend of word vectors, here they are read as probabilities. Softmax is also the many-class face of the sigmoid (see Exercise 7.36 below).

As loss we keep cross-entropy in the form maximum likelihood handed us in (7.18): if the correct next symbol has class t and the network outputs the probabilities $y = \operatorname{softmax}(z)$, the loss of the example is

$$L = -\ln(y_t),$$

the price for the probability the network gave to what actually came next. And now the small miracle of Section 7.9 repeats itself: the nudge of the loss with respect to the scores z is again *prediction minus target*,

$$\frac{\partial L}{\partial z_j} = y_j - t_j,$$

where $t \in \mathbb{R}^{27}$ is the one-hot target. This is the same formula that ran the single neuron, the XOR network and the digits network — so the backward pass below is, line for line, the code you have already studied.

> **Why the cancellation survives.** Insert $y_t = e^{z_t}/(e^{z_1} + \cdots + e^{z_V})$ into the loss and use $\ln(a/b) = \ln(a) - \ln(b)$:
>
> $$L = -\ln(y_t) = -z_t + \ln\left(e^{z_1} + \cdots + e^{z_V}\right).$$
>
> Now differentiate with respect to z_j. The first term contributes -1 if $j = t$ and 0 otherwise — that is exactly $-t_j$. For the second term the chain rule gives
>
> $$\frac{\partial}{\partial z_j} \ln\left(e^{z_1} + \cdots + e^{z_V}\right) = \frac{e^{z_j}}{e^{z_1} + \cdots + e^{z_V}} = y_j,$$
>
> and adding the two contributions gives $\dfrac{\partial L}{\partial z_j} = y_j - t_j$. Compare this with the computation for the single neuron in Section 7.9 — same cancellation, same reason: the derivative of ln undoes the exponential.

So the network is

$$\mathbb{R}^{81} \longrightarrow \mathbb{R}^{64} \longrightarrow \mathbb{R}^{27},$$

a sigmoid hidden layer followed by a softmax output layer — counting as in Exercise 1.126, that is $81 \cdot 64 + 64 + 64 \cdot 27 + 27 = 7003$ parameters. Before training these are small random numbers, so the network spreads

its probability almost evenly over the 27 symbols and the mean loss starts near $-\ln(1/27) = \ln(27) \approx 3.3$: the loss of pure guessing. Watch it fall.

After training comes the payoff: *writing*. Start from the empty context ". . .", let the network output its 27 probabilities, roll a die weighted by them, append the resulting symbol to the context and repeat — until the die shows ".": the network has decided that its name is finished. This is composition one last time: the network fed its own output, over and over. The cell below does all of it — the data, the training and twenty rolls of the die. Training takes a little longer than the digits network; the names are worth the wait.

python cell — not displayed in the pdf version.

Look at the output. Some inventions are flagged: the network has simply memorized a name from the list. But most are new, and they do not look like random letters — they look *Danish*. From nothing but gradient descent on cross-entropy, the network has learned the statistics of Danish names: which letters like to follow which, how names tend to begin, and when they should stop. A few outputs are honest garbage; a window of three symbols only carries so much memory. The balance on display here — between memorizing the training data and learning its shape — is the overfitting story of Chapter 2 (Figure 2.41) in new clothes.

Now zoom out. Replace the 27 symbols by roughly 100 000 word fragments (*tokens*), the context of 3 symbols by a context of many thousands of tokens, the two layers by around a hundred, the 7003 parameters by hundreds of billions, and the list of names by a large chunk of everything ever written — and you have a modern large language model. When the chatbot from Section 1.1 answers you, it does *literally* what the cell above does: compute probabilities for the next token with softmax, roll the die, append, repeat. There is one genuinely new architectural idea in the scaled-up version — *attention*, layers that let far-apart parts of the context exchange information, built on the dot products of Section 5.3.2 — but no new kind of mathematics: matrices, activation functions, cross-entropy, the chain rule and gradient descent, all of which you now own.

(7.36) EXERCISE.

Softmax with $V = 2$ classes *is* the sigmoid: show that

$$\text{softmax}(z_1, z_2)_1 = \sigma(z_1 - z_2),$$

where $\sigma(u) = 1/(1+e^{-u})$. So a softmax output layer with two classes is a single sigmoid neuron reading the difference of the two scores — logistic regression once more.

Hint. Divide the numerator and the denominator of

$$\frac{e^{z_1}}{e^{z_1} + e^{z_2}}$$

by e^{z_1}.

♠

(7.37) EXERCISE.

The cell prints the mean loss before training and after training.

1. Explain why the loss starts close to $\ln(27) \approx 3.3$.
2. After training the mean loss is close to 1.1. Compute $e^{-1.1}$. In what sense does the network now give the correct next symbol "a third of the probability" — and how much better is that than the $1/27 \approx 4\%$ it started from?

Hint. Before training the weights are small random numbers, so all 27 probabilities are close to $1/27$ and every example costs about $-\ln(1/27) = \ln(27)$. For the second part: if every example had loss exactly ℓ, the probability given to the correct symbol would be $e^{-\ell}$ every time.

♠

(7.38) EXERCISE.

The cell is yours — experiment.

1. Set the context length to $C = 1$, so the network only sees one symbol back, and rerun. The invented names get worse in a specific way — describe it. What statistics can the network still learn?
2. Try $C = 5$. More of the output gets flagged as memorized — why is a longer context *more* prone to memorizing a small dataset?
3. Change the die roll `np.random.default_rng(1)` to a different seed and rerun the last part — new names, same network. Then replace the names by a word list of your own choice (capital cities, Pokémon, dinosaurs, ...) and train a different writer.

♠

7.10 Lagrange multipliers

The method of Lagrange multipliers is a super classical way of solving optimization problems with non-linear (equality) constraints. We will only consider the special case

$$\max/\min_{g(x_1,\dots,x_d)=0} \quad f(x_1,\dots,x_d), \tag{7.20}$$

where both $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ are differentiable functions.

There is a very useful trick for attacking (7.20). One introduces an extra variable λ (a Lagrange multiplier) and the Lagrangian function $L : \mathbb{R}^{d+1} \to \mathbb{R}$ given by

$$L(x_1,\dots,x_d,\lambda) = f(x_1,\dots,x_d) + \lambda g(x_1,\dots,x_d).$$

The main result is the following.

(7.39) THEOREM.

Suppose that $(z_1,\dots,z_d)$ is a local maximum/minimum for (7.20). Then there exists $\lambda \in \mathbb{R}$, such that $(z_1,\dots,z_d,\lambda)$ is a critical point for L.

So to solve (7.20) we simply (well, this is not always so simple) look for critical points for L. This amounts to solving the $d+1$ (non-linear) equations coming from $\nabla L = 0$ i.e.,

$$g(x_1,\dots,x_d) = 0 \tag{7.21}$$

$$\frac{\partial f}{\partial x_1}(x_1,\dots,x_d) + \lambda \frac{\partial g}{\partial x_1}(x_1,\dots,x_d) = 0 \tag{7.22}$$

$$\vdots \tag{7.23}$$

$$\frac{\partial f}{\partial x_d}(x_1,\dots,x_d) + \lambda \frac{\partial g}{\partial x_d}(x_1,\dots,x_d) = 0 \tag{7.24}$$

For $d = 2$ we can quickly give a sketch of the idea behind the proof. The (difficult) fact is that we may find a differentiable function $x(t)$ in one variable t, such that

$$g(t, x(t)) = 0$$

and the local minimum has the form $v_0 = (t_0, x(t_0))$.

Once we have this, the chain rule does its magic. We consider the one variable functions

$$F(t) = f(t, x(t)) \tag{7.25}$$
$$G(t) = g(t, x(t)) \tag{7.26}$$

For both of these we have $F'(t_0) = G'(t_0) = 0$ (why?). The chain rule now gives a non-zero vector orthogonal to $\nabla f(v_0)$ and $\nabla g(v_0)$. This is only possible if they are parallel as vectors i. e. , there exists λ, such that

$$\nabla f(v_0) = \lambda \nabla g(v_0).$$

(7.40) EXAMPLE.

Consider the minimization problem

$$\min_{x^2+y^2=1} \quad x + y.$$

First of all, why does this problem have a solution at all? We write the non-linear equations

$$1 + 2x\lambda = 0$$
$$1 + 2y\lambda = 0$$
$$x^2 + y^2 - 1 = 0$$

up coming from the critical points of the Lagrange function. Now we know that these can be solved and that amongst our solutions there is a minimum! ♠

(7.41) EXERCISE.

Computing the distance from the line $y = x + 1$ to the point $(1, 1)$ gives rise to the minimization problem

$$\min_{y=x+1} \quad (x-1)^2 + (y-1)^2.$$

Solve this minimization problem using Theorem 7.39. ♠

(7.42) EXERCISE.

Use Theorem 7.39 to maximize $x^2 + y^2$ subject to $x^2 + xy + y^2 = 4$.

Hint. Here you end up with the system

$$(2\lambda + 2)x + \lambda y = 0$$
$$\lambda x + (2\lambda + 2)y = 0$$

of linear equations in x and y, where you regard λ as a constant. Use Gaussian elimination to solve this system in order to derive a (nice) quadratic equation in λ coming from

$$-\frac{\lambda}{2\lambda + 2}\lambda y + (2\lambda + 2)y = 0,$$

where you assume that $y \neq 0$. Handle the case $y = 0$ separately.

Consider the subset $C = \{(x,y) \in \mathbb{R}^2 \mid x^2+xy+y^2 = 4\}$. Why is C a closed subset? Why is C bounded?

Hint. To prove that C is bounded you can keep y fixed in

$$x^2+yx+y^2-4=0 \tag{7.27}$$

and solve for x. A last resort is using the plot in the Hint button below, but that does not give any real insight unless you explain how the plot is made from the equation (7.27).

How does this relate to Theorem 5.84?

Does the optimization problem have a geometric interpretation?

Hint.

python cell — not displayed in the pdf version.

♠

(7.43) EXERCISE.

A rectangular box has side lengths x, y and z. What is its maximal volume when we assume that (x,y,z) lies on the plane

$$\frac{x}{a}+\frac{y}{b}+\frac{z}{c}=1$$

for $a,b,c>0$. ♠

(7.44) EXERCISE.

A company is planning to produce a box with volume 2 m^3. For design reasons it needs different materials for the sides, top and bottom. The cost of the materials per square meter is 1 dollar for the sides, 1.5 dollars for the bottom and the top. Find the measurements of the box minimizing the production costs.

Hint. Let x,y and z be the measurements. Use $xyz=2$ to rewrite the Lagrange equations so that y and z are expressed in terms of x.

♠

(7.45) EXERCISE.

$$\max_{\substack{p_1+\cdots+p_n=1\\ p_1>0,\ldots,p_n>0}} \quad -p_1\log_2(p_1)-\cdots-p_n\log_2(p_n).$$

The sum

$$H(p_1,\ldots,p_n) = -p_1\log_2(p_1)-\cdots-p_n\log_2(p_n)$$

is called the (Shannon) entropy of the discrete probability distribution $p_1,\ldots,p_n$. One may use Jensen's inequality applied to the convex function $-\log_2(x)$ to prove that

$$H(p_1,\ldots,p_n) \leq \log_2(n).$$

♠

7.11 Optimization using the interior and boundary of a subset

Suppose that $C \subseteq \mathbb{R}^d$ is a closed subset and $f : \mathbb{R}^d \to \mathbb{R}$ is a continuous function. Recall (see Theorem 5.84) that the optimization problem

$$\max/\min_{v \in C} f(v)$$

always has a solution if C in addition to being closed is also bounded. To solve such an optimization problem, it often pays to decompose C as

$$C = \partial C \cup C^o,$$

where ∂C is the boundary of C (recall Definition 5.56) and C^o the interior of C (recall Definition 5.58). The strategy is then to look for an optimal solution both in ∂C and C^o and then compare these. In some sense we are making a "recursive" call to a lower dimensional optimization problem for the boundary ∂C. This is illustrated by the basic example: $f(x) = x^2 - 5x + 6$ and $C = [0,4]$. Here $\partial C = \{0,4\}$ and $C^o = (0,4)$. Notice that ∂C is finite here.

If v_0 is an element of C^o, then there exists an open subset $U \subseteq C$, such that $v_0 \in U$. Therefore the following proposition holds, when you take Proposition 7.20 into account.

(7.46) PROPOSITION.

Consider an optimization problem

$$\max/\min_{v \in C} f(v), \tag{7.28}$$

where $C \subseteq \mathbb{R}^d$ is a subset, $f : \mathbb{R}^d \to \mathbb{R}$ a differentiable function and v_0 an optimal solution to (7.28). If $v_0 \in C^o$, then v_0 is a critical point of f.

Basically, to solve an optimization problem like (7.28) one needs to consider the boundary and interior as separate cases. For points on the boundary we cannot use the critical point test in Proposition 7.20. This test only applies to the interior points.

Usually the boundary cases are of smaller dimension and easier to handle as illustrated in the example below.

(7.47) EXAMPLE.

Consider the minimization problem

$$\min_{x^2+y^2=1} \quad x+y$$

from Example 7.40. Let us modify it to

$$\min_{(x,y) \in C} \quad x+y, \tag{7.29}$$

where

$$C = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}.$$

We are now minimizing not only over the unit circle, but the whole unit disk. Here

$$\partial C = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\} \quad \text{and} \quad C^o = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1\}.$$

Proposition 7.46 guides us first to look for optimal points in C^o. Here we use Proposition 7.20 to show that there can be no optimal points in C^o, because the gradient of the function $f(x,y) = x + y$ is

$$\nabla f = (1,1).$$

Therefore the boundary needs to be analyzed and the usual technique (as was implicit in Lagrange multipliers) is to find a parametrization for the points (x,y) satisfying $x^2+y^2=1$. There are two of those (one for the upper unit circle and one for the lower unit circle):

$$\left(t,\sqrt{1-t^2}\right)$$
$$\left(t,-\sqrt{1-t^2}\right),$$

where $t\in[-1,1]$. This means that the optimization problem for the boundary ∂C turns into the two simpler optimization problems of minimizing

$$t+\sqrt{1-t^2} \qquad \text{and} \qquad t-\sqrt{1-t^2}$$

subject to $t\in[-1,1]$. These can as one variable optimization problems be solved the usual way. ♠

The exercises below are taken from an older Calculus course at Aarhus.

(7.48) EXERCISE.

Solve the two optimization problems

$$\max/\min_{(x,y)\in C} x^2-2xy+2y,$$

where $C=\{(x,y)\in\mathbb{R}^2 \mid 0\le x\le 3, 0\le y\le 2\}$. But first give a reason as to why they both are solvable.

> **Hint.** First find ∂C and C^o. Then try with Proposition 7.46 supposing that a maximal point really is to be found in C^o and not on ∂C.

♠

(7.49) EXERCISE.

Solve the two optimization problems

$$\max/\min_{(x,y)\in C} 1+4x-5y,$$

where $C=\{(x,y)\in\mathbb{R}^2 \mid 0\le x, 0\le y, 3x+2y\le 6\}$. But first give a reason as to why they both are solvable. ♠

(7.50) EXERCISE.

Solve the two optimization problems

$$\max/\min_{(x,y)\in C} 3+xy-x-2y,$$

where C is the triangle with vertices in $(1,0),(5,0)$ and $(1,4)$. But first give a reason as to why they both are solvable. ♠

(7.51) EXERCISE.

Use Proposition 7.46 to give all the minute details in applying Theorem 7.39 to solve Exercise 7.44.

Hint: First rewrite to the problem, where you minimize $6/y + 4/x + 2xy$ subject to $x > 0, y > 0$ by using $xyz = 2$. Then explain why this problem may be solved by restricting with upper and lower bounds on x and y. The minimum $(6\sqrt[3]{6})$ is attained in a critical point and not on the boundary. For $N \in \mathbb{N} \setminus \{0\}$ one may optimize over the compact subset

$$C_N = \{(x,y) \mid \frac{1}{N} \leq x \leq N, \frac{1}{N} \leq y \leq N\}$$

and analyze what happens when $N \to \infty$. ♠

8 THE HESSIAN

In Chapter 6 we exploited the second derivative $f''(x)$ of a one variable real function $f : (a,b) \to \mathbb{R}$ to analyze convexity along with local minima and maxima.

In this chapter we introduce an analogue of the second derivative for real functions $f : \mathbb{R}^d \to \mathbb{R}$ of several variables. This will be a $d \times d$ matrix. The important notion of a matrix being positive (semi-) definite introduced in Section 3.7 will now make its appearance.

8.1 Introduction

In Section 6.4 the Taylor expansion for a one variable differentiable function $f : \mathbb{R} \to \mathbb{R}$ centered at x_0 with step size $h = x - x_0$ was introduced as

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2!}f''(x_0)h^2 + \cdots \tag{8.1}$$

Recall that the second derivative $f''(x_0)$ contains a wealth of information about the function. Especially if $f'(x_0) = 0$, then we might glean from $f''(x_0)$ if x_0 is a local maximum or minimum or none of these (see Theorem 6.50 and review Exercise 6.53).

We also saw — remember the zigzag path across the narrow valley in Section 7.5.1 — that plain gradient descent can be painfully slow. Taking the second derivative into account gives a more detailed picture of the function and, as in Newton's method below, much faster algorithms.

8.2 Several variables

Our main character is a differentiable function $F : \mathbb{R}^d \to \mathbb{R}$ in several variables. We already know that

$$F(v_0 + h) = F(v_0) + \nabla F(v_0)h + \varepsilon(h)\,|h|\,,$$

where v_0 and h are vectors in $\mathbb{R}^d$ (as opposed to the good old numbers in (8.1)). Take a look back at Definition 7.3 for the general definition of differentiability.

We wish to have an analogue of the Taylor expansion in (8.1) for such a function of several variables. To this end we introduce the function $g : \mathbb{R} \to \mathbb{R}$ given by

$$g(t) = F(v_0 + th). \tag{8.2}$$

Notice that

$$g(t) = (F \circ A)(t),$$

where $A : \mathbb{R} \to \mathbb{R}^d$ is the function given by $A(t) = v_0 + th$. In particular we get

$$g'(t) = F'(v_0 + th)h = \nabla F(v_0 + th)h \tag{8.3}$$

by using the chain rule (see Theorem 7.23).

(8.1) EXERCISE.

Explain how the chain rule is applied to get (8.3). ♠

The derivative $g'(t)$ is also composed of several functions and again we may compute $g''(t)$ by using the chain rule:

$$\begin{aligned} g''(t) &= (C \circ B \circ A)'(t) \\ &= (C \circ B)'(A(t))A'(t) \\ &= C'(B(A(t)))B'(A(t))A'(t), \end{aligned}$$

where $B : \mathbb{R}^d \to \mathbb{R}^d$ is defined by

$$B(v) = \nabla F(v)^\top$$

and $C : \mathbb{R}^d \to \mathbb{R}$ by

$$C(v) = v^\top h.$$

(8.2) DEFINITION.

The Hessian matrix *of F at the point $v \in \mathbb{R}^d$ is defined by*

$$\nabla^2 F(v) := \begin{pmatrix} \frac{\partial^2 F}{\partial x_1 \partial x_1}(v) & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_d}(v) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_d \partial x_1}(v) & \cdots & \frac{\partial^2 F}{\partial x_d \partial x_d}(v) \end{pmatrix}.$$

A very important observation is that $\nabla^2 F(v)$ above is a symmetric matrix if F satisfies the condition in the last part of Theorem 7.11.

(8.3) EXAMPLE.

Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$f(x,y) = \sin(xy) + x^2 y^2 + y.$$

Then the gradient

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

and the Hessian

$$\nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

of f are computed in the python cell below.

python cell — not displayed in the pdf version.

See the calculus tutorial in the `sympy` documentation for more. ♠

(8.4) EXERCISE.

Verify (just this once) by hand the computations done by `sympy` in Example 8.3.

Also, experiment with a few other functions in the python cell and compute their Hessians. ♠

By applying Proposition 7.9 it is not too hard to see that the Hessian matrix fits nicely into the framework above, since

$$B'(v) = \nabla^2 F(v). \tag{8.4}$$

The full application of the chain rule then gives

$$g''(t) = h^\top \nabla^2 F(v_0 + th)h. \tag{8.5}$$

(8.5) EXERCISE.

Give a detailed explanation as to why (8.4) holds. ♠

8.3 Newton's method for finding critical points

We may use Newton's method for computing critical points for a function $F : \mathbb{R}^d \to \mathbb{R}$ of several variables. Recall that a critical point is a point $v_0 \in \mathbb{R}^d$ with $\nabla F(v_0) = 0$. By (7.9) and (8.4) the computation in Newton's method becomes

$$v_1 = v_0 - \left(\nabla^2 F(v_0)\right)^{-1} \nabla F(v_0). \tag{8.6}$$

In practice the (inverse) Hessian appearing in (8.6) is often a heavy computational burden. This leads to the socalled quasi-Newton methods, where the inverse Hessian in (8.6) is replaced by other matrices.

(8.6) EXAMPLE.

Let us watch (8.6) in action on the function from Exercise 7.21,

$$f(x,y) = x^3 + xy + y^3,$$

with gradient and Hessian

$$\nabla f(x,y) = \begin{pmatrix} 3x^2 + y \\ 3y^2 + x \end{pmatrix} \qquad \text{and} \qquad \nabla^2 f(x,y) = \begin{pmatrix} 6x & 1 \\ 1 & 6y \end{pmatrix}.$$

This function offers a rare luxury: the equation $\nabla f(x,y) = (0,0)$ can be solved by hand. The first coordinate gives $y = -3x^2$, and substituting this into the second,

$$3\left(-3x^2\right)^2 + x = x\left(27x^3 + 1\right) = 0,$$

so $x = 0$ or $x = -\frac{1}{3}$. The two critical points are

$$(0,0) \qquad \text{and} \qquad \left(-\tfrac{1}{3}, -\tfrac{1}{3}\right).$$

Since we know the answers in advance, we can concentrate on *how* Newton's method finds them. The cell below starts at `v0` and prints the point and the length of the gradient after every step of (8.6).

python cell — not displayed in the pdf version.

Two things in the output are worth savoring. First, the speed. Starting from $(1,1)$ the iteration converges to $(0,0)$, and the lengths of the gradients fall like

$$10^{-2}, \quad 10^{-3}, \quad 10^{-5}, \quad 10^{-9}, \quad 10^{-17}.$$

The exponent roughly doubles in every step, so the number of correct digits doubles too. This *quadratic convergence* is the trademark of Newton's method — compare with the 2000 small steps gradient ascent needed in Example 7.33.

Second, the destination depends on the start. From $(-1,-1)$ the very same iteration converges to the other critical point $(-\frac{1}{3},-\frac{1}{3})$. Newton's method solves $\nabla f(x,y)=(0,0)$ and nothing else; it has no opinion on whether the point it finds is a local minimum, a local maximum or neither. So which of the three did we just find? The answer needs the main character of this chapter, the Hessian: see Example 8.14 and Exercise 8.18. ♠

(8.7) EXERCISE.

Run the cell in Example 8.6 with the initial point `v0 = [1/6, 1/6]`. What goes wrong?

Show that $\nabla^2 f(x,y)$ is invertible if and only if $36xy \neq 1$, so the failing initial points lie exactly on a hyperbola in the plane. If you nudge the initial point to `v0 = [1/6 + 0.001, 1/6]`, which critical point does Newton's method find? ♠

8.4 The Hessian and critical points

Now we are in a position to state at least the first terms in the Taylor expansion for a differentiable function $F:\mathbb{R}^d\to\mathbb{R}$. The angle of the proof is to reduce to the one-dimensional case through the function $g(t)$ defined in (8.2). Here one may prove that

$$g(t)=g(0)+g'(0)t+\frac{1}{2}g''(0)t^2+\varepsilon(t)t^2, \tag{8.7}$$

where $\varepsilon(0)=0$ with ε continuous at 0, much like in the definition of differentiability except that we also include the second derivative.

Now (8.7) translates into

$$F(v_0+th)=F(v_0)+(\nabla F(v_0)h)\,t+\frac{1}{2}\left(h^\top\nabla^2F(v_0)h\right)t^2+\varepsilon(t)t^2 \tag{8.8}$$

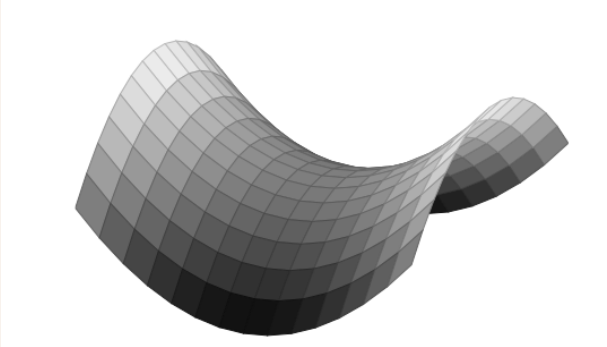
by using (8.3) and (8.5).

(8.8) DEFINITION.

A critical point v_0 is called a saddle point *for F if there exists two vectors $u, v \in \mathbb{R}^d$, such that*

$$\begin{aligned} t = 0 \quad & \textit{is a local minimum for the function} \quad F(v_0 + tu) \\ t = 0 \quad & \textit{is a local maximum for the function} \quad F(v_0 + tv) \end{aligned}$$

as illustrated in the graphics below.



Now go back and recall the definition of positive definite matrices in Section 3.7. We call a symmetric A matrix *negative definite* if $-A$ is positive definite. One more concept (related to the definition of saddle point above):

(8.9) DEFINITION.

A symmetric $d \times d$ matrix A is called indefinite *if there exists $u, v \in \mathbb{R}^d$ with*

$$\begin{aligned} u^\top A u > 0 \quad & \textit{and} \\ v^\top A v < 0. & \end{aligned}$$

So an indefinite matrix is mixed up in the sense that it is neither positive definite, nor negative definite.

(8.10) EXAMPLE.

$$\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \quad \text{is positive definite}$$

$$\begin{pmatrix} -2 & 0 \\ 0 & -3 \end{pmatrix} \quad \text{is negative definite}$$

$$\begin{pmatrix} 2 & 0 \\ 0 & -3 \end{pmatrix} \quad \text{is indefinite}$$

$$\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{is neither positive definite, negative definite, nor indefinite}$$

♠

We have the following addition to Proposition 3.43 (with a similar proof).

(8.11) PROPOSITION.

Let A be a symmetric $d \times d$ matrix and B an invertible $d \times d$ matrix. Then A is indefinite (positive definite, negative definite) if and only if

$$B^\top AB$$

is indefinite (positive definite, negative definite).

From (8.8) one can prove the following nice criterion, which may be viewed as a several variable generalization of Theorem 6.50.

(8.12) THEOREM.

Let v_0 be a critical point for $F : \mathbb{R}^d \to \mathbb{R}$. Then

(i) v_0 is a local minimum if $\nabla^2 F(v_0)$ is positive definite.

(ii) v_0 is a local maximum if $\nabla^2 F(v_0)$ is negative definite.

(iii) v_0 is a saddle point if $\nabla^2 F(v_0)$ is indefinite.

(8.13) QUIZ.

quiz — not displayed in the pdf version.

♠

(8.14) EXAMPLE.

Consider, with our new technology in Theorem 8.12, Exercise 7.21 once again. Here we analyzed the point $v_0 = (0,0)$ for the function

$$f(x,y) = x^3 + xy + y^3$$

and showed (by a trick) that v_0 is neither a local maximum nor a local minimum for f. The Hessian matrix for $f(x,y)$ at v_0 is

$$H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Now Theorem 8.12(iii) shows that v_0 is a saddle point, since

$$\begin{pmatrix} x & y \end{pmatrix} H \begin{pmatrix} x \\ y \end{pmatrix} = 2xy$$

and

$$u^\top Hu > 0 \qquad \text{for } u = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$v^\top Hv < 0 \qquad \text{for } v = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

python cell — not displayed in the pdf version.

♠

(8.15) EXERCISE.

Try plotting the graph for different values of `a`[1] in the python cell in Example 8.14. What do you observe for the point v_0 with respect to the function? Does `a` have to be a number? Could it be an expression in the variables `x` and `y` like `a = -10*np.cos(x)*np.sin(y)`? ♠

(8.16) EXERCISE.

Check the computation of the Hessian matrix H in Example 8.14 by showing that the Hessian matrix for f at the point (x,y) is

$$\begin{pmatrix} 6x & 1 \\ 1 & 6y \end{pmatrix}.$$

♠

(8.17) EXERCISE.

What about u and v in Example 8.14? How do they relate to the hint given in Exercise 7.21? ♠

(8.18) EXERCISE.

Newton's method in Example 8.6 also found the critical point $\left(-\frac{1}{3},-\frac{1}{3}\right)$ for $f(x,y) = x^3+xy+y^3$. Compute the Hessian of f at this point and use Theorem 8.12 to decide whether it is a local minimum, a local maximum or a saddle point. ♠

(8.19) EXERCISE.

Give an example of a function $F : \mathbb{R}^2 \to \mathbb{R}$ having a local minimum at x_0, where $\nabla^2 F(x_0)$ is not positive definite. ♠

(8.20) EXERCISE.

(i) The point $\left(0, \frac{\sqrt{3}}{3}\right)$ is a critical point for

$$f(x,y) = x^3+y^3-y.$$

What does Theorem 8.12 say about this point?

(ii) The point $\left(\frac{1}{3}, \frac{1}{3}\right)$ is a critical point for

$$f(x,y) = -x^3-x^2+x-y^3+2y^2-y.$$

What does Theorem 8.12 say about this point?

[1] `a=4` shows the saddle point clearly.

(iii) The point $(0,1)$ is a critical point for

$$f(x,y) = x^3 - x^2 + y^3 - y^2 - y.$$

What does Theorem 8.12 say about this point?

♠

(8.21) EXERCISE.

Consider the function

$$f(x,y) = x^4y^2 + x^2y^4 - 3x^2y^2.$$

Compute its critical points and decide on their types according to Theorem 8.12. Try to convince yourself that

$$f(x,y) \geq -1$$

for every $x, y \in \mathbb{R}$.

python cell — not displayed in the pdf version.

Hint:

Look at the minimization problem

$$\min f(x,y)$$

subject to

$$(x,y) \in C = \{(x,y) \mid -M \leq x \leq M,\ -M \leq y \leq M\},$$

where M is a big number. ♠

(8.22) EXERCISE.

Give an example of a function $f : \mathbb{R} \to \mathbb{R}$ that has a local maximum, but where there exists $x \in \mathbb{R}$ with $f(x) > M$ for any given (large) number M. ♠

8.5 Differentiable convex functions of several variables

Below is the generalization of Theorem 6.61 to several variables. You have already seen this in Exercise 6.62, right?

(8.23) THEOREM.

Let $f : U \to \mathbb{R}$ be a differentiable function, where $U \subseteq \mathbb{R}^d$ is an open convex subset. Then f is convex if and only if

$$f(x) \geq f(x_0) + \nabla f(x_0)(x - x_0) \tag{8.9}$$

for every $x, x_0 \in U$.

Proof. Suppose that (8.9) holds and let $x_t = (1-t)x_0 + tx$ with $0 \le t \le 1$, where $x_0, x \in U$. To prove that f is convex we must verify the inequality

$$f(x_t) \le (1-t)f(x_0) + tf(x). \tag{8.10}$$

Let $\xi = \nabla f(x_t)$. Then

$$\begin{aligned} f(x) &\ge f(x_t) + \xi(1-t)(x-x_0) \\ f(x_0) &\ge f(x_t) - \xi t(x-x_0) \end{aligned}$$

by (8.9). If you multiply the first inequality by t, the second by $1-t$ and then add the two, you get (8.10).

Suppose on the other hand that f is a convex function. Let $x_0, x \in U$. Since U is an open subset, it follows that $(1-t)x_0 + tx \in U$ for $t \in I = (-\delta, 1+\delta)$, where $\delta > 0$ is sufficiently small. Now define the function $g : I \to \mathbb{R}$ by

$$g(t) = f((1-t)x_0 + tx) = f(x_0 + t(x-x_0)).$$

Being the composition of two differentiable functions, g is differentiable. Suppose that $0 \le \alpha \le 1$ and $t_1, t_2 \in I$. Then

$$\begin{aligned} g((1-\alpha)t_1 + \alpha t_2) &= f(x_0 + ((1-\alpha)t_1 + \alpha t_2)(x-x_0)) \\ &= f((1-\alpha)(x_0 + t_1(x-x_0)) + \alpha(x_0 + t_2(x-x_0))) \\ &\le (1-\alpha)f(x_0 + t_1(x-x_0)) + \alpha f(x_0 + t_2(x-x_0)) \\ &= (1-\alpha)g(t_1) + \alpha g(t_2) \end{aligned}$$

showing that g is a convex function. By Theorem 6.61,

$$g(1) \ge g(0) + g'(0),$$

which translates into

$$f(x) \ge f(x_0) + \nabla f(x_0)(x-x_0)$$

by using the chain rule in computing $g'(0)$.

(8.24) EXERCISE.

Prove that a bounded convex differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is constant. ♠

The following is the generalization of Corollary 6.55.

(8.25) THEOREM.

Let $f : U \to \mathbb{R}$ be a differentiable function with continuous second order partial derivatives, where $U \subseteq \mathbb{R}^d$ is a convex open subset. Then f is convex if and only if the Hessian $\nabla^2 f(x)$ is positive semidefinite for every $x \in U$. If $\nabla^2 f(x)$ is positive definite for every $x \in U$, then f is strictly convex.

Proof. We have done all the work for a convenient reduction to the one variable case. Suppose that f is convex. Then the same reasoning as in the proof of Theorem 8.23 shows that

$$g(t) = f(x + tv)$$

is a convex function for every $x \in U$ and every $v \in \mathbb{R}^d$ from an open interval $(-\delta, \delta)$ to $\mathbb{R}$ for suitable $\delta > 0$. Therefore $g''(0) = v^\top \nabla^2 f(x) v \geq 0$ by Theorem 6.54. This proves that the matrix $\nabla^2 f(x)$ is positive semidefinite for every $x \in U$. Suppose on the other hand that $\nabla^2 f(x)$ is positive semidefinite for every $x \in U$. Then Theorem 6.54 shows that $g(t) = f(x + t(y-x))$ is a convex function from $(-\delta, 1+\delta)$ to $\mathbb{R}$ for $\delta > 0$ small and $x, y \in U$, since

$$g''(\alpha) = (y-x)^\top \nabla^2 f(x + \alpha(y-x))(y-x) \geq 0$$

for $0 \leq \alpha \leq 1$. Therefore f is a convex function, since

$$\begin{aligned} f((1-t)x + ty) =& g((1-t)\cdot 0 + t \cdot 1) \\ &\leq (1-t)g(0) + tg(1) = (1-t)f(x) + tf(y). \end{aligned}$$

The same argument (using the last part of Theorem 6.54 on strict convexity), shows that g is strictly convex if $\nabla^2 f(x)$ is positive definite. It follows that f is strictly convex if $\nabla^2 f(x)$ is positive definite for every $x \in U$.

(8.26) EXERCISE.

Prove that

$$f(x,y) = x^2 + y^2$$

is a strictly convex function from $\mathbb{R}^2$ to $\mathbb{R}$. Also, prove that

$$\{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$$

is a convex subset of $\mathbb{R}^2$. ♠

(8.27) EXERCISE.

Is $f(x,y) = \cos(x) + \sin(y)$ strictly convex on some non-empty open convex subset of the plane? ♠

(8.28) EXERCISE.

Show that $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x,y) = \log(e^x + e^y)$$

is a convex function. Is f strictly convex? ♠

(8.29) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x,y) = ax^2 + by^2 + cxy,$$

where $a, b, c \in \mathbb{R}$.

(i) Show that f is a strictly convex function if and only if $a > 0$ and $4ab - c^2 > 0$.

Hint: This is a hint for the only if part. If H is the Hessian for f, then

$$f(v) = \frac{1}{2} v^\top H v,$$

where $v = (x,y)^\top$ - this is seen by a matrix multiplication computation. We know that H is positive semidefinite. If H was not positive definite, there would exist $v \neq 0$ with $f(v) = 0$. Now use $f(tv) = t^2 f(v)$ to complete the proof that H is positive definite by looking at $f((1-t)\cdot 0 + t \cdot v)$.

(ii) Suppose now that $a > 0$ and $4ab - c^2 > 0$. Show that $g(x,y) = f(x,y) + x + y$ has a unique global minimum and give a formula for this minimum in terms of a, b and c.

♠

8.6 How to decide the definiteness of a matrix

In this section we will outline a straightforward method for deciding if a matrix is positive definite, positive semidefinite, negative definite or indefinite.

Before proceeding it is a must that you do the following exercise.

(8.30) EXERCISE.

Show that a diagonal matrix

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

is positive definite if and only if $\lambda_1 > 0, \dots, \lambda_n > 0$, positive semidefinite if and only if $\lambda_1 \geq 0, \dots, \lambda_n \geq 0$ and indefinite if and only if there exists $i \neq j$ with $\lambda_i > 0$ and $\lambda_j < 0$.

♠

The crucial ingredient is the following result.

(8.31) THEOREM.

Let A be a real symmetric $n \times n$ matrix. Then there exists an invertible matrix B, such that $B^\top AB$ is a diagonal matrix.

The proof contains an algorithm for building B by different steps. We will supply examples afterwards illustrating these. An operational procedure implementing the steps is outlined in section 8.7.

Proof. Suppose that $A = (a_{ij})$. If A has a non-zero entry in the upper left hand corner i.e., $a_{11} \neq 0$, then

$$B_1^\top AB_1 = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & c_{11} & \cdots & c_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & c_{n-1,1} & \cdots & c_{n-1,n-1}, \end{pmatrix}$$

where $C = (c_{ij})$ is a real symmetric matrix and B_1 is the invertible $n \times n$ matrix

$$\begin{pmatrix} 1 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

By induction on n we may find an invertible matrix $(n-1)\times(n-1)$ matrix B_2 such that

$$B_2^\top C B_2 = \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-1} \end{pmatrix}.$$

Putting

$$B = B_1 \begin{pmatrix} 1 & 0 \\ 0 & B_2 \end{pmatrix},$$

it follows that

$$B^\top A B = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n-1} \end{pmatrix}.$$

We now treat the case of a zero entry in the upper left hand corner i.e., $a_{11} = 0$. Suppose first that $a_{jj} \neq 0$ for some $j > 1$. Let P denote the identity matrix with the first and j-th rows interchanged. The operation $A \mapsto AP$ amounts to interchanging the first and j-th columns in A. Similarly $A \mapsto P^\top A$ is interchanging that first and j-th rows in A. The matrix P is invertible and $P^\top AP$ is a symmetric matrix with $(P^\top AP)_{11} = a_{jj} \neq 0$ and we have reduced to the case of a non-zero entry in the upper left hand corner.

If $a_{ii} = 0$ for every $i = 1,\ldots,n$ we may assume that $a_{1j} \neq 0$ for some $j > 1$. Let B denote the identity matrix where the entry in the first column and j-th row is 1. The operation $A \mapsto AB$ amounts to adding the j-th column to the first column in A. Similarly $A \mapsto B^\top A$ is adding the j-th row to the first row in A. All in all we get $(B^\top AB)_{11} = 2a_{1j} \neq 0$, where we have used that $a_{ii} = 0$ for $i = 1,\ldots,n$. Again we have reduced to the case of a non-zero entry in the upper left hand corner.

(8.32) EXAMPLE.

Consider the 3×3 real symmetric matrix.

$$A = (a_{ij}) = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 8 & 4 \\ 3 & 4 & 16 \end{pmatrix}.$$

Here $a_{11} = 1 \neq 0$. Therefore the fundamental step in the proof of Theorem 8.31 applies and

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} A \begin{pmatrix} 1 & -2 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & -2 \\ 0 & -2 & 7 \end{pmatrix}$$

and again

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & -2 \\ 0 & -2 & 7 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{pmatrix}.$$

Summing up we get

$$B = \begin{pmatrix} 1 & -2 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -2 & -4 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}.$$

You are invited to check that

$$B^\top AB = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{pmatrix}.$$

♠

(8.33) EXAMPLE.

Let

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 3 \\ 1 & 2 & 1 & 4 \\ 1 & 3 & 4 & 0 \end{pmatrix}.$$

Here $a_{11} = a_{22} = 0$, but the diagonal element $a_{33} \neq 0$. So we are in the second step of the proof of Theorem 8.31. Using the matrix

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

we get

$$P^\top AP = \begin{pmatrix} 1 & 2 & 1 & 4 \\ 2 & 0 & 0 & 3 \\ 1 & 0 & 0 & 1 \\ 4 & 3 & 1 & 0 \end{pmatrix}.$$

As argued in the proof, this corresponds to interchanging the first and third columns and then interchanging the first and third rows. In total you move the non-zero a_{33} to the upper left corner in the matrix. ♠

(8.34) EXAMPLE.

Consider the symmetric matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

We have zero entries in the diagonal. As in the third step in the proof of Theorem 8.31 we must find an invertible matrix B_1, such that the upper left corner in $B_1^\top AB_1$ is non-zero. In the proof it is used that every diagonal element is zero: if we locate a non-zero element in the j-th column in the first row, we can add the j-th column to the first column and then the j-th row to the first row obtaining a non-zero element in the upper left corner. For A above we choose $j = 2$ and the matrix B_1 becomes

$$B_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

so that

$$B_1^\top AB_1 = \begin{pmatrix} 2 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix}.$$

♠

(8.35) EXERCISE.

Let A be any matrix. Show that

$$A^\top A$$

is positive semidefinite. ♠

(8.36) QUIZ.

quiz — not displayed in the pdf version.

♠

(8.37) EXERCISE.

Find inequalities defining the set

$$\left\{ (a,b) \in \mathbb{R}^2 \,\middle|\, \begin{pmatrix} 2 & 1 & a \\ 1 & 1 & 1 \\ a & 1 & b \end{pmatrix} \text{ is positive definite} \right\}.$$

Same question with positive semidefinite. Sketch and compare the two subsets of the plane $\{(a,b) \mid a,b \in \mathbb{R}\}$. ♠

(8.38) EXERCISE.

Let $f : \mathbb{R}^3 \to \mathbb{R}$ denote the function given by

$$f(x,y,z) = x^2 + y^2 + z^2 + axy + xz + yz,$$

where $a \in \mathbb{R}$. Let H denote the Hessian of f in a point $(x,y,z) \in \mathbb{R}^3$.

(i) Compute H.

(ii) Show that $f(v) = v^\top Av$ for $v = (x,y,z) \in \mathbb{R}^3$ and $A = \frac{1}{2}H$.

(iii) Compute a non-zero vector $v \in \mathbb{R}^3$, such that $Hv = 0$ in the case, where $a = 2$. Is H invertible in this case?

(iv) Show that f is strictly convex if $-1 < a < 2$.

(v) Is f strictly convex if $a = 2$?

Hint. Consider the line segment between 0 and a suitable vector $u \neq 0$, where $f(u) = 0$.

♠

(8.39) EXERCISE.

Why is the subset given by the inequalities

$$\begin{aligned} x &\geq 0 \\ y &\geq 0 \\ xy - z^2 &\geq 0 \end{aligned}$$

a convex subset of $\mathbb{R}^3$? ♠

8.7 A schematic procedure for transforming symmetric matrices

Suppose that A is a symmetric $n \times n$ matrix. We wish to find an invertible matrix B and a diagonal matrix D so that

$$B^\top AB = D.$$

Every step in the algorithm in the proof of Theorem 8.31 involve an operation on the columns of A followed by a similar operation on the rows. These steps can be carried out systematically by transforming the extended $(2n) \times n$ matrix

$$\begin{pmatrix} I_n \\ A \end{pmatrix} \qquad \text{into} \qquad \begin{pmatrix} B \\ D \end{pmatrix}. \tag{8.11}$$

The recipe is: every column operation (on A) is carried out on the full $(2n) \times n$ matrix, whereas every row operation is only carried out on the lower $n \times n$ matrix in (8.11).

Here is how this plays out for the examples above.

(8.40) EXAMPLE.

Here is the schematic procedure applied to Example 8.32:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 2 & 3 \\ 2 & 8 & 4 \\ 3 & 4 & 16 \end{pmatrix} \to \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 3 \\ 2 & 4 & 4 \\ 3 & -2 & 16 \end{pmatrix} \to \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 3 \\ 0 & 4 & -2 \\ 3 & -2 & 16 \end{pmatrix} \to \begin{pmatrix} 1 & -2 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 4 & -2 \\ 3 & -2 & 7 \end{pmatrix} \to$$

$$\begin{pmatrix} 1 & -2 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 4 & -2 \\ 0 & -2 & 7 \end{pmatrix} \to \begin{pmatrix} 1 & -2 & -4 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & -2 & 6 \end{pmatrix} \to \begin{pmatrix} 1 & -2 & -4 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{pmatrix}.$$

♠

(8.41) EXAMPLE.

Here is the schematic procedure applied to Example 8.33:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 3 \\ 1 & 2 & 1 & 4 \\ 1 & 3 & 4 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 3 \\ 1 & 2 & 1 & 4 \\ 4 & 3 & 1 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 4 \\ 2 & 0 & 0 & 3 \\ 1 & 0 & 0 & 1 \\ 4 & 3 & 1 & 0 \end{pmatrix}.$$

♠

(8.42) EXAMPLE.

Here is the schematic procedure applied to Example 8.34:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix}.$$

♠

The schematic procedure is exactly the kind of mechanical recipe a computer is made for. In the cell below the computer does the exact arithmetic — you step through the algorithm with the `Next` button. Run the cell, then press `Next` ▶: every press performs one arrow from the chains above — a column operation on the whole extended matrix, or the matching row operation on the lower half — and highlights the entries that changed. Watch B being built in the upper half while A is hammered into the diagonal matrix D in the lower half. Step back and forth until every move makes sense, then replace `A` with the matrices from Examples 8.33 and 8.34 — the stepper knows the swap and the zero-diagonal trick from the proof of Theorem 8.31 — or with a symmetric matrix of your own.

python cell — not displayed in the pdf version.

In the button below is a complete implementation. The function `diagonalize` carries out the steps from the proof of Theorem 8.31, returning the matrices B and D, and `definiteness` reads the answer off the diagonal of D. Try it on the examples above and check it against the steps in the widget.

Complete implementation.

python cell — not displayed in the pdf version.

The algorithm you have just seen is old. In 1759, the 23-year-old Joseph-Louis Lagrange published *Recherches sur la méthode de maximis et minimis*, asking precisely the question of this chapter: when is a critical point of a function of several variables a minimum? His answer amounted to repeatedly completing the square in the second order term — in modern language, the diagonal entry a_{kk} is used to eliminate the mixed terms, which is exactly step one of our procedure. For this reason the method is still known as *Lagrange reduction* of a quadratic form. The bookkeeping with matrices came much later. The systematic treatment, including the tricks for a vanishing diagonal, was collected in Felix Gantmacher's *The Theory of Matrices* (1953, English translation 1959), for decades the standard reference on the subject and still in print. It is a pleasant thought that when you press Run in the cells above, your browser is carrying out, step by step, a computation Lagrange did by hand more than 250 years ago.

9 CONVEX OPTIMIZATION

In this last chapter we will deal exclusively with convex optimization problems.

Recall that a convex optimization problem has the form

$$\min_{(x_1,\ldots,x_d)\in C} \quad f(x_1,\ldots,x_d),$$

written in the compact notation from Definition 4.6: the constraint lives under min and the function to be minimized follows. Here $C \subseteq \mathbb{R}^d$ is a *convex subset* (see Definition 4.21) and $f : C \to \mathbb{R}$ a *convex function* (see Definition 4.26). We will mainly deal with the case, where f is differentiable defined on all of $\mathbb{R}^d$ in addition to just being convex defined on C. Why single out the convex problems? Because the previous chapters showed how unwieldy optimization is in general. Gradient descent (Section 7.5.1) walks downhill to *some* point where the gradient vanishes, with no way of knowing whether a deeper valley hides behind the next ridge — training the digits network in Section 7.9 was exactly such a leap of faith. Newton's method is fast, but in Example 8.6 it cheerfully delivered a critical point that turned out to be a local *maximum*, and every critical point it finds costs a Hessian analysis before we know what we are looking at. Worst of all: having found a local minimum of a general function, no computation at that point can rule out that a better one exists somewhere else.

Convexity removes all of these obstacles in one stroke. A differentiable convex function has no saddle points and no local maxima — a critical point can only be a minimum — and a local minimum is automatically a *global* minimum (Theorem 6.13). Better still, as this chapter will show, a proposed solution can be *certified* optimal by checking finitely many conditions at that single point, without looking anywhere else. This is why convex problems form the largest class of optimization problems that can be solved reliably and at scale, and why applied mathematicians work hard to model their problems as convex ones — the fire station problem below is a first taste of such modeling.

This final chapter asks three questions about such a problem, and its parts answer them in turn. *What is it good for?* We derive the optimal hyperplane separating labeled data and meet the support vector machine, one of the most celebrated algorithms of machine learning (Sections 9.1 and 9.2). *How does a machine solve it?* Interior point methods built on logarithmic barrier functions (Section 9.3). *And how do you recognize — and certify — an optimal solution exactly?* A geometric criterion and the famous Karush-Kuhn-Tucker conditions (Sections 9.4 and 9.5).

(9.1) EXAMPLE.

Below is an example of a convex optimization problem in the plane $\mathbb{R}^2$.

$$\min_{(x,y)\in C} \quad x^2+y^2,$$

where C is the subset of points (x,y) in $\mathbb{R}^2$ satisfying

$$\begin{aligned} x+y &\geq 2 \\ y &\leq 2 \\ x &\leq 3 \\ y &\geq 1. \end{aligned}$$

♠

(9.2) EXERCISE.

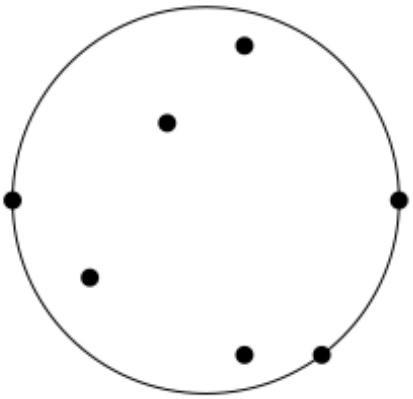
Sketch the subset C in Example 9.1. Show that Example 9.1 really is a convex optimization problem and solve it. ♠

Keep Example 9.1 in mind: it returns as a running example when we write down the KKT conditions in Section 9.5.

Below is an example of a convex optimization problem modelling the real life problem of placing a fire station (center of circle) so that the maximal distance to the surrounding houses (points to be enclosed) is minimal.

(9.3) EXAMPLE.

Given n points $(x_1,y_1),\ldots,(x_n,y_n) \in \mathbb{R}^2$, what is the center and radius of the smallest circle containing these points?



We can write this optimization problem as

$$\min_{(x,y)\in C} \quad r, \tag{9.1}$$

where

$$C = \{(x,y) \in \mathbb{R}^2 \mid (x-x_i)^2 + (y-y_i)^2 \leq r^2 \quad \text{for} \quad i = 1,\ldots,n\}.$$

Upon rewriting this turns into the optimization problem

$$\min_{(x,y)\in C'} \quad x^2+y^2+\lambda, \tag{9.2}$$

where

$$C' = \{(x,y,\lambda) \in \mathbb{R}^3 \mid x_i^2 + y_i^2 \leq 2x_i x + 2y_i y + \lambda \quad \text{for} \quad i = 1,\ldots,n\}$$

and $\lambda = r^2 - x^2 - y^2$. ♠

(9.4) EXERCISE.

Prove that (9.1) and (9.2) both are convex optimization problems. Explain how (9.1) is rewritten into (9.2).

Hint. Expand

$$(x-x_i)^2 + (y-y_i)^2 \leq r^2$$

and put $\lambda = r^2 - x^2 - y^2$.

♠

The rewriting (9.2) may look unmotivated now, but it earns its keep in Section 9.3, where an interior point method computes the optimal fire station location.

9.1 The optimal separating hyperplane

In section 5.3.2 we were presented with labeled data

$$(v_1, y_1), \ldots, (v_m, y_m), \tag{9.3}$$

where $v_i \in \mathbb{R}^d$ and $y_i = \pm 1$. The task at hand was to separate differently labeled data by a hyperplane $\alpha^\top v + \beta = 0$, such that

$$\alpha^\top v_i + \beta > 0 \quad \text{if} \quad y_i = 1 \tag{9.4}$$
$$\alpha^\top v_i + \beta < 0 \quad \text{if} \quad y_i = -1 \tag{9.5}$$

for $i = 1, \ldots, m$. Please browse back to Definition 4.40 for the definition of a hyperplane in $\mathbb{R}^d$.

Such a hyperplane need not exist: for the points $(1,1), (-1,-1)$ with label $+1$ and $(-1,1), (1,-1)$ with label -1, the four inequalities in (9.4) contradict each other, as you showed in Exercise 5.9. Separability is a genuine assumption on the data — we return to what can be done without it with the kernel functions of Section 9.2.

If the data in (9.3) can be separated according to (9.4), we may find a hyperplane $(\alpha^*)^\top x + \beta^* = 0$, such that

$$(\alpha^*)^\top v_i + \beta^* \geq 1 \quad \text{if} \quad y_i = 1 \tag{9.6}$$
$$(\alpha^*)^\top v_i + \beta^* \leq -1 \quad \text{if} \quad y_i = -1. \tag{9.7}$$

(9.5) EXERCISE.

How do you go from (9.4) to (9.6)?

Hint: Suppose that

$$\alpha^\top v_i + \beta > 0 \quad \text{if} \quad y_i = 1$$
$$\alpha^\top v_i + \beta < 0 \quad \text{if} \quad y_i = -1.$$

Let

$$N = \min\{\alpha^\top v_i + \beta \mid y_i = 1\}$$
$$M = \max\{\alpha^\top v_i + \beta \mid y_i = -1\}.$$

Show that $N > 0$ and $M < 0$. How can N and M be applied in constructing α^* and β^*?

Hint: Consider

$$\alpha^\top v_i + \beta \geq N > 0 \quad \text{if} \quad y_i = 1$$
$$\alpha^\top v_i + \beta \leq M < 0 \quad \text{if} \quad y_i = -1.$$

What is special about $s = \min(N, |M|)$?

♠

What does optimal hyperplane mean in this setting? It is the one maximizing the width of the strip between the two labeled clusters.

(9.6) FIGURE.

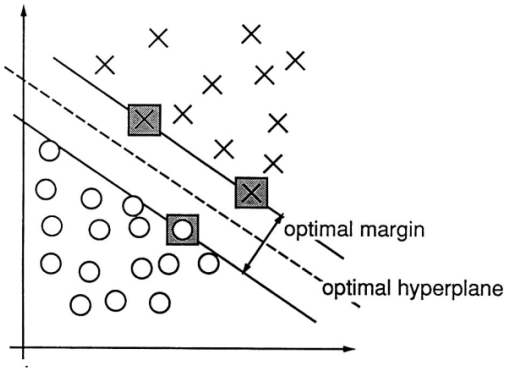


Figure 9.1: Figure from the Cortes and Vapnik paper: Support vector networks, Machine Learning, 1995.

The inequalities (9.6) describe this strip precisely. The two parallel hyperplanes

$$H_+ \colon\ \alpha^\top v + \beta = 1 \qquad \text{and} \qquad H_- \colon\ \alpha^\top v + \beta = -1$$

bound a strip containing no data points: by (9.6) the points labeled $+1$ lie on or beyond H_+ and the points labeled -1 on or beyond H_-. The separating hyperplane H given by $\alpha^\top v + \beta = 0$ runs down the middle of the strip. This is exactly the picture above: H is the solid line, H_+ and H_- the dashed ones.

Here lies a crucial insight, only implicit in the paper by Cortes and Vapnik: the normalization to ± 1 in (9.6) costs nothing — rescale α and β, as in the exercise above — and it makes the width of the strip visible in α alone.

Indeed, the distance from a point u to the hyperplane H is

$$\frac{|\alpha^\top u + \beta|}{|\alpha|}$$

by Exercise 9.7 below. A point u on H_+ satisfies $\alpha^\top u + \beta = 1$ and therefore has distance $1/|\alpha|$ to H, and the same holds on H_-. The strip therefore has width

$$\frac{1}{|\alpha|} + \frac{1}{|\alpha|} = \frac{2}{|\alpha|},$$

and the optimal hyperplane is the one making $|\alpha|$ as small as possible subject to (9.6). Since minimizing $|\alpha|$ is the same as minimizing $|\alpha|^2$ — and the square is a differentiable (convex!) function — we have arrived at a convex optimization problem.

(9.7) EXERCISE.

Let H be the hyperplane in $\mathbb{R}^d$ given by $\alpha^\top v + \beta = 0$ and let $u \in \mathbb{R}^d$. The point closest to u in H can be found by solving the optimization problem

$$\min_{v \in H} |v - u|^2. \tag{9.8}$$

Explain why (9.8) is a convex optimization problem.

Show how Theorem 7.39 can be used to solve this optimization problem by first deducing the equations

$$-2(u - v) + \lambda \alpha = 0 \tag{9.9}$$

$$\alpha^\top v + \beta = 0 \tag{9.10}$$

for the Lagrange multiplier λ. Notice here that $-2(u-v) + \lambda\alpha = 0$ above really contains d equations, whereas $\alpha^\top v + \beta = 0$ is only one equation in $x_1, \ldots, x_d$, where $v = (x_1, \ldots, x_d)^\top$. Solve the equations (9.9) for $v \in \mathbb{R}^d$ and $\lambda \in \mathbb{R}$. How can we be sure that v really is a minimum in (9.8)?

Finally show that the distance from H to u is given by the formula

$$\left| \frac{\alpha^\top u + \beta}{|\alpha|} \right|.$$

♠

Summing up, we have proved the following result.

(9.8) THEOREM.

Let the points $v_1, \dots, v_m \in \mathbb{R}^d$ be labeled by $y_1, \dots, y_m \in \{\pm 1\}$. Then the optimal hyperplane $H = \{v \in \mathbb{R}^d \mid \alpha^\top v + \beta = 0\}$ separating the points is given by the optimization problem

$$\begin{array}{ll} \min & |\alpha|^2 \\ \alpha^\top v_i + \beta \geq 1 \text{ if } y_i = 1 & \\ \alpha^\top v_i + \beta \leq -1 \text{ if } y_i = -1 & \end{array}$$

for $i = 1, \dots, m$.

The vectors v_i among the data points $v_1, \dots, v_m$ satisfying $\alpha^\top v_i + \beta = 1$ or $\alpha^\top v_i + \beta = -1$ are called *support vectors*.

An optimal solution always has support vectors. If no data point satisfied its constraint with equality, all the numbers $|\alpha^\top v_i + \beta|$ would exceed 1 by a common factor $s > 1$, and dividing both α and β by s would preserve the constraints in Theorem 9.8 while shrinking $|\alpha|$ — so the strip was not the widest after all. A little more thought (shift β to re-center the strip, then rescale) shows that *both* dashed lines must carry support vectors, just as in the figure above.

(9.9) EXAMPLE.

Let us explicitly write up the optimization problem in Theorem 9.8 in a very simple situation: finding the best line $y = ax + b$ separating the points $(1,1)$ and $(2,2)$. In the notation of (9.6), we have (without the stars on α and β)

$$\alpha = \begin{pmatrix} a \\ -1 \end{pmatrix} \qquad \text{and} \qquad \beta = b$$

so that

$$\alpha^\top \begin{pmatrix} x \\ y \end{pmatrix} + b = ax - y + b = 0.$$

The points are

$$v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \text{and} \qquad v_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix},$$

where $y_1 = 1$ and $y_2 = -1$.

Therefore the optimization problem in Theorem 9.8 becomes

$$\begin{array}{ll} \min & 1 + a^2. \\ a + b \geq 2 & \\ 2a + b \leq 1 & \end{array} \tag{9.11}$$

♠

(9.10) EXERCISE.

Solve the optimization problem (9.11) and verify that the best line from the optimization problem is the one we expect it to be. Also, check how WolframAlpha solves this optimization problem.

Hint. You could maybe use Fourier-Motzkin elimination to show that

$$\begin{aligned} a+b &\geq 2 \\ 2a+b &\leq 1 \end{aligned}$$

implies $a \leq -1$.

♠

Notice that the optimization problem in Theorem 9.8 has number of constraints equal to the number of points to be separated. Notice also that it has no solution at all when the data cannot be separated — the constraints are then contradictory. The remedy, called the *soft margin*, is stated precisely when we meet the parameter `C` of `sklearn` in the next section.

Usually one does not use the optimization problem in Theorem 9.8, but rather its socalled (Lagrange) dual for finding the optimal hyperplane. This dual optimization problem uses that the normal vector α is a linear combination

$$\alpha = \lambda_1 y_1 v_1 + \cdots + \lambda_m y_m v_m \tag{9.12}$$

of the support vectors weighted by their labels. It is an optimization problem in $\Lambda^\top = (\lambda_1, \ldots, \lambda_m)$ from (9.12) and looks like

$$\max_{\substack{\Lambda \geq 0 \\ \Lambda^\top Y = 0}} \quad \lambda_1 + \cdots + \lambda_m - \frac{1}{2}\Lambda^\top D \Lambda, \tag{9.13}$$

where $Y = (y_1, \ldots, y_m)^\top$ is the vector of labels attached to the points $v_1, \ldots, v_m$ and D is the symmetric $m \times m$ matrix given by

$$D_{ij} = y_i y_j v_i^\top v_j = y_i y_j v_i \cdot v_j. \tag{9.14}$$

(9.11) REMARK.

Notice that the dual optimization problem is an optimization problem in $\mathbb{R}^m$, where m is the number of data points. This can be in stark contrast to the original optimization problem in Theorem 9.8, which is an optimization problem in $\mathbb{R}^d$, where d is the dimension of the data points. Sometimes the data points are high dimensional and it pays to solve the dual optimization problem.

(9.12) EXAMPLE.

Let us write down the dual optimization problem for the points in Example 9.9. Here

$$\Lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}, \quad Y = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} 2 & -4 \\ -4 & 8 \end{pmatrix}$$

so that the dual optimization problem becomes

$$\max_{\substack{\lambda_1 \geq 0,\ \lambda_2 \geq 0 \\ \lambda_1 - \lambda_2 = 0}} \quad \lambda_1 + \lambda_2 - \lambda_1^2 - 4\lambda_2^2 + 4\lambda_1\lambda_2.$$

This reduces to the optimization problem of maximizing $2\lambda - \lambda^2$ subject to $\lambda \geq 0$, which has the solution $\lambda = 1$. Therefore the optimal hyperplane has normal vector

$$\alpha = \lambda_1 y_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \lambda_2 y_2 \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}.$$

The constraints hold with equality at the two support vectors, so $\beta = 3$ and the optimal hyperplane is the line $x+y=3$ — the perpendicular bisector between the two points, exactly as geometry demands. ♠

The dual optimization problem (9.13) can be derived formally from the original optimization problem in Theorem 9.8. This is, however, beyond the scope of this course (see section 2.1 of the Cortes and Vapnik paper).

9.2 Support vector machines

The optimization problem of Theorem 9.8, its dual (9.13) and the kernel functions introduced below make up one of the most celebrated algorithms of machine learning: the *support vector machine* (SVM) from the Cortes and Vapnik paper cited above. You do not have to implement it yourself: the python library `sklearn` ships an industrial implementation in `sklearn.svm`, and with `kernel="linear"` and a very large parameter `C` it solves exactly the optimization problem of Theorem 9.8. The parameter `C` is the price of violating the margin, and we can say precisely what that means: for any labeled data — separable or not — `sklearn` solves the *soft margin* problem

$$\min_{\substack{y_i(\alpha^\top v_i+\beta)\geq 1-\xi_i \\ \xi_i \geq 0}} \frac{1}{2}|\alpha|^2 + C(\xi_1+\cdots+\xi_m)$$

for $i = 1,\ldots,m$, from section 3 of the Cortes and Vapnik paper. The new *slack variable* ξ_i measures how far the point v_i falls short of its margin constraint (for $\xi_i = 0$ the constraint is Theorem 9.8's, with the two label cases abbreviated into one inequality), and the objective charges C per unit of shortfall. A huge `C` makes slack unaffordable and recovers the hard margin of Theorem 9.8; a moderate `C` buys a wider strip on messy data, at the price of points inside the strip or even on the wrong side — the Gaussian kernel cell later in this section runs with `C=10` in exactly this spirit. In the dual, `C` shows up as the extra constraint $\lambda_i \leq C$: a cap on how much influence any single data point can buy.

(9.13) EXERCISE.

Three small observations that explain what `sklearn` is doing.

1. Check that for $\xi_i = 0$ the constraint $y_i(\alpha^\top v_i+\beta) \geq 1-\xi_i$ is exactly the constraint of Theorem 9.8, for both values of the label y_i.

2. Show that the soft margin problem is feasible for *any* labeled data — no separability needed. So unlike the problem in Theorem 9.8, it always has something to optimize.

3. Suppose the data *can* be separated, and let (α^*,β^*) be the optimum of Theorem 9.8. Show that the slack at the soft margin optimum satisfies
$$\xi_1+\cdots+\xi_m \leq \frac{|\alpha^*|^2}{2C},$$
so the total slack vanishes as C grows — the hard margin reappears.

Hint. For the second part: $\alpha = 0$, $\beta = 0$ and $\xi_i = 1$ satisfy every constraint. For the third: (α^*,β^*) with all $\xi_i = 0$ is one candidate in the soft margin problem, so the optimum costs at most $\frac{1}{2}|\alpha^*|^2$ — and the term $C(\xi_1+\cdots+\xi_m)$ is part of that cost.

♠

9.2.1 Two clusters in the plane

The cell below separates two clusters of points. It prints the optimal α and β, draws the optimal hyperplane (solid) with the two margin lines $\alpha^\top v + \beta = \pm 1$ (dashed), and circles the support vectors. Look closely at the printed numbers: at the support vectors, $\alpha^\top v_i + \beta$ equals ± 1 exactly — these are the constraints of Theorem 9.8 holding with equality, computed numerically.

python cell — not displayed in the pdf version.

Only the circled points matter: move any other point a little (rerun and try!) and the optimal hyperplane does not budge. The hyperplane is carried entirely by its support vectors — hence the name.

(9.14) QUIZ.

quiz — not displayed in the pdf version.

♠

9.2.2 Reading handwritten digits

Support vector machines are not a museum piece. Before the deep learning era they were the state of the art for tasks like text classification and protein classification, and for small data sets with many measurements per data point they remain a favorite — this is exactly the situation of Remark 9.11, where the dual problem has one variable per data point rather than one per dimension.

Here is an example you already know: the 8×8 handwritten digits that the neural network of Section 7.9 read in the previous chapter — every digit its vector of 64 pixel values, a data point in $\mathbb{R}^{64}$. We ask the machine the honest yes/no question of practice: *is this digit a* 5*?* Label every 5 with $+1$ and everything else with -1, and Theorem 9.8 applies unchanged. Note what just happened: the "two clusters" of the theorem are two *labels*, nothing more. The -1 class is a motley union of nine kinds of digit, and the mathematics does not blink — the two-blob pictures we have drawn are visual aids, not hypotheses. This one-against-the-rest trick is also how binary classifiers handle many classes in practice: train ten machines, one per digit, and let the largest decision value win.

It is far from obvious that a single hyperplane in $\mathbb{R}^{64}$ can separate the fives from the nine-digit crowd — but for these 1500 training digits it can, so we demand the hard margin (a huge `C`) and get exactly the optimal hyperplane of Theorem 9.8.

The best part is the picture: the support vectors are themselves digits, and they are the machine's *entire concept of five-ness*. The optimization problem singles out a handful of borderline fives (top row below) together with a gallery of *five-impersonators* — digits of other classes that flirt with looking like a 5, shown with their true labels in the middle row. When we ran it, the impersonators were mostly 9s, 7s, 3s and 6s: exactly the digits that share strokes with a five. The bottom row shows some of the ignored, neatly written majority — move any of those and the hyperplane does not budge. The cell prints your exact counts, which may drift slightly with the library version.

Better still, the number $\alpha^\top v + \beta$ measures *how manifestly* a digit is a 5. Recall from (9.6) that the optimal hyperplane was normalized to give values ≥ 1 on one cluster and ≤ -1 on the other, with the margin strip $-1 < \alpha^\top v + \beta < 1$ in between. So a digit with $\alpha^\top v + \beta \geq 1$ is manifestly a five — it clears the margin — while a value near 0 means the machine is honestly unsure. The cell sorts the 297 unseen digits by $\alpha^\top v + \beta$ and displays the extremes and the ambiguous middle. When we ran it, every unseen five landed on the correct side — nearly all manifestly, a couple hesitantly inside the margin — at the price of a couple of false alarms among the non-fives; again the cell prints your exact counts.

python cell — not displayed in the pdf version.

Study the strip: on the far left (large negative $\alpha^\top v + \beta$) unmistakably not fives, on the far right textbook fives, and in the middle handwriting that would make a human hesitate too. Note the family resemblance to the Challenger neuron of the previous chapter: there the same affine expression was fed through the sigmoid to become a probability; here it is left raw and read against the margin.

Compare with the previous chapter, too: there, reading digits took a 2410-parameter neural network and thousands of gradient descent steps. Here the single yes/no question *is it a five?* is answered by one convex optimization problem carried by a few dozen support vectors, with the guarantee from Theorem 6.13 that the hyperplane found is globally optimal.

9.2.3 Separating sentences

Digits were easy to feed to the machine: an image is born a vector. But most of what the world wants classified is *text* — support requests, product reviews, spam. Chapter 5 already showed the way in: after the one-hot encoding (5.17) came its modern replacement, the folded cell of 18 words embedded by the small language model all-MiniLM-L6-v2 — the model behind *Ask the book* on this very page — which turns any sentence into a learned vector in $\mathbb{R}^{384}$ whose geometry carries meaning.

That is one half of the modern recipe for classifying text. The other half is this chapter. Take a big pretrained model, *freeze it*, embed your sentences, and train nothing but a separating hyperplane on the resulting vectors. Practitioners call this a *linear probe*, and it is everyday industrial machine learning: under many a spam filter, content moderator and ticket router sits exactly the soft-margin problem of Exercise 9.13, working on top of somebody else's embeddings.

Let us run the recipe, honestly and end to end. The task: do film reviews express enthusiasm or disappointment? Below are 46 short reviews written for this book — the first 20 enthusiastic, the next 20 disappointed, and a final 6 held back as unseen test sentences. A language model is far too large to run inside a web page, so the reviews were embedded once, in advance, and their vectors travel with this page exactly like the 18 words of Chapter 5: 8-bit integers, one scaling factor per vector. Open the fold and you are looking at rows of $\mathbb{R}^{384}$; run the cell to decode them into the matrix `VECS`.

Python: 46 film reviews as learned vectors in dimension 384.

python cell — not displayed in the pdf version.

Now the machine of this chapter takes over: a soft-margin hyperplane in $\mathbb{R}^{384}$, computed from the 40 labeled training reviews. For the digits we could insist on the hard margin; here we deliberately keep `sklearn`'s default soft margin ($C = 1$) — you will see why in a moment. The six unseen reviews are then judged by the sign of $\alpha^\top v + \beta$ alone. The machine never sees their labels; read its verdicts and judge it yourself.

python cell — not displayed in the pdf version.

Read the strip of verdicts as you read the strip of digits: at the bottom, confident disappointment; at the top, confident enthusiasm; in the middle, hesitation. In our run the machine judges five of the six unseen reviews correctly and hesitates exactly where a human must read twice — "Ninety minutes I will never get back" contains not a single negative word. The one it gets wrong, "I want to watch it again tomorrow", is the same phenomenon in a purer form: its warmth is entirely implication, and this small embedding model reads vocabulary more fluently than implication. Look also at the training reviews the soft margin gives up on — "Two hours flew by; I did not look at my phone once" is enthusiasm expressed in the words of boredom. The machine's failures are not random noise; they trace the precise boundary of what its embedding understands. (The embeddings inside today's large chatbot models draw that boundary much further out.)

One number above deserves a hard look: nearly every training review is a support vector — 39 of 40 in our run, against a few dozen of 1500 for the digits. This is the signature of high dimension. Forty points in $\mathbb{R}^{384}$ have so much room that a hyperplane can do almost anything — separating them is nearly free, the way a degree-39 polynomial fits any 40 points of Chapter 2 effortlessly — so the optimal hyperplane ends up leaning on almost

all of them, and a perfect fit to the training data proves nothing. That is why the six held-out sentences are the only honest measure in this subsection, and why we kept the soft margin: the two training reviews it refuses to honor are genuinely misleading sentences, and forcing the hyperplane to honor them anyway is possible — but you pay for it on the unseen six (see the exercise below).

An embedding, then, is a *learned lift*: raw text has no coordinates at all, and the language model supplies 384 of them — so good that a plain hyperplane suffices. Creating coordinates to create separability is precisely the theme of the next subsection, where, instead of borrowing a lift learned from text, we build one by hand from polynomials.

(9.15) EXERCISE.

In the training cell, replace the machine by `SVC(kernel="linear", C=1e6)` — the hard margin of the digits example — and rerun.

1. Check that the list of sacrificed training reviews is now empty: all 40 training reviews are on their correct sides.
2. Now read the verdicts on the six unseen reviews. Compare with the soft margin's verdicts. Which figure of Chapter 2 does this pair of observations redraw (see Figure 2.41)?

♠

(9.16) EXERCISE.

The machine misjudged "I want to watch it again tomorrow". Fix it the practitioner's way: move that review into the training data,

```
Xtr = VECS[:41]
ytr = np.array([1]*20 + [-1]*20 + [1])
Xte = VECS[41:]
```

and retrain. The repaired review is now judged correctly — but watch the two hesitant negative reviews in our run drift across to the wrong side. Explain how adding a single training point can move the boundary *against* other points, and why the honest cure is more data rather than one patch per mistake.

> **Hint.** The new point lands inside the old margin on the wrong side, so the optimization must tilt the hyperplane to accommodate it — and a tilt moves the decision value of *every* sentence, including the ones that were barely correct. A machine patched one complaint at a time plays whack-a-mole; a machine trained on hundreds of reviews pins the hyperplane down from all sides at once.

♠

(9.17) EXERCISE.

Why is almost every review a support vector, when only a few dozen of the 1500 digits were? Explain in terms of the dimension and the number of data points, using the polynomial analogy above: separating 40 labeled points in $\mathbb{R}^{384}$ leaves a hyperplane enormous freedom, just as threading a degree-39 polynomial through 40 data points is always possible (Chapter 2). What would you expect to happen to the fraction of support vectors if the training set grew to thousands of reviews? ♠

9.2.4 Separating by non-linear functions

Sometimes no line can separate the data — we promised in Section 9.1 to deal with this. The remedy is not to bend the line. It is to move the points.

> If no hyperplane separates the data, map the data into a higher dimensional space, where a hyperplane can do the job.

Here is the idea in dimension one, where a hyperplane is just a point. The numbers $-1, 1, 2$ with labels $+1, -1, +1$ cannot be separated by any point on the line. Now lift the numbers to the parabola in the plane using

$$\varphi(x) = (x, x^2).$$

The lifted points $(-1,1), (1,1)$ and $(2,4)$ *can* be separated — for example by the line $y = x + 1$, as the cell below shows.

And a separation upstairs is a separation downstairs: the number x is classified by the sign of

$$f(x) = x^2 - x - 1,$$

which is positive at -1 and 2 and negative at 1. A hyperplane in the plane became a *polynomial* on the line.

python cell — not displayed in the pdf version.

The four points from Section 9.1 that no line could separate — $(1,1), (-1,-1)$ labeled $+1$ and $(-1,1), (1,-1)$ labeled -1 — surrender to the same trick, and here a single extra number suffices: the product

$$\varphi(x, y) = xy$$

equals $+1$ on the first pair and -1 on the second. Keep this little product in mind; you will meet it again inside the kernel of Example 9.18.

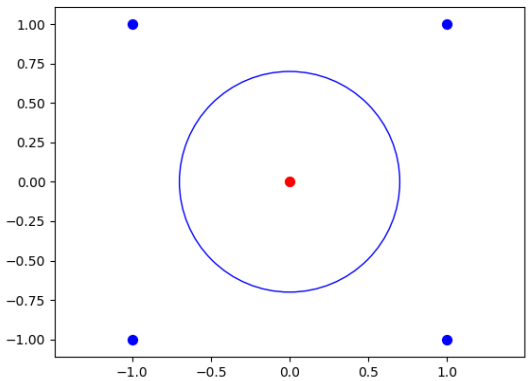
For a taste of the geometry, consider the five points

$$(-1,1), \quad (1,1), \quad (1,-1), \quad (-1,-1) \quad \text{and} \quad (0,0),$$

where $(0,0)$ is to be separated from the four corners. No line can do it, but a circle

$$x^2 + y^2 = r^2, \qquad 0 < r < \sqrt{2}, \tag{9.15}$$

certainly can:



Under the map $\varphi(x, y) = (x^2, y^2)$ something drastic happens: all four corners land on the *same* point $(1,1)$, while $(0,0)$ stays put. Separating two points is no challenge — and pulling a separating line $x_1 + y_1 = r^2$ back through φ gives exactly the circle (9.15). The right map does not just help; it can make the problem trivial.

The general recipe is now clear: find a map

$$\varphi : \mathbb{R}^d \to \mathbb{R}^N,$$

such that the transformed data

$$(\varphi(x_1), y_1), \ldots, (\varphi(x_m), y_m)$$

becomes linearly separable, and run the machinery of Theorem 9.8 in $\mathbb{R}^N$. When the coordinates of φ are monomials, the optimal hyperplane $\alpha^\top \varphi(v) + \beta = 0$ upstairs is an intricate *polynomial curve* downstairs.

The cell below shows the recipe at full power. It scatters random points in the plane, labels them by which side of the snaky cubic curve $y = \frac{x^3 - 2x}{2}$ they fall on, lifts them to $\mathbb{R}^9$ using *all* monomials of degree at most three,

$$\varphi(x, y) = (x, y, xy, x^2, y^2, x^3, x^2y, xy^2, y^3),$$

and hands the lifted points to the support vector machine with `kernel="linear"` — a plain hyperplane in $\mathbb{R}^9$, nothing we have not seen before. Pulled back to the plane, that hyperplane is the black cubic curve in the plot, hugging the gap in the data and separating the two clusters perfectly — carried, as always, by a handful of support vectors.

python cell — not displayed in the pdf version.

There is a catch. For interesting data the dimension N explodes: degree three in two variables already took 9 coordinates, and higher degrees in more variables quickly outgrow any computer.

The dual optimization problem comes to the rescue. In (9.13) the number of variables is the number of data points m — not N — and the transformed data enter only through the $m \times m$ matrix

$$D_{ij} = y_i y_j \, \varphi(x_i) \cdot \varphi(x_j). \tag{9.16}$$

The dimension N appears nowhere, provided we find a clever way of getting our hands on the dot products $\varphi(x_i) \cdot \varphi(x_j)$ in (9.16) without ever computing φ itself. Here an old concept from pure mathematics called kernel functions helps us.

9.2.5 Kernel functions

A kernel function is a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, that is a hidden dot product in the following sense: there exists a function

$$\varphi : \mathbb{R}^d \to \mathbb{R}^N,$$

such that

$$k(u, v) = \varphi(u) \cdot \varphi(v). \tag{9.17}$$

(9.18) EXAMPLE.

Let $k : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ be given by

$$k(u, v) = (u^\top v + 1)^2.$$

Then

$$k((x_1, y_1), (x_2, y_2)) = (x_1 x_2 + y_1 y_2 + 1)^2 \tag{9.18}$$

$$= x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 x_2 y_1 y_2 + 2x_1 x_2 + 2y_1 y_2 + 1. \tag{9.19}$$

One gleans from (9.18) that k is a kernel function, since (9.17) is satisfied for $\varphi : \mathbb{R}^2 \to \mathbb{R}^6$ given by

$$\varphi(a, b) = (a^2, b^2, \sqrt{2}ab, \sqrt{2}a, \sqrt{2}b, 1).$$

♠

Once we have a kernel function for φ we can replace the matrix in (9.16) by

$$D_{ij} = y_i y_j \, k(x_i, x_j)$$

and proceed to solve the optimization problem without worrying about the sometimes insurmountable size of $\mathbb{R}^N$.

The kernel of Example 9.18 is built into `sklearn` as `kernel="poly", degree=2, coef0=1`. The cell below turns it loose on the five points from the beginning of this subsection, where $(0,0)$ had to be separated from the four corners. The support vector machine — solving the dual optimization problem with the kernel in place of the dot product — finds a closed separating curve, and we never leave $\mathbb{R}^2$ to compute in $\mathbb{R}^6$.

python cell — not displayed in the pdf version.

Compare the black curve with the circle (9.15) you would draw by hand — and with the curve you will compute in the kernel perceptron exercise below.

9.2.6 The Gaussian kernel and infinite dimension

The most famous kernel of all is the *Gaussian kernel*

$$k(u,v) = e^{-|u-v|^2},$$

known to `sklearn` as `kernel="rbf"` (for radial basis function). Which map φ hides behind this one? Let us look in dimension one, where the answer is both elementary and dizzying. Using the series $e^t = \sum_{k=0}^{\infty} \frac{t^k}{k!}$ of the exponential function on the last factor in $e^{-(u-v)^2} = e^{-u^2} e^{-v^2} e^{2uv}$, we get

$$e^{-(u-v)^2} = \sum_{k=0}^{\infty} \left(\sqrt{\frac{2^k}{k!}} \, e^{-u^2} u^k \right) \left(\sqrt{\frac{2^k}{k!}} \, e^{-v^2} v^k \right).$$

This says precisely that $k(u,v) = \varphi(u) \cdot \varphi(v)$ for

$$\varphi(u) = e^{-u^2} \left(1, \ \sqrt{2}\,u, \ \sqrt{\tfrac{2^2}{2!}}\,u^2, \ \sqrt{\tfrac{2^3}{3!}}\,u^3, \ \ldots \right)$$

— one coordinate for *every* power of u.

The feature space behind the Gaussian kernel is *infinite dimensional*. No computer could store a single vector $\varphi(u)$, let alone find a separating hyperplane among such vectors. And none has to: the dual problem only ever asks for the numbers $k(x_i, x_j)$ — one exponential each. The support vector machine quietly optimizes over an infinite dimensional space while your laptop multiplies small matrices. A similar expansion works in $\mathbb{R}^d$ for every d.

There is also a completely down-to-earth way of viewing the resulting classifier. As in (9.12), the optimal normal vector is a linear combination of the transformed support vectors, so a new point v is classified by the sign of

$$\alpha^\top \varphi(v) + \beta = c_1 \, e^{-|v-x_1|^2} + \cdots + c_m \, e^{-|v-x_m|^2} + \beta$$

for suitable numbers c_i, non-zero only at the support vectors: a landscape of Gaussian hills (where $c_i > 0$) and valleys (where $c_i < 0$), with the separating curve at sea level. With enough hills and valleys any coastline can be traced, which is why the Gaussian kernel separates essentially any labeled data set.

The cell below turns it loose on data that would humiliate any polynomial of modest degree: two interlocking spirals. The background coloring is exactly the landscape $\alpha^\top \varphi(v) + \beta$, and the black curve is sea level.

python cell — not displayed in the pdf version.

Zero training errors on the two spirals, and the landscape in the picture is built from one Gaussian hill or valley per support vector — the cell prints how many there are (a substantial share of the 200 points; wiggly boundaries are expensive). Everything computed in $\mathbb{R}^2$, everything happening in $\mathbb{R}^\infty$.

9.2.7 The kernel perceptron algorithm

Recall the stunningly simple perceptron algorithm from section 5.3.2. This algorithm can be modified to handle non-linear separation too by using kernel functions. In fact, this modification was one of the inspirations for the development of the support vector machines described above.

After having mapped a set of vectors $x_1,\dots,x_m \in \mathbb{R}^d$ with labels $y_1,\dots,y_m \in \{\pm 1\}$ to $\varphi(x_1),\dots,\varphi(x_m)$, via $\varphi : \mathbb{R}^d \to \mathbb{R}^N$, we are looking for a vector $w \in \mathbb{R}^N$, such that

$$w^\top \varphi(x_i) > 0 \quad \text{if} \quad y_i = 1 \tag{9.20}$$
$$w^\top \varphi(x_i) < 0 \quad \text{if} \quad y_i = -1. \tag{9.21}$$

Such a vector is expressible as

$$w = \lambda_1 \varphi(x_1) + \cdots + \lambda_m \varphi(x_m).$$

The (dual) perceptron algorithm works adjusting the coefficients $\lambda_1,\dots,\lambda_m$ successively as follows: if $w^\top$ is wrong about the placement of $\varphi(x_j)$ in (9.20) i.e., if $y_j w^\top \varphi(x_j) < 0$, then let

$$\lambda_j := \lambda_j + y_j.$$

If we have a kernel function k for φ, then

$$w^\top \varphi(x_j) = w \cdot \varphi(x_j) = \sum_{i=1}^m \lambda_i \varphi(x_i) \cdot \varphi(x_j) = \sum_{i=1}^m \lambda_i k(x_i, x_j)$$

and we can use the kernel function in the algorithm without resorting to computing φ and the inner product in $\mathbb{R}^N$.

(9.19) EXERCISE.

Use the kernel function in Example 9.18 and the kernel perceptron algorithm to separate

$$((-1,-1),-1), \quad ((-1,1),-1), \quad ((1,-1),-1), \quad ((0,0),1), \quad ((1,1),1).$$

Sketch the points and the separating curve. ♠

9.3 Interior point methods

The support vector machines of the previous section simply called `sklearn`, which solved the optimization problem of Theorem 9.8 for us in milliseconds. But how? For the record, `sklearn` attacks the *dual* problem (9.13) with sequential minimal optimization (SMO): repeatedly pick two of the multipliers λ_i, λ_j — two at a time, since the constraint $\Lambda^\top Y = 0$ must be preserved — and improve them by solving the resulting two-variable problem, for which there is a closed formula. That the dual has one variable per data point and touches the data only through the matrix D is precisely what makes this fast, kernels included. So the dual problem of Section 9.1 is not just theory; it is what actually runs on your machine.

In this section we develop a different, completely general method for constrained convex optimization — the classical workhorse known as the *interior point method*. There is a very nice trick (probably going back to von Neumann) for solving constrained optimization problems of the form

$$\min_{(x_1,\dots,x_d)\in C} \quad f(x_1,\dots,x_d), \tag{9.22}$$

where C is defined by the differentiable functions $g_i : \mathbb{R}^d \to \mathbb{R}$ as

$$C = \{v \in \mathbb{R}^d \mid g_1(v) \le 0, \dots, g_m(v) \le 0\}.$$

The functions g_i define the boundary (or barrier) of C. We use them to define the logarithmic *barrier function*

$$B(v) = -\sum_{i=1}^{m} \log(-g_i(v))$$

defined on the interior

$$C^o = \{v \in \mathbb{R}^d \mid g_1(v) < 0, \ldots, g_m(v) < 0\}.$$

The boundary of C is

$$\partial C = \{v \in C \mid g_1(v) = 0 \vee \cdots \vee g_m(v) = 0\}.$$

You can see that the logarithmic barrier function explodes (becomes unbounded), when a vector $v \in C^o$ approaches ∂C, since $-\log(t)$ is unbounded as $t \to 0$ for $t > 0$.

The cool idea is to consider the function

$$f_\varepsilon(v) = f(v) + \varepsilon B(v) \tag{9.23}$$

for $\varepsilon > 0$. This function has a global minimum $v_\varepsilon \in C^o$.

(9.20) EXERCISE.

Prove that f_ε is a convex function if f and $g_1, \ldots, g_m$ are convex functions.

> **Hint.** Prove and use that if f is a decreasing convex function (in one variable) and g is a convex function, then $f(-g(x))$ is a convex function, where we assume the composition makes sense.

♠

The upshot is that $v_\varepsilon \to v_0$ as $\varepsilon \to 0$. This is the content of the following theorem, which we will not prove.

(9.21) THEOREM.

Let v_ε be a point in C^o with

$$f_\varepsilon(v_\varepsilon) = \min\{f_\varepsilon(v) | v \in C^o\}$$

for $\varepsilon > 0$ and $f^* = \min\{f(v) | v \in C\}$. Then

$$0 \le f(v_\varepsilon) - f^* \le \varepsilon m$$

and $f(v_\varepsilon) \to f^*$ as $\varepsilon \to 0$. If (9.22) has a unique optimum v^*, then by using $\varepsilon = \frac{1}{n}$ we obtain a sequence $v_{\frac{1}{n}} \to v^*$ as $n \to \infty$.

We move on to give concrete examples of Theorem 9.21 in action.

9.3.1 Quadratic function with polyhedral constraints

A much used setup in optimization is minimizing a quadratic functions subject to polyhedral constraints. This is the optimization problem

$$\min_{Av \le b} \; v^\top Q v + c^\top v, \tag{9.24}$$

where Q is a $d \times d$ matrix, A is an $m \times d$ matrix, $c \in \mathbb{R}^d$ and $b \in \mathbb{R}^m$.

Certainly the constraints $Av \le b$ define a convex subset of $\mathbb{R}^d$, but the function $v^\top Q v + c^\top v$ is not strictly convex unless Q is positive definite. If Q is not positive semidefinite (9.24) is difficult.

If Q is positive semidefinite, the interior point method outlined above usually works well.

(9.22) EXAMPLE.

The optimization problem (9.2) has the form (9.24), when we put

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$c = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

$$A = \begin{pmatrix} -2x_1 & -2y_1 & -1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & -1 \end{pmatrix}$$

$$b = \begin{pmatrix} -x_1^2 - y_1^2 \\ \vdots \\ -x_n^2 - y_n^2 \end{pmatrix}$$

♠

(9.23) EXERCISE.

Show that the optimization problem in Theorem 9.8 also has the form (9.24) by writing up the matrices Q and A and the vectors c and b for the labeled points $(v_1, y_1), \ldots, (v_m, y_m)$ with $v_i \in \mathbb{R}^d$. ♠

(9.24) EXAMPLE.

Quadratic optimization problems, such as the one in Theorem 9.8, can be solved numerically in python. The code below uses the function `minimize` from `scipy`. It attempts (in general) to solve the optimization problem

$$\min_{\substack{Gx \leq h \\ Ax = b}} \frac{1}{2} x^\top Q x + p^\top x.$$

In the cell below the optimization problem

$$\min_{\substack{x_1 \geq 0,\ x_2 \geq 0 \\ x_1 + x_2 = 1}} \quad 2x_1^2 + x_2^2 + x_1 x_2 + x_1 + x_2$$

has been entered.

python cell — not displayed in the pdf version.

♠

(9.25) EXERCISE.

Take a look at the input format in Example 9.24. Can you tell which optimization problem in this chapter is solved below? Compare the numerical solution with your solution to the exercise below Example 9.9.

python cell — not displayed in the pdf version.

♠

Let us watch Theorem 9.21 in action on the fire station problem of Example 9.3: find the smallest circle enclosing the seven houses

$$(0,0),\quad (2,2),\quad (-3,2),\quad (1,0),\quad (-2,1),\quad (-1,3)\quad \text{and}\quad (0,4).$$

The cell below minimizes the barrier function f_ε from (9.23) for the constraints of (9.2) with

$$\varepsilon = 1,\quad 0.1,\quad 0.01,\quad 0.001,\quad 0.0001,$$

each unconstrained minimization done by `minimize` from `scipy` as in Example 9.24, and each starting from the minimum v_ε found for the previous ε — a *warm start*, which is how interior point methods work in practice. Watch the printed output converge to the exact solution: the circle of radius $\frac{5}{2}$ centered at $\left(-\frac{1}{2},2\right)$, touching the three houses $(2,2)$, $(-3,2)$ and $(1,0)$. The plot shows the shrinking dashed circles closing in on the optimal solid one, and in red the path $\varepsilon \mapsto v_\varepsilon$ — the celebrated central path of interior point methods.

python cell — not displayed in the pdf version.

If you are curious about the real algorithmic machinery — Newton's method from the previous chapter with exact line search applied to f_ε — see Section 10.5.1 of my book Undergraduate Convexity.

(9.26) EXERCISE.

Solve the two optimization problems in each of the Exercises 7.48, 7.49 and 7.50 numerically with `minimize` from `scipy` as in Example 9.24 (a maximum is found by minimizing $-f$). Check the numerical output by actually solving the exercises. ♠

(9.27) EXERCISE.

Compute the best line separating the labeled data

```
((1, 0), +1), ((2, 0), +1), ((3, 0), +1), ((3, 2), +1), ((1, 1), -1), ((2, 2), -1).
```

using the support vector machine cell from Section 9.2. Which of the six points are the support vectors? ♠

The interior point method delivers approximations v_ε marching toward the optimum. The rest of the chapter treats the complementary, exact question: given a point v_0, how do we *recognize* — and certify with a proof — that it is optimal?

9.4 A geometric optimality criterion

Consider the general optimization problem

$$\min_{(x_1,\ldots,x_d)\in C} \quad f(x_1,\ldots,x_d), \tag{9.25}$$

where C is a subset of $\mathbb{R}^d$.

(9.28) PROPOSITION.

Suppose that $C \subseteq \mathbb{R}^d$ is a convex subset and $f : \mathbb{R}^d \to \mathbb{R}$ a differentiable function in (9.25). If $v_0 \in C$ is an optimal solution of (9.25), then

$$\nabla f(v_0)(v - v_0) \geq 0 \qquad \text{for every } v \in C. \tag{9.26}$$

If f in addition is a convex function, then (9.26) implies that v_0 is optimal.

Proof. If v_0 is an optimal solution and $v \in C \setminus \{v_0\}$, then

$$\begin{aligned} 0 &\leq f((1-t)v_0 + tv) - f(v_0) = f(v_0 + t(v - v_0)) - f(v_0) \\ &= t\,(\nabla f(v_0)(v - v_0) + \varepsilon(t(v - v_0))\,|v - v_0|) \end{aligned}$$

for every t with $0 \leq t \leq 1$, where ε denotes the epsilon function in the definition of differentiability (see Definition 7.3). Therefore

$$\nabla f(v_0)(v - v_0) + \varepsilon(t(v - v_0))\,|v - v_0| \geq 0$$

for $0 \leq t \leq 1$. This is only possible if $\nabla f(v_0)(v - v_0) \geq 0$. We have silently applied the convexity of C and the differentiability of f at v_0.

If f in addition is convex and (9.26) holds, then Theorem 8.23 shows that v_0 is an optimal solution.

A nice application of Proposition 9.28 is where the constraints in C are linear. Then you can test if v_0 is an optimal solution by solving the linear program

$$\min \nabla f(v_0) v$$

for $v \in C$. If the optimal value is $\geq \nabla f(v_0) v_0$, then v_0 is an optimal solution.

(9.29) EXAMPLE.

Let us test drive this on the convex optimization problem

$$\min_{\substack{x \geq 0,\ y \geq 0 \\ x+y \leq 2}} f(x,y),$$

where $f(x,y) = (x-2)^2 + (y-1)^2$ and $\nabla f(x,y) = (2x-4,\, 2y-2)$. The constraints are linear: C is the triangle with vertices $(0,0)$, $(2,0)$ and $(0,2)$.

Is the vertex $v_0 = (2,0)$ an optimal solution? Here $\nabla f(v_0) = (0,-2)$, so the linear program to solve is

$$\min_{(x,y) \in C} -2y.$$

A linear function attains its minimum over the triangle at a vertex (this is the content of the exercise below), so we only have to check three points: $-2y$ takes the values 0, 0 and -4 at $(0,0)$, $(2,0)$ and $(0,2)$. The optimal value -4 is *smaller* than $\nabla f(v_0)\, v_0 = 0$, so the test fails — and it fails informatively. For $v = (0,2)$ we have $\nabla f(v_0)(v - v_0) = -4 < 0$, so by Proposition 9.28 the vertex $v_0 = (2,0)$ is *not* optimal: f decreases along the edge from $(2,0)$ toward $(0,2)$.

Let us instead try $v_0 = \left(\frac{3}{2}, \frac{1}{2}\right)$, which lies on the same edge. Now $\nabla f(v_0) = (-1,-1)$ and the linear program is

$$\min_{(x,y) \in C} -x - y$$

with the values 0, -2 and -2 at the three vertices: the optimal value is -2. Since

$$-2 \geq \nabla f(v_0)\, v_0 = -\tfrac{3}{2} - \tfrac{1}{2} = -2,$$

the test succeeds and Proposition 9.28 (here we use that f is convex) certifies that $\left(\frac{3}{2}, \frac{1}{2}\right)$ is an optimal solution — no further theory needed, just one linear program.

The cell below shows both verdicts geometrically. The level sets of f are circles around the unconstrained minimum $(2,1)$. The dotted red circle is the level set through $(2,0)$: it dips into the triangle, so C contains points with smaller f-value and $(2,0)$ cannot be optimal — the red arrow is the descent along the edge found by the failed test. The green circle is the level set through $\left(\frac{3}{2}, \frac{1}{2}\right)$: it only *touches* the triangle. It is tangent to the edge $x+y=2$, and the tangent line is exactly the hyperplane $\nabla f(v_0)\, v = \nabla f(v_0)\, v_0$ from Proposition 9.28 — compare with the picture for a curved boundary in the next example.

python cell — not displayed in the pdf version.

♠

(9.30) EXAMPLE.

Proposition 9.28 also applies when the boundary of C is curved, as in the optimization problem

$$\min_{x^2+y^2 \leq \frac{1}{2}} \quad (x+1)^2 + (y+1)^2.$$

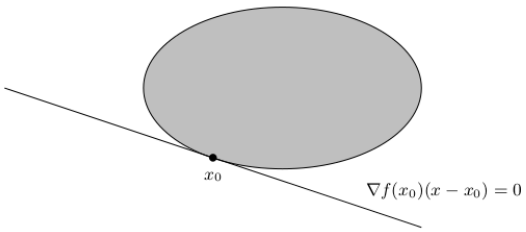
Here Proposition 9.28 shows that $v_0 = \left(-\frac{1}{2}, -\frac{1}{2}\right)$ is optimal: with $\nabla f(v_0) = (1,1)$, the hyperplane

$$\nabla f(v_0)\, v = \nabla f(v_0)\, v_0,$$

which is the line $x+y=-1$, touches the boundary of

$$C = \left\{(x,y) \in \mathbb{R}^2 \,\middle|\, x^2+y^2 \leq \tfrac{1}{2}\right\}$$

exactly at v_0, and C lies on the side where $\nabla f(v_0)(v-v_0) \geq 0$ — as sketched below.



♠

(9.31) EXERCISE.

Sketch how Proposition 9.28 applies to show that an optimum in a linear programming problem

$$\min_{A\begin{pmatrix} x \\ y \end{pmatrix} \leq b} \quad cx + dy$$

in the plane $\mathbb{R}^2$ always can be found in a vertex. ♠

(9.32) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable convex function and

$$S = \{(x,y) | -1 \leq x \leq 2, -1 \leq y \leq 1\}.$$

Suppose that $\nabla f(v_0) = (1,0)$ for $v_0 = (-1, \frac{1}{2})$. Prove that v_0 is a minimum for f defined on S. ♠

(9.33) EXERCISE.

Guess the solution to the optimization problem

$$\min \left\{ (x-5)^2 + (y-5)^2 \,\middle|\, x \geq 0, y \geq 0, x^2 + y^2 \leq 25 \right\}.$$

Show that your guess was correct! ♠

9.5 The KKT conditions

The KKT in the title of this section is short for Karush-Kuhn-Tucker.

We will limit ourselves to a convex optimization problem of the form

$$\min_{(x_1,\ldots,x_d) \in C} f(x_1,\ldots,x_d), \tag{9.27}$$

where C is defined by the differentiable convex functions $g_i : \mathbb{R}^d \to \mathbb{R}$ for $i = 1,\ldots,m$ as

$$C = \{v \in \mathbb{R}^d \mid g_1(v) \leq 0, \ldots, g_m(v) \leq 0\}$$

and $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function.

To the optimization problem (9.27) we associate the (famous) Karush-Kuhn-Tucker (KKT) conditions:

$$\lambda_1 \geq 0, \ldots, \lambda_m \geq 0 \tag{9.28}$$

$$\tag{9.29}$$

$$g_1(v_0) \leq 0, \ldots, g_m(v_0) \leq 0 \tag{9.30}$$

$$\tag{9.31}$$

$$\lambda_1 g_1(v_0) = 0, \ldots, \lambda_m g_m(v_0) = 0 \tag{9.32}$$

$$\tag{9.33}$$

$$\nabla f(v_0) + \lambda_1 \nabla g_1(v_0) + \cdots + \lambda_m \nabla g_m(v_0) = 0. \tag{9.34}$$

Notice that the KKT conditions consist of $2m$ inequalities and $m+d$ equations in the $m+d$ unknowns $\lambda_1,\ldots,\lambda_m, v_0 = (x_1,\ldots,x_d)$. The KKT conditions form a surprising theoretical foundation for optimization problems of the type in (9.27). You should take a peek back to the theory of Lagrange multipliers in section 7.10 and compare with (9.28).

(9.34) EXAMPLE.

The KKT conditions associated with the convex optimization problem in Example 9.1 are

$$\begin{aligned}
\lambda_1, \lambda_2, \lambda_3, \lambda_4 &\geq 0 \\
-x - y + 2 &\leq 0 \\
y - 2 &\leq 0 \\
x - 3 &\leq 0 \\
-y + 1 &\leq 0 \\
\lambda_1(-x - y + 2) &= 0 \\
\lambda_2(y - 2) &= 0 \\
\lambda_3(x - 3) &= 0 \\
\lambda_4(-y + 1) &= 0 \\
2x - \lambda_1 + \lambda_3 &= 0 \\
2y - \lambda_1 + \lambda_2 - \lambda_4 &= 0.
\end{aligned}$$

♠

(9.35) EXERCISE.

Verify that the KKT conditions of the optimization problem in Example 9.1 are the ones given in Example 9.34. ♠

To state our main theorem we need a definition.

(9.36) DEFINITION.

The optimization problem (9.27) *is called* strictly feasible *if there exists* $z_0 \in \mathbb{R}^d$ *with*

$$\begin{aligned}
g_1(z_0) &< 0 \\
&\vdots \\
g_m(z_0) &< 0.
\end{aligned}$$

Below is the main result in our limited convex setting. We will not go into the proof, which can be found in my book Undergraduate Convexity.

(9.37) THEOREM.

(i) Let v_0 *be an optimal solution of* (9.27)*. If* (9.27) *is strictly feasible, then the KKT conditions are satisfied at* v_0 *for suitable* $\lambda_1, \ldots, \lambda_m$*.*

(ii) If the KKT conditions are satisfied at $z \in \mathbb{R}^d$ *for some* $\lambda_1, \ldots, \lambda_m$*, then* z *is an optimal solution to* (9.27)*.*

(9.38) EXAMPLE.

Let us now touch base with a rather simple example. Consider the optimization problem

$$\min_{x\in[1,2]} \; x. \tag{9.35}$$

Here $f(x)=x, g_1(x)=-x+1$ and $g_2(x)=x-2$ in (9.27). Therefore the KKT conditions in (9.28) are

$$\lambda_1 \geq 0 \tag{9.36}$$
$$\lambda_2 \geq 0 \tag{9.37}$$
$$-x+1 \leq 0 \tag{9.38}$$
$$x-2 \leq 0 \tag{9.39}$$
$$\lambda_1(-x+1)=0 \tag{9.40}$$
$$\lambda_2(x-2)=0 \tag{9.41}$$
$$1-\lambda_1+\lambda_2=0. \tag{9.42}$$

Before even thinking about reading on, you should attempt to find a solution x,λ_1,λ_2 to the above KKT conditions (inequalities) and then verify using Theorem 9.37(ii) that x is optimal. Also, try only using Theorem 9.37(i) and (9.36) to show that $x=2$ is not a solution to (9.35).

♠

(9.39) QUIZ.

quiz — not displayed in the pdf version.

♠

(9.40) EXERCISE.

Give an example of a convex optimization problem as in (9.27), which is not strictly feasible and with an optimal solution v_0 that does not satisfy the KKT conditions. Such an example shows that strict feasibility is necessary in Theorem 9.37(i). ♠

9.5.1 Strategy

A general strategy for finding solutions to the KKT conditions in (9.28) is zooming in on (the Lagrange multipliers) $\lambda_1,\ldots,\lambda_m$ testing each of them for the two cases $\lambda_i=0$ and $\lambda_i>0$.

One important point, which you can read from (9.28), is that $g_i(v_0)=0$ if $\lambda_i>0$. To further elaborate, if $\lambda_i>0$, then an optimal solution must satisfy $g_i(v_0)=0$.

(9.41) EXERCISE.

So where exactly in (9.28) is the above claim verified? ♠

The condition $\lambda_i=0$ simplifies the equations

$$\nabla f(v_0)+\lambda_1\nabla g_1(v_0)+\cdots+\lambda_m\nabla g_m(v_0)=0$$

in (9.28).

In principle to solve the KKT conditions, one has to try out all the 2^m possibilities coming from $\lambda_i = 0$ or $\lambda_i > 0$ for $i = 1, \ldots, m$.

(9.42) EXERCISE.

Why 2^m possibilities above? ♠

(9.43) EXERCISE.

How do you solve the optimization problem (or decide there is no solution) if $\lambda_1 = \cdots = \lambda_m = 0$? ♠

9.5.2 Example

Let C denote the set (see Figure 9.2) of points $(x,y) \in \mathbb{R}^2$ with

$$\begin{aligned} x^2 + 2y^2 &\leq 1 \\ x + y &\leq 1 \\ y &\leq x. \end{aligned}$$

We will illustrate the mechanics of solving the KKT conditions in finding an optimal solution for

$$\min_{(x,y)\in C} \quad x + 3y. \tag{9.43}$$

(9.44) FIGURE.

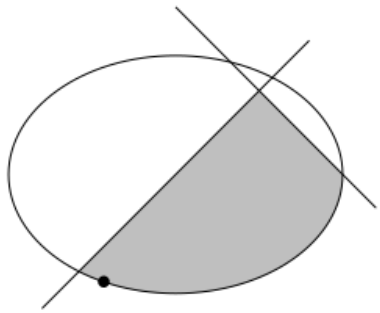


Figure 9.2: The convex set C with optimal solution for (9.43) marked.

Putting

$$\begin{aligned} g_1(x,y) &= x^2 + 2y^2 - 1 \\ g_2(x,y) &= x + y - 1 \\ g_3(x,y) &= y - x \end{aligned}$$

and $f(x,y) = x + 3y$, we are in a position to apply Theorem 9.37, since g_1, g_2, g_3 are convex functions and $g_1(z_0) < 0, g_2(z_0) < 0, g_3(z_0) < 0$ for $z_0 = (0, -\frac{1}{2})$. This means that an optimal solution of (9.43) satisfies

the KKT conditions. The same theorem tells us that the v_0 in a solution of the KKT conditions is an optimal solution (here we also use that f is a convex function). The full set of KKT conditions in $x, y, \lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$ are

$$\begin{aligned} x^2+2y^2-1 &\leq 0 \\ x+y-1 &\leq 0 \\ y-x &\leq 0 \\ \lambda_1, \lambda_2, \lambda_3 &\geq 0 \\ \lambda_1(x^2+2y^2-1) &= 0 \\ \lambda_2(x+y-1) &= 0 \\ \lambda_3(-x+y) &= 0 \\ 1+2\lambda_1 x+\lambda_2-\lambda_3 &= 0 \\ 3+4\lambda_1 y+\lambda_2+\lambda_3 &= 0. \end{aligned}$$

A strategy for finding a solution to the KKT conditions is trying (the eight) different combinations of strict inequalities in $\lambda_1, \lambda_2, \lambda_3 \geq 0$. You can see from the last two equations that $\lambda_1 = 0$ is impossible. The condition $\lambda_1 > 0$ shows that an optimal solution has to occur on the lower arc in Figure 9.2. If $\lambda_3 > 0$, then $x = y$ and $\lambda_2 = 1+3\lambda_3 > 0$ by the last two equations. This implies $x = y = \frac{1}{2}$ violating $x^2+2y^2-1 = 0$. Therefore $\lambda_3 = 0$. If $\lambda_2 > 0$, then $y = 1-x$ and $5+4\lambda_1+3\lambda_2 = 0$ by $\lambda_3 = 0$ and the last two equations. Therefore $\lambda_2 = 0$. So we are left with the case $\lambda_1 > 0$ and $\lambda_2 = \lambda_3 = 0$ giving

$$x = -\frac{1}{2\lambda_1} \quad \text{and} \quad y = -\frac{3}{4\lambda_1}.$$

Inserting this into $x^2+2y^2-1 = 0$ we end up with (see Figure 9.2)

$$\lambda_1 = \frac{\sqrt{11}}{2\sqrt{2}}, \quad x = -\sqrt{\frac{2}{11}} \quad \text{and} \quad y = -\frac{3}{\sqrt{22}}.$$

Theorem 9.37 is beautiful mathematics. Going through the KKT conditions as above can be quite lengthy if not impossible in practice. As we have seen, there are other methods for (at least) approximating an optimal solution.

9.5.3 Solving the KKT conditions by computer

Lengthy for a human — but look again at the strategy: 2^m cases, and each case is nothing but a system of equations together with some sign checks. That is a job for a computer, and `sympy` can do it with exact arithmetic. The program below is the strategy of this section made executable. For each of the 2^m cases it decides, for every constraint, between $\lambda_i = 0$ and $g_i = 0$ (complementary slackness!), solves the resulting equations together with stationarity

$$\nabla f(v_0) + \lambda_1 \nabla g_1(v_0) + \cdots + \lambda_m \nabla g_m(v_0) = 0,$$

and keeps only solutions passing the remaining KKT inequalities $\lambda_i \geq 0$ and $g_i(v_0) \leq 0$.

python cell — not displayed in the pdf version.

Compare with the hand computation above: seven of the eight cases die — for exactly the reasons we found — and the surviving case $\lambda_1 > 0$, $\lambda_2 = \lambda_3 = 0$ delivers

$$x = -\frac{\sqrt{22}}{11} = -\sqrt{\tfrac{2}{11}}, \qquad y = -\frac{3\sqrt{22}}{22} = -\frac{3}{\sqrt{22}}, \qquad \lambda_1 = \frac{\sqrt{22}}{4} = \frac{\sqrt{11}}{2\sqrt{2}},$$

the same numbers in a different radical costume, together with the optimal value $f = -\sqrt{22}/2$. The program is not tied to this example: replace f and the list g (and the symbols) with your own convex optimization problem and it will grind through the cases for you — bearing in mind that the number of cases doubles with every constraint, which is precisely why practical solvers use the barrier and descent methods of the earlier sections instead. You are welcome to check your hand computations in the exercises below against it.

9.5.4 Closing the circle: KKT and the optimal hyperplane

This chapter opened with the optimal separating hyperplane and ends with certificates of optimality — and the two meet. The dual problem (9.13) was presented without derivation, and its two most mysterious ingredients were the linear combination (9.12) and the constraint $\Lambda^\top Y = 0$. Both fall out of the KKT conditions, and for the two points that Example 9.12 treated with the dual, every step can be done by hand.

(9.45) EXAMPLE.

Return to the two points of Example 9.9: $v_1 = (1,1)^\top$ with label $y_1 = 1$ and $v_2 = (2,2)^\top$ with label $y_2 = -1$. This time we keep all three variables $\alpha = (\alpha_1, \alpha_2)^\top$ and β of Theorem 9.8 and treat the problem as (9.27) with

$$f(\alpha, \beta) = |\alpha|^2 = \alpha_1^2 + \alpha_2^2$$

and the two convex constraint functions

$$g_1(\alpha, \beta) = 1 - (\alpha^\top v_1 + \beta) \le 0$$
$$g_2(\alpha, \beta) = 1 + (\alpha^\top v_2 + \beta) \le 0.$$

The problem is strictly feasible — the optimal line scaled up, say $\alpha = (-2,-2)^\top$ and $\beta = 6$, satisfies both constraints strictly — so by Theorem 9.37 the KKT conditions characterize the optimum exactly.

The gradients with respect to the three variables $(\alpha_1, \alpha_2, \beta)$ are

$$\nabla f = (2\alpha_1, 2\alpha_2, 0), \qquad \nabla g_1 = (-v_1^\top, -1), \qquad \nabla g_2 = (v_2^\top, 1),$$

so the last KKT condition $\nabla f + \lambda_1 \nabla g_1 + \lambda_2 \nabla g_2 = 0$ splits into two parts. The α-part reads

$$2\alpha = \lambda_1 v_1 - \lambda_2 v_2 = \lambda_1 y_1 v_1 + \lambda_2 y_2 v_2$$

— the mysterious linear combination (9.12)! That the normal vector is built from the data points, weighted by multipliers and labels, is not an assumption of the dual; it is forced by the KKT conditions. The β-part reads

$$-\lambda_1 + \lambda_2 = 0, \qquad \text{i.e.} \qquad \lambda_1 y_1 + \lambda_2 y_2 = 0,$$

which is exactly the dual constraint $\Lambda^\top Y = 0$.

Next, complementary slackness. The multipliers cannot vanish: if $\lambda_1 = \lambda_2 = 0$, the α-part would give $\alpha = 0$, and then $g_1 \le 0$ and $g_2 \le 0$ would demand $\beta \ge 1$ and $\beta \le -1$ at once. So $\lambda_1 = \lambda_2 > 0$, and $\lambda_i g_i(v_0) = 0$ forces *both* constraints to hold with equality:

$$\alpha^\top v_1 + \beta = 1 \qquad \text{and} \qquad \alpha^\top v_2 + \beta = -1.$$

These are the support vector equations. The claim that both dashed lines carry data points — argued after Theorem 9.8 with a paragraph of shifting and rescaling — is here simply complementary slackness.

The rest is arithmetic. Write $\lambda = \lambda_1 = \lambda_2$. The α-part gives $\alpha = \frac{\lambda}{2}(v_1 - v_2) = -\frac{\lambda}{2}(1,1)^\top$, and subtracting the two support vector equations gives $\alpha^\top(v_1 - v_2) = 2$, that is $\frac{\lambda}{2}|v_1 - v_2|^2 = 2$, so $\lambda = 2$. Then

$$\alpha = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \qquad \text{and} \qquad \beta = 1 - \alpha^\top v_1 = 3,$$

and by Theorem 9.37(ii) the line $x + y = 3$ is *certifiably* optimal — the same answer as the dual computation in Example 9.12, now with a certificate attached. (The multiplier $\lambda = 2$ is twice the dual variable $\lambda = 1$ found in Example 9.12; the factor is a bookkeeping constant coming from the objective $|\alpha|^2$ versus $\frac{1}{2}|\alpha|^2$, absorbed by (9.12).) ♠

Every mystery of the dual has now been witnessed at small scale: the linear combination (9.12) is the α-part of KKT stationarity, the constraint $\Lambda^\top Y = 0$ is its β-part, and the support vectors are complementary slackness in action. The general derivation of (9.13) — the one we referred to the literature for — is exactly the computation above with m points in place of two.

9.6 Optimization exercises

Below are some exercises especially related to the KKT conditions. In some of the exercises the minimization problem

$$\min_{(x_1,\ldots,x_d)\in C} f(x_1,\ldots,x_d),$$

is denoted

$$\min\{f(x_1,\ldots,x_d) \mid (x_1,\ldots,x_d) \in C\}.$$

This should cause no confusion.

(9.46) EXERCISE.

Consider the optimization problem

$$\min_{\substack{x^2+y^2\le 1\\ (x+1)^2+(y-1)^2\le 1}} -x+y \tag{9.44}$$

(a) Show that (9.44) is a convex optimization problem.

(b) Sketch the set of constraints in $\mathbb{R}^2$ and show that $\left(-\frac{1}{2},\frac{1}{2}\right)$ cannot be an optimal solution to (9.44).

(c) Write up the KKT conditions for (9.44) and explain theoretically (without actually solving them) why they must have a solution.

(d) Now solve (9.44). Is the solution unique?

♠

(9.47) EXERCISE.

Consider the function $f:\mathbb{R}^3\to\mathbb{R}$ given by

$$f(x_1,x_2,x_3) = 2x_1^2+3x_2^2+4x_3^2.$$

(a) Show that f is strictly convex.

(b) Let $S\subseteq\mathbb{R}^3$ denote the subset of points $(x_1,x_2,x_3)\in\mathbb{R}^3$ satisfying

$$\begin{aligned} x_1+x_2+x_3 &\ge 1\\ x_1+2x_2+3x_3 &\le 5 \end{aligned}$$

Show that S is a closed convex subset.

(c) Solve the optimization problem

$$\min_{(x_1,x_2,x_3)\in S} f(x_1,x_2,x_3) \tag{9.45}$$

♠

(9.48) EXERCISE.

Let $S\subseteq B\subseteq\mathbb{R}^3$, where

$$\begin{aligned} S &= \{(x,y,z)\in\mathbb{R}^3 \mid x^2+y^2+z^2=1\} \qquad \text{and}\\ B &= \{(x,y,z)\in\mathbb{R}^3 \mid x^2+y^2+z^2\le 1\}. \end{aligned}$$

(a) Why does the optimization problem

$$\min_{(x,y,z)\in S} \quad x+2y+z \tag{9.46}$$

have a solution?

(b) Find all optimal solutions to (9.46).

(c) Let $a,b,c \in \mathbb{R}$, where at least one of a,b,c is non-zero. Show that an optimal solution to

$$\min_{(x,y,z)\in B} \quad ax+by+cz$$

belongs to S.

♠

(9.49) EXERCISE.

Let

$$S = \left\{ (x,y) \in \mathbb{R}^2 \middle| \begin{array}{rcrcl} -x & - & y & \leq & 0 \\ 2x & - & y & \leq & 1 \\ -x & + & 2y & \leq & 1 \end{array} \right\}$$

1. Use the KKT conditions to solve the minimization problem

$$\min\{-x-4y|(x,y) \in S\}.$$

2. Use the KKT conditions to solve the minimization problem

$$\min\{x+y|(x,y) \in S\}.$$

♠

(9.50) EXERCISE.

Solve the optimization problem

$$\min \left\{ x^2+2y^2+3z^2-2xz-xy \middle| \begin{array}{ccc} 2x^2+y^2+z^2 & \leq & 4 \\ 1 & \geq & x+y+z \end{array} \right\}.$$

♠

(9.51) EXERCISE.

Let $S = \{(x,y)|2x^2+y^2 \leq 3, x^2+2y^2 \leq 3\}$ and $f(x,y) = (x-4)^2+(y-4)^2$.

1. State the KKT conditions for $\min\{f(x,y)(x,y) \in S\}$ for $(x,y) = (1,1)$.

2. Suppose now that $g(x,y) = (x-a)^2+(y-b)^2$. For which a and b does $\min\{g(x,y)|(x,y) \in S\}$ have optimum in $(1,1)$? State the KKT conditions when $(a,b) = (1,1)$.

♠

(9.52) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x,y) = (x-1)^2 + (y-1)^2 + 2xy.$$

1. Show that f is a convex function.
2. Find $\min\{f(x,y)|(x,y) \in \mathbb{R}^2\}$. Is this minimum unique? Is f a strictly convex function.

   Let

   $$S = \{(x,y) \in \mathbb{R}^2 | x+y \leq 0, \quad x-y \leq 0\}.$$

3. Apply the KKT-conditions to decide if $(-1,-1)$ is an optimal solution to

   $$\min\{f(x,y)|(x,y) \in S\}.$$

4. Find

   $$m = \min\{f(x,y)|(x,y) \in S\}$$

   and

   $$\{(x,y) \in \mathbb{R}^2 | f(x,y) = m\}.$$

♠

(9.53) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x,y) = x^2 + y^2 - e^{x-y-1}$$

and let

$$C = \{(x,y)|x-y \leq 0\}.$$

1. Show that $f : \mathbb{R}^2 \to \mathbb{R}$ is not a convex function.
2. Show that f is a convex function on the open subset

   $$\{(x,y) \in \mathbb{R}^2 | x-y < \tfrac{1}{2}\}$$

   and conclude that f is convex on C.
3. Show that $v = (0,0)$ is an optimal solution for the optimization problem $\min\{f(v)|v \in C\}$. Is v a unique optimal solution here?

♠

(9.54) EXERCISE.

Let $f : \mathbb{R}^4 \to \mathbb{R}$ be given by

$$f(x_1,x_2,x_3,x_4) = (x_1 - x_3)^2 + (x_2 - x_4)^2$$

and $C \subseteq \mathbb{R}^4$ by

$$C = \{(x_1,x_2,x_3,x_4) \in \mathbb{R}^4 | x_1^2 + (x_2-2)^2 \leq 1, x_3 - x_4 \geq 0\}.$$

1. Show that f is a convex function. Is f strictly convex?
2. Show that C is a convex subset of $\mathbb{R}^4$.
3. Does there exist an optimal point $v = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4$ for the minimization problem
$$\min_{v \in C} f(v)$$
with $x_3 = x_4 = 0$?
4. Does there exist an optimal point $v = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4$ for the minimization problem
$$\min_{v \in C} f(v)$$
with $x_3 = x_4 = 1$?

♠

(9.55) EXERCISE.

Let
$$f(x,y) = (x-1)^2 + y^2$$
and
$$C = \{(x,y) \in \mathbb{R}^2 | -1 \leq x \leq 0,\ -1 \leq y \leq 1\}.$$
Solve the optimization problem
$$\min\{f(x,y) | (x,y) \in C\}.$$

♠

(9.56) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by
$$f(x,y) = \tfrac{1}{2}x^2 + y^2 - 2y + 2.$$
Below, the minimization problem
$$\min\{f(x,y) | (x,y) \in S\} \tag{9.47}$$
is analyzed for various subsets $S \subseteq \mathbb{R}^2$.

1. Show that f is a convex function
2. Let
$$S = \{(x,y) \in \mathbb{R}^2 | -x + 2y \leq 1\}.$$
Show that $(-1,0) \in S$ cannot be an optimal solution to (9.47). Find an optimal solution to (9.47).
3. Find an optimal solution in (9.47) for
$$S = \{(x,y) \in \mathbb{R}^2 | -x + 2y \geq 1\}.$$
4. Are the optimal solutions in 2 and 3 unique?

♠

(9.57) EXERCISE.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by

$$f(x,y) = 2x^2 + 3x + y^2 + y.$$

1. Show that f is a convex function and solve the minimization problem $\min\{f(x,y)|x,y \in \mathbb{R}\}$.
   Now let
   $$S = \left\{ (x,y) \in \mathbb{R}^2 \middle| \begin{array}{rcl} x^2 + (y+1)^2 & \leq & 1 \\ y - x & \leq & 0 \end{array} \right\}$$
   and consider the minimization problem (P) given by
   $$\min\{f(x,y)|(x,y) \in S\}.$$
2. Show using the KKT conditions that $(0,0)$ is not optimal for (P).
3. Find an optimal solution for (P). Is it unique?

♠