Foreword

Machine learning is the latest in a long line of attempts to capture human knowledge and reasoning into a form that is suitable for constructing machines and engineering automated systems. As machine learning becomes more ubiquitous and its software packages become easier to use it is natural and desirable that the low-level technical details are abstracted away and hidden from the practitioner. However, this brings with it the danger that a practitioner becomes unaware of the design decisions and, hence, the limits of machine learning algorithms. The enthusiastic practitioner who is interested to learn more about the magic behind successful machine learning algorithms currently faces a daunting set of pre-requisite knowledge:

- Programming languages and data analysis tools
- Large-scale computation and the associated frameworks
- Mathematics and statistics and how machine learning builds on it

At universities, introductory courses on machine learning tend to spend early parts of the course covering some of these pre-requisites. For historical reasons, courses in machine learning tend to be taught in the computer science department, where students are often trained in the first two areas of knowledge, but not so much in mathematics and statistics. Current machine learning textbooks try to squeeze in one or two chapters of background mathematics, either at the beginning of the book or as appendices. This book brings the mathematical foundations of basic machine learning concepts to the fore and collects the information in a single place.

Why Another Book on Machine Learning?

Machine learning builds upon the language of mathematics to express concepts that seem intuitively obvious but which are surprisingly difficult to formalize. Once properly formalized we can then use the tools of mathematics to derive the consequences of our design choices. This allows us to gain insights into the task we are solving and also the nature of intelligence. One common complaint of students of mathematics around the globe is that the topics covered seem to have little relevance to practical problems. We believe that machine learning is an obvious and direct motivation for people to learn mathematics.

"Math is linked in the popular mind with phobia and anxiety. You'd think we're discussing spiders." (Strogatz, 2014)

This book is intended to be a guidebook to the vast mathematical literature that forms the foundations of modern machine learning. We motivate the need for mathematical concepts by directly pointing out their usefulness in the context of fundamental machine learning problems. In the interest of keeping the book short, many details and more advanced concepts have been left out. Equipped with the basic concepts presented here, and how they fit into the larger context of machine learning, the reader can find numerous resources for further study, which we provide at the end of the respective chapters. For readers with a mathematical background, this book provides a brief but precisely stated glimpse of machine learning. In contrast to other books that focus on methods and models of machine learning (MacKay, 2003b; Bishop, 2006; Alpaydin, 2010; Rogers and Girolami, 2016; Murphy, 2012; Barber, 2012; Shalev-Shwartz and Ben-David, 2014) or programmatic aspects of machine learning (Müller and Guido, 2016; Raschka and Mirjalili, 2017; Chollet and Allaire, 2018) we provide only four representative examples of machine learning algorithms. Instead we focus on the mathematical concepts behind the models themselves, with the intent of illuminating their abstract beauty. We hope that all readers will be able to gain a deeper understanding of the basic questions in machine learning and connect practical questions arising from the use of machine learning with fundamental choices in the mathematical model.

Who is the Target Audience?

As applications of machine learning become widespread in society we believe that everybody should have some understanding of its underlying principles. This book is written in an academic mathematical style, which enables us to be precise about the concepts behind machine learning. We encourage readers unfamiliar with this seemingly terse style to persevere and to keep the goals of each topic in mind. We sprinkle comments and remarks throughout the text, in the hope that it provides useful guidance with respect to the big picture. The book assumes the reader to have mathematical knowledge commonly covered in high-school mathematics and physics. For example, the reader should have seen derivatives and integrals before, and geometric vectors in two or three dimensions. Starting from there we generalize these concepts. Therefore, the target audience of the book includes undergraduate university students, evening learners and people who participate in online machine learning courses.

In analogy to music, there are three types of interaction, which people have with machine learning:

Astute Listener

The democratization of machine learning by the provision of open-source software, online tutorials, and cloud-based tools allows users to not worry about the nitty gritty details of pipelines. Users can focus on extracting

insights from data using off-the-shelf tools. This enables non-tech savvy domain experts to benefit from machine learning. This is similar to listening to music; the user is able to choose and discern between different types of machine learning, and benefits from it. More experienced users are like music critics, asking important questions about the application of machine learning in society such as ethics, fairness, and privacy of the individual. We hope that this book provides a framework for thinking about the certification and risk management of machine learning systems, and allow them to use their domain expertise to build better machine learning systems.

Experienced Artist

Skilled practitioners of machine learning are able to plug and play different tools and libraries into an analysis pipeline. The stereotypical practitioner would be a data scientist or engineer who understands machine learning interfaces and their use cases, and is able to perform wonderful feats of prediction from data. This is similar to virtuosos playing music, where highly skilled practitioners can bring existing instruments to life, and bring enjoyment to their audience. Using the mathematics presented here as a primer, practitioners would be able to understand the benefits and limits of their favorite method, and to extend and generalize existing machine learning algorithms. We hope that this book provides the impetus for more rigorous and principled development of machine learning methods.

Fledgling Composer

As machine learning is applied to new domains, developers of machine learning need to develop new methods and extend existing algorithms. They are often researchers who need to understand the mathematical basis of machine learning and uncover relationships between different tasks. This is similar to composers of music who, within the rules and structure of musical theory, create new and amazing pieces. We hope this book provides a high-level overview of other technical books for people who want to become composers of machine learning. There is a great need in society for new researchers who are able to propose and explore novel approaches for attacking the many challenges of learning from data.

Contents

1

Introduction and Motivation

1.1 Finding Words for Intuitions

Machine learning is about designing algorithms that learn from data. The goal is to find good models that generalize well to future data. The challenge is that the concepts and words are slippery, and a particular component of the machine learning system can be abstracted to different mathematical concepts. For example, the word "algorithm" is used in at least two different senses in the context of machine learning. In the first sense, we use the phrase "machine learning algorithm" to mean a system that makes predictions based on input data. We refer to these algorithms as *predictors*. In the second sense, we use the exact same phrase "machine learning algorithm" to mean a system that adapts some internal parameters of the predictor so that it performs well on future unseen input data. Here we refer to this adaptation as *training* a predictor.

predictors

training

The first part of this book describes the mathematical concepts and foundations needed to talk about the three main components of a machine learning system: data, models, and learning. We will briefly outline these components here, and we will revisit them again in Chapter 8 once we have the mathematical language under our belt. Adding to the challenge is the fact that the same English word could mean different mathematical concepts, and we can only work out the precise meaning via the context. We already remarked about the overloaded use of the word "algorithm", and the reader will be faced with other such phrases. We advise the reader to use the idea of "type checking" from computer science and apply it to machine learning concepts. Type checking allows the reader to sanity check whether the equation that they are considering contains inputs and outputs of the correct type, and whether they are mixing different types of objects.

While not all data is numerical it is often useful to consider data in a number format. In this book, we assume that the *data* has already been appropriately converted into a numerical representation suitable for reading into a computer program. In this book, we think of data as vectors. As another illustration of how subtle words are, there are three different ways to think about vectors: a vector as an array of numbers (a computer science view), a vector as an arrow with a direction and magnitude (a

data

data as vectors

physics view), and a vector as an object that obeys addition and scaling (a mathematical view).

model

What is a *model*? Models are simplified versions of reality, which capture aspects of the real world that are relevant to the task. Users of the model need to understand what the model does not capture, and hence obtain an appreciation of the limitations of it. Applying models without knowing their limitations is like driving a vehicle without knowing whether it can turn left or not. Machine learning algorithms adapt to data, and therefore their behavior will change as it learns. Applying machine learning models without knowing their limitations is like sitting in a self-driving vehicle without knowing whether it has encountered enough left turns during its training phase. In this book, we use the word "model" to distinguish between two schools of thought about the construction of machine learning predictors: the probabilisitic view and the optimization view. The reader is referred to Domingos (2012) for a more general introduction to the five schools of machine learning.

learning

We now come to the crux of the matter, the *learning* component of machine learning. Assume we have a way to represent data as vectors and that we have an appropriate model. We are interested in training our model based on data so that it performs well on unseen data. Predicting well on data that we have already seen (training data) may only mean that we found a good way to memorize the data. However, this may not generalize well to unseen data, and in practical applications we often need to expose our machine learning system to situations that it has not encountered before. We use numerical methods to find good parameters that "fit" the model to data, and most training methods can be thought of as an approach analogous to climbing a hill to reach its peak. The peak of the hill corresponds to a maximization of some desired performance measure. The challenge is to design algorithms that learn from past data but generalizes well.

Let us summarize the main concepts of machine learning:

- We use domain knowledge to represent data as vectors.
- We choose an appropriate model, either using the probabilisitic or optimization view.
- We learn from past data by using numerical optimization methods with the aim that it performs well on unseen data.

1.2 Two Ways to Read this Book

We can consider two strategies for understanding the mathematics for machine learning:

- Building up the concepts from foundational to more advanced. This is often the preferred approach in more technical fields, such as mathematics. This strategy has the advantage that the reader at all times is

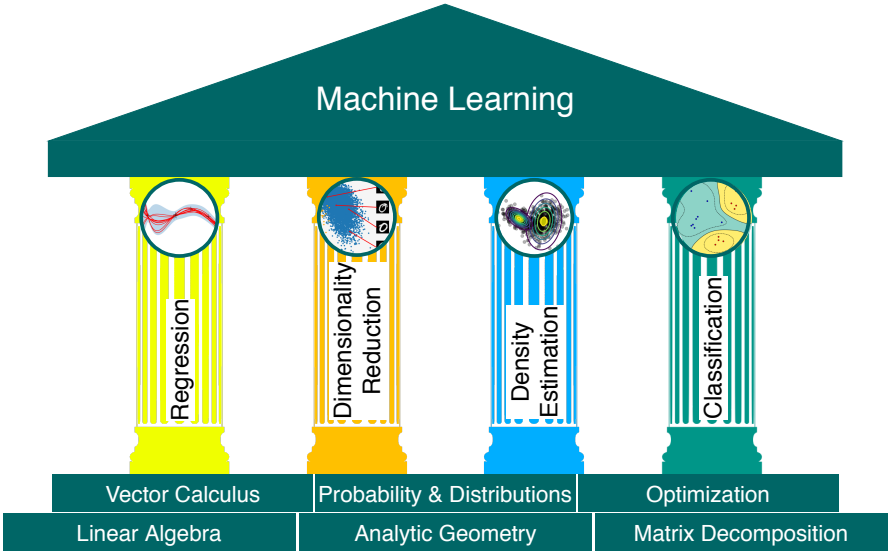


Figure 1.1 The foundations and four pillars of machine learning.

able to rely on their previously learned definitions, and there are no murky hand-wavy arguments that the reader needs to take on faith. Unfortunately, for a practitioner many of the foundational concepts are not particularly interesting by themselves, and the lack of motivation means that most foundational definitions are quickly forgotten.

- Drilling down from practical needs to more basic requirements. This goal-driven approach has the advantage that the reader knows at all times why they need to work on a particular concept, and there is a clear path of required knowledge. The downside of this strategy is that the knowledge is built on shaky foundations, and the reader has to remember a set of words for which they do not have any way of understanding.

This book is split into two parts, where Part I lays the mathematical foundations and Part II applies the concepts from Part I to a set of basic machine learning problems, which form four pillars of machine learning as illustrated in Figure 1.1.

Part I is about Mathematics

We represent numerical data as vectors and represent a table of such data as a matrix. The study of vectors and matrices is called *linear algebra*, which we introduce in Chapter 2. The collection of vectors as a matrix is also described there. Given two vectors, representing two objects in the real world, we want to be able to make statements about their similarity. The idea is that vectors that are similar should be predicted to have similar outputs by our machine learning algorithm (our predictor). To formalize the idea of similarity between vectors, we need to introduce operations

linear algebra

that take two vectors as input and return a numerical value representing their similarity. This construction of similarity and distances is called *analytic geometry* and is discussed in Chapter 3. In Chapter 4, we introduce some fundamental concepts about matrices and *matrix decomposition*. It turns out that operations on matrices are extremely useful in machine learning, and we use them for representing data as well as for modeling.

analytic geometry

matrix decomposition

We often consider data to be noisy observations of some true underlying signal, and hope that by applying machine learning we can identify the signal from the noise. This requires us to have a language for quantifying what noise means. We often would also like to have predictors that allow us to express some sort of uncertainty, e.g., to quantify the confidence we have about the value of the prediction for a particular test data point. Quantification of uncertainty is the realm of *probability theory* and is covered in Chapter 6. Instead of considering a predictor as a single function, we could consider predictors to be probabilistic models, i.e., models describing the distribution of possible functions.

probability theory

To apply hill-climbing approaches for training machine learning models, we need to formalize the concept of a gradient, which tells us the direction which to search for a solution. This idea of the direction to search is formalized by *calculus*, which we present in Chapter 5. How to use a sequence of these search directions to find the top of the hill is called *optimization*, which we introduce in Chapter 7.

calculus

optimization

It turns out that the mathematics for discrete categorical data is different from the mathematics for continuous real numbers. Most of machine learning assumes continuous variables, and except for Chapter 6 the other chapters in Part I of the book only discuss continuous variables. However, for many application domains, data is categorical in nature, and naturally there are machine learning problems that consider categorical variables. For example, we may wish to model sex (male/female). Since we assume that our data is numerical, we encode sex as the numbers -1 and $+1$ for male and female, respectively. However, it is worth keeping in mind when modeling that sex is a categorical variable, and the actual difference in value between the two numbers should not have any meaning in the model. This distinction between continuous and categorical variables gives rise to different machine learning approaches.

Part II is about Machine Learning

four pillars of machine learning

The second part of the book introduces *four pillars of machine learning* as listed in Table 1.1. The rows in the table distinguish between problems where the variable of interest is continuous or categorical. We illustrate how the mathematical concepts introduced in the first part of the book can be used to design machine learning algorithms. In Chapter 8, we restate the three components of machine learning (data, models and parameter estimation) in a mathematical fashion. In addition, we provide some guidelines for building experimental setups that guard against overly op-

	Supervised	Unsupervised
Continuous latent variables	Regression (Chapter 9)	Dimensionality reduction (Chapter 10)
Categorical latent variables	Classification (Chapter 12)	Density estimation (Chapter 11)

Table 1.1 The four pillars of machine learning

timistic evaluations of machine learning systems. Recall that the goal is to build a predictor that performs well on future data.

The terms "supervised" and "unsupervised" (the columns in Table 1.1) learning refer to the question of whether or not we provide the learning algorithm with labels during training. An example use case of *supervised learning* is when we build a classifier to decide whether a tissue biopsy is cancerous. For training, we provide the machine learning algorithm with a set of images and a corresponding set of annotations by pathologists. This expert annotation is called a *label* in machine learning, and for many supervised learning tasks it is obtained at great cost or effort. After the classifier is trained, we show it an image from a new biopsy and hope that it can accurately predict whether the tissue is cancerous. An example use case of unsupervised learning (using the same cancer biopsy problem) is if we want to visualize the properties of the tissue around which we have found cancerous cells. We could choose two particular features of these images and plot them in a scatter plot. Alternatively we could use all the features and find a two dimensional representation that approximates all the features, and plot this instead. Since this type of machine learning task does not provide a label during training, it is called *unsupervised learning*. The second part of the book provides a brief overview of two fundamental supervised (*regression* and *classification*) and unsupervised (*dimensionality reduction* and *density estimation*) machine learning problems.

supervised learning

label

unsupervised learning

regression

classification

dimensionality reduction

density estimation

Of course there are more than two ways to read this book. Most readers learn using a combination of top-down and bottom-up approaches, sometimes building up basic mathematical skills before attempting more complex concepts, but also choosing topics based on applications of machine learning. Chapters in Part I mostly build upon the previous ones, but the reader is encouraged to skip to a chapter that covers a particular gap the reader's knowledge and work backwards if necessary. Chapters in Part II are loosely coupled and are intended to be read in any order. There are many pointers forward and backward between the two parts of the book to assist the reader in finding their way.

1.3 Exercises and Feedback

We provide some exercises in Part I, which can be done mostly by pen and paper. For Part II we provide programming tutorials (jupyter notebooks) to explore some properties of the machine learning algorithms we discuss in this book.

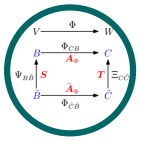
We appreciate that Cambridge University Press strongly supports our aim to democratize education and learning by making this book freely available for download at

https://mml-book.com

where you can also find the tutorials, errata and additional materials. You can also report mistakes and provide feedback using the URL above.

2

Linear Algebra



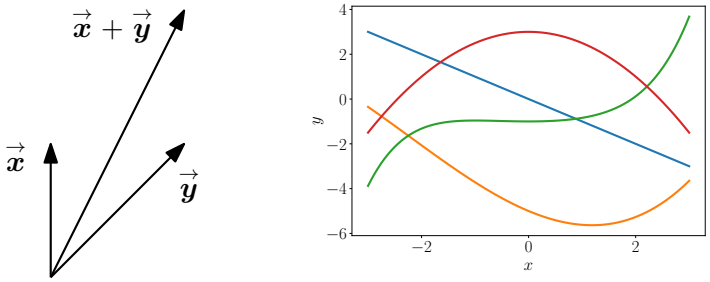
When formalizing intuitive concepts, a common approach is to construct a set of objects (symbols) and a set of rules to manipulate these objects. This is known as an *algebra*.

algebra

Linear algebra is the study of vectors and certain rules to manipulate vectors. The vectors many of us know from school are called "geometric vectors", which are usually denoted by having a small arrow above the letter, e.g., $\vec{x}$ and $\vec{y}$. In this book, we discuss more general concepts of vectors and use a bold letter to represent them, e.g., $\boldsymbol{x}$ and $\boldsymbol{y}$.

In general, vectors are special objects that can be added together and multiplied by scalars to produce another object of the same kind. Any object that satisfies these two properties can be considered a vector. Here are some examples of such vector objects:

1. Geometric vectors. This example of a vector may be familiar from school. Geometric vectors are directed segments, which can be drawn, see Figure 2.1(a). Two geometric vectors $\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}$ can be added, such that $\vec{\boldsymbol{x}} + \vec{\boldsymbol{y}} = \vec{\boldsymbol{z}}$ is another geometric vector. Furthermore, multiplication by a scalar $\lambda \vec{\boldsymbol{x}}, \lambda \in \mathbb{R}$ is also a geometric vector. In fact, it is the original vector scaled by λ. Therefore, geometric vectors are instances of the vector concepts introduced above.
2. Polynomials are also vectors, see Figure 2.1(b): Two polynomials can be added together, which results in another polynomial; and they can be multiplied by a scalar $\lambda \in \mathbb{R}$, and the result is a polynomial as well. Therefore, polynomials are (rather unusual) instances of vectors.



(a) Geometric vectors.

(b) Polynomials.

Figure 2.1 Different types of vectors. Vectors can be surprising objects, including (a) geometric vectors and (b) polynomials.

Note that polynomials are very different from geometric vectors. While geometric vectors are concrete "drawings", polynomials are abstract concepts. However, they are both vectors in the sense described above.

3. Audio signals are vectors. Audio signals are represented as a series of numbers. We can add audio signals together, and their sum is a new audio signal. If we scale an audio signal, we also obtain an audio signal. Therefore, audio signals are a type of vector, too.
4. Elements of $\mathbb{R}^n$ are vectors. In other words, we can consider each element of $\mathbb{R}^n$ (the tuple of n real numbers) to be a vector. $\mathbb{R}^n$ is more abstract than polynomials, and it is the concept we focus on in this book. For example,

$$\boldsymbol{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in \mathbb{R}^3 \tag{2.1}$$

is an example of a triplet of numbers. Adding two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ component-wise results in another vector: $\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{c} \in \mathbb{R}^n$. Moreover, multiplying $\boldsymbol{a} \in \mathbb{R}^n$ by $\lambda \in \mathbb{R}$ results in a scaled vector $\lambda\boldsymbol{a} \in \mathbb{R}^n$.

Linear algebra focuses on the similarities between these vector concepts. We can add them together and multiply them by scalars. We will largely focus on vectors in $\mathbb{R}^n$ since most algorithms in linear algebra are formulated in $\mathbb{R}^n$. Recall that in machine learning, we often consider data to be represented as vectors in $\mathbb{R}^n$. In this book, we will focus on finite-dimensional vector spaces, in which case there is a 1:1 correspondence between any kind of (finite-dimensional) vector and $\mathbb{R}^n$. By studying $\mathbb{R}^n$, we implicitly study all other vectors such as geometric vectors and polynomials. Although $\mathbb{R}^n$ is rather abstract, it is most useful.

One major idea in mathematics is the idea of "closure". This is the question: What is the set of all things that can result from my proposed operations? In the case of vectors: What is the set of vectors that can result by starting with a small set of vectors, and adding them to each other and scaling them? This results in a vector space (Section 2.4). The concept of a vector space and its properties underlie much of machine learning.

matrix

A closely related concept is a *matrix*, which can be thought of as a collection of vectors. As can be expected, when talking about properties of a collection of vectors, we can use matrices as a representation. The concepts introduced in this chapter are shown in Figure 2.2

Pavel Grinfeld's series on linear algebra: `http://tinyurl.com/nahclwm`

This chapter is largely based on the lecture notes and books by Drumm and Weil (2001); Strang (2003); Hogben (2013); Liesen and Mehrmann (2015) as well as Pavel Grinfeld's Linear Algebra series. Another excellent source is Gilbert Strang's Linear Algebra course at MIT.

Gilbert Strang's course on linear algebra: `http://tinyurl.com/29p5q8j`

Linear algebra plays an important role in machine learning and general mathematics. In Chapter 5, we will discuss vector calculus, where a principled knowledge of matrix operations is essential. In Chapter 10,

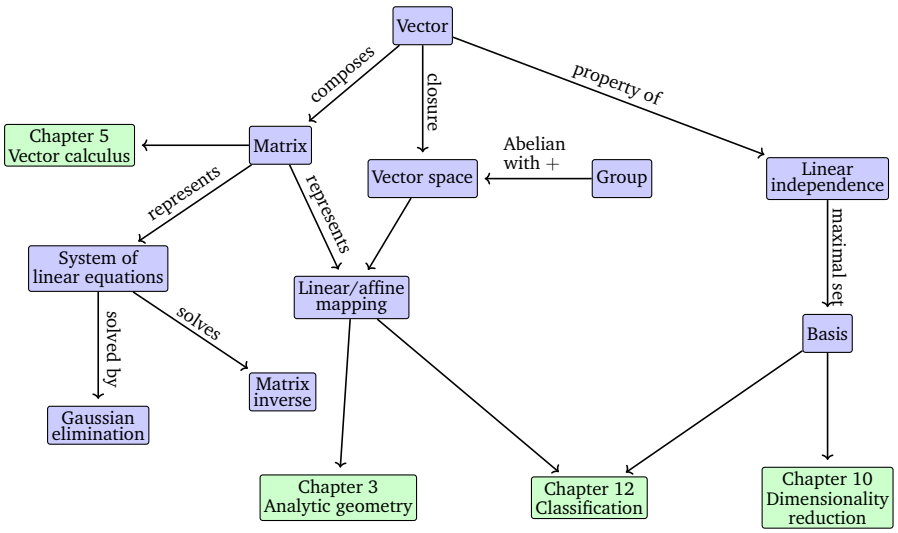


Figure 2.2 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

we will use projections (to be introduced in Section 3.7) for dimensionality reduction with Principal Component Analysis (PCA). In Chapter 9, we will discuss linear regression where linear algebra plays a central role for solving least-squares problems.

2.1 Systems of Linear Equations

Systems of linear equations play a central part of linear algebra. Many problems can be formulated as systems of linear equations, and linear algebra gives us the tools for solving them.

Example 2.1

A company produces products $N_1, \dots, N_n$ for which resources $R_1, \dots, R_m$ are required. To produce a unit of product N_j, a_{ij} units of resource R_i are needed, where $i = 1, \dots, m$ and $j = 1, \dots, n$.

The objective is to find an optimal production plan, i.e., a plan of how many units x_j of product N_j should be produced if a total of b_i units of resource R_i are available and (ideally) no resources are left over.

If we produce $x_1, \dots, x_n$ units of the corresponding products, we need a total of

$$a_{i1}x_1 + \cdots + a_{in}x_n \tag{2.2}$$

many units of resource R_i. The optimal production plan $(x_1, \dots, x_n) \in \mathbb{R}^n$, therefore, has to satisfy the following system of equations:

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n &= b_m \end{aligned}, \tag{2.3}$$

where $a_{ij} \in \mathbb{R}$ and $b_i \in \mathbb{R}$.

system of linear equations
unknowns
solution

Equation (2.3) is the general form of a *system of linear equations*, and $x_1, \ldots, x_n$ are the *unknowns* of this system of linear equations. Every n-tuple $(x_1, \ldots, x_n) \in \mathbb{R}^n$ that satisfies (2.3) is a *solution* of the linear equation system.

Example 2.2
The system of linear equations

$$\begin{array}{rcrcrcll} x_1 & + & x_2 & + & x_3 & = & 3 & (1) \\ x_1 & - & x_2 & + & 2x_3 & = & 2 & (2) \\ 2x_1 & & & + & 3x_3 & = & 1 & (3) \end{array} \tag{2.4}$$

has *no solution:* Adding the first two equations yields $2x_1 + 3x_3 = 5$, which contradicts the third equation (3).

Let us have a look at the system of linear equations

$$\begin{array}{rcrcrcll} x_1 & + & x_2 & + & x_3 & = & 3 & (1) \\ x_1 & - & x_2 & + & 2x_3 & = & 2 & (2) \\ & & x_2 & + & x_3 & = & 2 & (3) \end{array} \quad . \tag{2.5}$$

From the first and third equation it follows that $x_1 = 1$. From (1)+(2) we get $2 + 3x_3 = 5$, i.e., $x_3 = 1$. From (3), we then get that $x_2 = 1$. Therefore, $(1, 1, 1)$ is the only possible and *unique solution* (verify that $(1, 1, 1)$ is a solution by plugging in).

As a third example, we consider

$$\begin{array}{rcrcrcll} x_1 & + & x_2 & + & x_3 & = & 3 & (1) \\ x_1 & - & x_2 & + & 2x_3 & = & 2 & (2) \\ 2x_1 & & & + & 3x_3 & = & 5 & (3) \end{array} \quad . \tag{2.6}$$

Since (1)+(2)=(3), we can omit the third equation (redundancy). From (1) and (2), we get $2x_1 = 5 - 3x_3$ and $2x_2 = 1 + x_3$. We define $x_3 = a \in \mathbb{R}$ as a free variable, such that any triplet

$$\left(\frac{5}{2} - \frac{3}{2}a, \frac{1}{2} + \frac{1}{2}a, a\right), \quad a \in \mathbb{R} \tag{2.7}$$

is a solution to the system of linear equations, i.e., we obtain a solution set that contains *infinitely many* solutions.

In general, for a real-valued system of linear equations we obtain either no, exactly one or infinitely many solutions.

Remark (Geometric Interpretation of Systems of Linear Equations). In a system of linear equations with two variables x_1, x_2, each linear equation determines a line on the x_1x_2-plane. Since a solution to a system of lin-

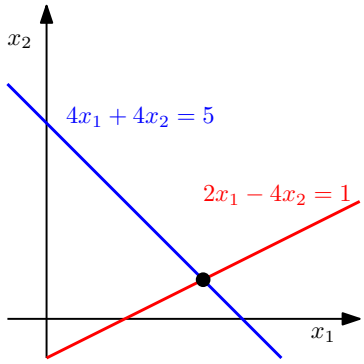


Figure 2.3 The solution space of a system of two linear equations with two variables can be geometrically interpreted as the intersection of two lines. Every linear equation represents a line.

ear equations must satisfy all equations simultaneously, the solution set is the intersection of these line. This intersection can be a line (if the linear equations describe the same line), a point, or empty (when the lines are parallel). An illustration is given in Figure 2.3. Similarly, for three variables, each linear equation determines a plane in three-dimensional space. When we intersect these planes, i.e., satisfy all linear equations at the same time, we can end up with solution set that is a plane, a line, a point or empty (when the planes are parallel). ♢

For a systematic approach to solving systems of linear equations, we will introduce a useful compact notation. We will write the system from (2.3) in the following form:

$$x_1 \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \tag{2.8}$$

$$\iff \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}. \tag{2.9}$$

In the following, we will have a close look at these *matrices* and define computation rules.

2.2 Matrices

Matrices play a central role in linear algebra. They can be used to compactly represent systems of linear equations, but they also represent linear functions (linear mappings) as we will see later in Section 2.7. Before we discuss some of these interesting topics, let us first define what a matrix is and what kind of operations we can do with matrices.

Definition 2.1 (Matrix). With $m, n \in \mathbb{N}$ a real-valued (m, n) *matrix* $\boldsymbol{A}$ is an $m \cdot n$-tuple of elements a_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, which is ordered matrix

according to a rectangular scheme consisting of m rows and n columns:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad a_{ij} \in \mathbb{R}\,. \tag{2.10}$$

We sometimes write $\boldsymbol{A} = ((a_{ij}))$ to indicate that the matrix $\boldsymbol{A}$ is a two-dimensional array consisting of elements a_{ij}. $(1, n)$-matrices are called *rows*, $(m, 1)$-matrices are called *columns*. These special matrices are also called *row/column vectors*.

rows
columns
row/column vectors

$\mathbb{R}^{m\times n}$ is the set of all real-valued (m, n)-matrices. $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ can be equivalently represented as $\boldsymbol{a} \in \mathbb{R}^{mn}$ by stacking all n columns of the matrix into a long vector.

2.2.1 Matrix Addition and Multiplication

The sum of two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$ is defined as the element-wise sum, i.e.,

$$\boldsymbol{A} + \boldsymbol{B} := \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n}\,. \tag{2.11}$$

Note the size of the matrices.

```
C = np.einsum('il, lj', A, B)
```

For matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{B} \in \mathbb{R}^{n\times k}$ the elements c_{ij} of the product $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B} \in \mathbb{R}^{m\times k}$ are defined as

$$c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, k. \tag{2.12}$$

There are n columns in $\boldsymbol{A}$ and n rows in $\boldsymbol{B}$, such that we can compute $a_{il}b_{lj}$ for $l = 1, \ldots, n$.

This means, to compute element c_{ij} we multiply the elements of the ith row of $\boldsymbol{A}$ with the jth column of $\boldsymbol{B}$ and sum them up. Later in Section 3.2, we will call this the *dot product* of the corresponding row and column.

Remark. Matrices can only be multiplied if their "neighboring" dimensions match. For instance, an $n \times k$-matrix $\boldsymbol{A}$ can be multiplied with a $k \times m$-matrix $\boldsymbol{B}$, but only from the left side:

$$\underbrace{\boldsymbol{A}}_{n\times k}\underbrace{\boldsymbol{B}}_{k\times m} = \underbrace{\boldsymbol{C}}_{n\times m} \tag{2.13}$$

The product $\boldsymbol{B}\boldsymbol{A}$ is not defined if $m \neq n$ since the neighboring dimensions do not match. ♢

Remark. Matrix multiplication is *not* defined as an element-wise operation on matrix elements, i.e., $c_{ij} \neq a_{ij}b_{ij}$ (even if the size of $\boldsymbol{A}, \boldsymbol{B}$ was chosen appropriately). This kind of element-wise multiplication often appears in programming languages when we multiply (multi-dimensional) arrays with each other. ♢

Example 2.3

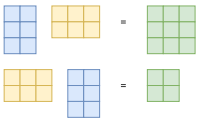
For $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}$, $\boldsymbol{B} = \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 2}$, we obtain

$$\boldsymbol{AB} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 2 & 5 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{2.14}$$

$$\boldsymbol{BA} = \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 4 & 2 \\ -2 & 0 & 2 \\ 3 & 2 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{2.15}$$

From this example, we can already see that matrix multiplication is not commutative, i.e., $\boldsymbol{AB} \neq \boldsymbol{BA}$, see also Figure 2.4 for an illustration.

Figure 2.4 Even if both matrix multiplications $\boldsymbol{AB}$ and $\boldsymbol{BA}$ are defined, the dimensions of the results can be different.



Definition 2.2 (Identity Matrix)**.** In $\mathbb{R}^{n\times n}$, we define the *identity matrix* as

$$\boldsymbol{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{2.16}$$

identity matrix

as the $n \times n$-matrix containing 1 on the diagonal and 0 everywhere else. With this, $\boldsymbol{A} \cdot \boldsymbol{I}_n = \boldsymbol{A} = \boldsymbol{I}_n \cdot \boldsymbol{A}$ for all $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.

Now that we have defined matrix multiplication, matrix addition and the identity matrix, let us have a look at some properties of matrices, where we will omit the "·" for matrix multiplication:

- Associativity:

$$\forall \boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{B} \in \mathbb{R}^{n\times p}, \boldsymbol{C} \in \mathbb{R}^{p\times q} : (\boldsymbol{AB})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{BC}) \tag{2.17}$$

- Distributivity:

$$\forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}, \boldsymbol{C}, \boldsymbol{D} \in \mathbb{R}^{n\times p} : (\boldsymbol{A} + \boldsymbol{B})\boldsymbol{C} = \boldsymbol{AC} + \boldsymbol{BC} \tag{2.18a}$$

$$\boldsymbol{A}(\boldsymbol{C} + \boldsymbol{D}) = \boldsymbol{AC} + \boldsymbol{AD} \tag{2.18b}$$

- Neutral element:

$$\forall \boldsymbol{A} \in \mathbb{R}^{m\times n} : \boldsymbol{I}_m \boldsymbol{A} = \boldsymbol{A}\boldsymbol{I}_n = \boldsymbol{A} \tag{2.19}$$

Note that $\boldsymbol{I}_m \neq \boldsymbol{I}_n$ for $m \neq n$.

2.2.2 Inverse and Transpose

A square matrix possesses the same number of columns and rows.

Definition 2.3 (Inverse)**.** For a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{AB} = \boldsymbol{I}_n = \boldsymbol{BA}$ the matrix $\boldsymbol{B}$ is called *inverse* and denoted by $\boldsymbol{A}^{-1}$.

inverse

Unfortunately, not every matrix $\boldsymbol{A}$ possesses an inverse $\boldsymbol{A}^{-1}$. If this inverse does exist, $\boldsymbol{A}$ is called *regular/invertible/non-singular*, otherwise *singular/non-invertible*.

regular
invertible
non-singular
singular
non-invertible

Remark (Existence of the Inverse of a 2×2-Matrix). Consider a matrix

$$\boldsymbol{A} := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in \mathbb{R}^{2\times 2}\,. \tag{2.20}$$

If we multiply $\boldsymbol{A}$ with

$$\boldsymbol{B} := \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \tag{2.21}$$

we obtain

$$\boldsymbol{AB} = \begin{bmatrix} a_{11}a_{22} - a_{12}a_{21} & 0 \\ 0 & a_{11}a_{22} - a_{12}a_{21} \end{bmatrix} = (a_{11}a_{22} - a_{12}a_{21})\boldsymbol{I} \tag{2.22}$$

so that

$$\boldsymbol{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \tag{2.23}$$

if and only if $a_{11}a_{22} - a_{12}a_{21} \neq 0$. In Section 4.1, we will see that $a_{11}a_{22} - a_{12}a_{21}$ is the determinant of a 2×2-matrix. Furthermore, we can generally use the determinant to check whether a matrix is invertible. ♢

Example 2.4 (Inverse Matrix)
The matrices

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ 4 & 4 & 5 \\ 6 & 7 & 7 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} -7 & -7 & 6 \\ 2 & 1 & -1 \\ 4 & 5 & -4 \end{bmatrix} \tag{2.24}$$

are inverse to each other since $\boldsymbol{AB} = \boldsymbol{I} = \boldsymbol{BA}$.

Definition 2.4 (Transpose)**.** For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ the matrix $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ with $b_{ij} = a_{ji}$ is called the *transpose* of $\boldsymbol{A}$. We write $\boldsymbol{B} = \boldsymbol{A}^\top$.

transpose

The main diagonal (sometimes called "principal diagonal", "primary diagonal", "leading diagonal", or "major diagonal") of a matrix $\boldsymbol{A}$ is the collection of entries A_{ij} where $i = j$.

For a square matrix $\boldsymbol{A}^\top$ is the matrix we obtain when we "mirror" $\boldsymbol{A}$ on its main diagonal. In general, $\boldsymbol{A}^\top$ can be obtained by writing the columns of $\boldsymbol{A}$ as the rows of $\boldsymbol{A}^\top$.

Some important properties of inverses and transposes are:

$$\boldsymbol{AA}^{-1} = \boldsymbol{I} = \boldsymbol{A}^{-1}\boldsymbol{A} \tag{2.25}$$

$$(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \tag{2.26}$$

$$(\boldsymbol{A}+\boldsymbol{B})^{-1} \neq \boldsymbol{A}^{-1}+\boldsymbol{B}^{-1} \tag{2.27}$$

$$(\boldsymbol{A}^\top)^\top = \boldsymbol{A} \tag{2.28}$$

$$(\boldsymbol{A}+\boldsymbol{B})^\top = \boldsymbol{A}^\top + \boldsymbol{B}^\top \tag{2.29}$$

$$(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top \tag{2.30}$$

Moreover, if $\boldsymbol{A}$ is invertible then so is $\boldsymbol{A}^\top$ and $(\boldsymbol{A}^{-1})^\top = (\boldsymbol{A}^\top)^{-1} =: \boldsymbol{A}^{-\top}$

In the scalar case $\frac{1}{2+4} = \frac{1}{6} \neq \frac{1}{2} + \frac{1}{4}$.

A matrix $\boldsymbol{A}$ is *symmetric* if $\boldsymbol{A} = \boldsymbol{A}^\top$. Note that this can only hold for (n, n)-matrices, which we also call *square matrices* because they possess the same number of rows and columns.

symmetric

square matrices

Remark (Sum and Product of Symmetric Matrices). The sum of symmetric matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ is always symmetric. However, although their product is always defined, it is generally not symmetric:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}. \tag{2.31}$$

♢

2.2.3 Multiplication by a Scalar

Let us have a brief look at what happens to matrices when they are multiplied by a scalar $\lambda \in \mathbb{R}$. Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\lambda \in \mathbb{R}$. Then $\lambda\boldsymbol{A} = \boldsymbol{K}$, $K_{ij} = \lambda\, a_{ij}$. Practically, λ scales each element of $\boldsymbol{A}$. For $\lambda, \psi \in \mathbb{R}$ it holds:

- Distributivity:
  $(\lambda+\psi)\boldsymbol{C} = \lambda\boldsymbol{C} + \psi\boldsymbol{C}, \quad \boldsymbol{C} \in \mathbb{R}^{m\times n}$
  $\lambda(\boldsymbol{B}+\boldsymbol{C}) = \lambda\boldsymbol{B} + \lambda\boldsymbol{C}, \quad \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{m\times n}$
- Associativity:
  $(\lambda\psi)\boldsymbol{C} = \lambda(\psi\boldsymbol{C}), \quad \boldsymbol{C} \in \mathbb{R}^{m\times n}$
  $\lambda(\boldsymbol{B}\boldsymbol{C}) = (\lambda\boldsymbol{B})\boldsymbol{C} = \boldsymbol{B}(\lambda\boldsymbol{C}) = (\boldsymbol{B}\boldsymbol{C})\lambda, \quad \boldsymbol{B} \in \mathbb{R}^{m\times n}, \boldsymbol{C} \in \mathbb{R}^{n\times k}$.
  Note that this allows us to move scalar values around.
- $(\lambda\boldsymbol{C})^\top = \boldsymbol{C}^\top\lambda^\top = \boldsymbol{C}^\top\lambda = \lambda\boldsymbol{C}^\top$ since $\lambda = \lambda^\top$ for all $\lambda \in \mathbb{R}$.

Example 2.5 (Distributivity)
If we define

$$\boldsymbol{C} := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{2.32}$$

then for any $\lambda, \psi \in \mathbb{R}$ we obtain

$$(\lambda+\psi)\boldsymbol{C} = \begin{bmatrix} (\lambda+\psi)1 & (\lambda+\psi)2 \\ (\lambda+\psi)3 & (\lambda+\psi)4 \end{bmatrix} = \begin{bmatrix} \lambda+\psi & 2\lambda+2\psi \\ 3\lambda+3\psi & 4\lambda+4\psi \end{bmatrix} \tag{2.33a}$$

$$= \begin{bmatrix} \lambda & 2\lambda \\ 3\lambda & 4\lambda \end{bmatrix} + \begin{bmatrix} \psi & 2\psi \\ 3\psi & 4\psi \end{bmatrix} = \lambda\boldsymbol{C} + \psi\boldsymbol{C} \tag{2.33b}$$

2.2.4 Compact Representations of Systems of Linear Equations

If we consider the system of linear equations

$$\begin{aligned} 2x_1 + 3x_2 + 5x_3 &= 1 \\ 4x_1 - 2x_2 - 7x_3 &= 8 \\ 9x_1 + 5x_2 - 3x_3 &= 2 \end{aligned} \tag{2.34}$$

and use the rules for matrix multiplication, we can write this equation system in a more compact form as

$$\begin{bmatrix} 2 & 3 & 5 \\ 4 & -2 & -7 \\ 9 & 5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \\ 2 \end{bmatrix}. \tag{2.35}$$

Note that x_1 scales the first column, x_2 the second one, and x_3 the third one.

Generally, system of linear equations can be compactly represented in their matrix form as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, see (2.3), and the product $\boldsymbol{A}\boldsymbol{x}$ is a (linear) combination of the columns of $\boldsymbol{A}$. We will discuss linear combinations in more detail in Section 2.5.

2.3 Solving Systems of Linear Equations

In (2.3), we introduced the general form of an equation system, i.e.,

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n &= b_m \,, \end{aligned} \tag{2.36}$$

where $a_{ij} \in \mathbb{R}$ and $b_i \in \mathbb{R}$ are known constants and x_j are unknowns, $i = 1, \ldots, m$, $j = 1, \ldots, n$. Thus far, we saw that matrices can be used as a compact way of formulating systems of linear equations so that we can write $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, see (2.9). Moreover, we defined basic matrix operations, such as addition and multiplication of matrices. In the following, we will focus on solving systems of linear equations.

2.3.1 Particular and General Solution

Before discussing how to solve systems of linear equations systematically, let us have a look at an example. Consider the system of equations

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 42 \\ 8 \end{bmatrix}. \tag{2.37}$$

This system of equations is in a particularly easy form, where the first two columns consist of a 1 and a 0. Remember that we want to find scalars $x_1, \ldots, x_4$, such that $\sum_{i=1}^{4} x_i \boldsymbol{c}_i = \boldsymbol{b}$, where we define $\boldsymbol{c}_i$ to be the ith column of the matrix and $\boldsymbol{b}$ the right-hand-side of (2.37). A solution to the problem in (2.37) can be found immediately by taking 42 times the first column and 8 times the second column so that

$$\boldsymbol{b} = \begin{bmatrix} 42 \\ 8 \end{bmatrix} = 42 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 8 \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.38}$$

Therefore, a solution vector is $[42, 8, 0, 0]^\top$. This solution is called a *particular solution* or *special solution*. However, this is not the only solution of this system of linear equations. To capture all the other solutions, we need to be creative of generating $\mathbf{0}$ in a non-trivial way using the columns of the matrix: Adding $\mathbf{0}$ to our special solution does not change the special solution. To do so, we express the third column using the first two columns (which are of this very simple form)

particular solution

special solution

$$\begin{bmatrix} 8 \\ 2 \end{bmatrix} = 8 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.39}$$

so that $\mathbf{0} = 8\boldsymbol{c}_1 + 2\boldsymbol{c}_2 - 1\boldsymbol{c}_3 + 0\boldsymbol{c}_4$ and $(x_1, x_2, x_3, x_4) = (8, 2, -1, 0)$. In fact, any scaling of this solution by $\lambda_1 \in \mathbb{R}$ produces the $\mathbf{0}$ vector, i.e.,

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \left(\lambda_1 \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix} \right) = \lambda_1 (8\boldsymbol{c}_1 + 2\boldsymbol{c}_2 - \boldsymbol{c}_3) = \mathbf{0}\,. \tag{2.40}$$

Following the same line of reasoning, we express the fourth column of the matrix in (2.37) using the first two columns and generate another set of non-trivial versions of $\mathbf{0}$ as

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \left(\lambda_2 \begin{bmatrix} -4 \\ 12 \\ 0 \\ -1 \end{bmatrix} \right) = \lambda_2 (-4\boldsymbol{c}_1 + 12\boldsymbol{c}_2 - \boldsymbol{c}_4) = \mathbf{0} \tag{2.41}$$

for any $\lambda_2 \in \mathbb{R}$. Putting everything together, we obtain all solutions of the equation system in (2.37), which is called the *general solution*, as the set

general solution

$$\left\{ \boldsymbol{x} \in \mathbb{R}^4 : \boldsymbol{x} = \begin{bmatrix} 42 \\ 8 \\ 0 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} -4 \\ 12 \\ 0 \\ -1 \end{bmatrix}, \lambda_1, \lambda_2 \in \mathbb{R} \right\}. \tag{2.42}$$

Remark. The general approach we followed consisted of the following three steps:

1. Find a particular solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$
2. Find all solutions to $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$

3. Combine the solutions from 1. and 2. to the general solution.

Neither the general nor the particular solution is unique. ♢

The system of linear equations in the example above was easy to solve because the matrix in (2.37) has this particularly convenient form, which allowed us to find the particular and the general solution by inspection. However, general equation systems are not of this simple form. Fortunately, there exists a constructive algorithmic way of transforming any system of linear equations into this particularly simple form: Gaussian elimination. Key to Gaussian elimination are elementary transformations of systems of linear equations, which transform the equation system into a simple form. Then, we can apply the three steps to the simple form that we just discussed in the context of the example in (2.37), see the remark above.

2.3.2 Elementary Transformations

elementary transformations

Key to solving a system of linear equations are *elementary transformations* that keep the solution set the same, but that transform the equation system into a simpler form:

- Exchange of two equations (or: rows in the matrix representing the equation system)
- Multiplication of an equation (row) with a constant $\lambda \in \mathbb{R}\backslash\{0\}$
- Addition of two equations (rows)

Example 2.6

For $a \in \mathbb{R}$, we seek all solutions of the following system of equations:

$$\begin{array}{rcrcrcrcrcr} -2x_1 & + & 4x_2 & - & 2x_3 & - & x_4 & + & 4x_5 & = & -3 \\ 4x_1 & - & 8x_2 & + & 3x_3 & - & 3x_4 & + & x_5 & = & 2 \\ x_1 & - & 2x_2 & + & x_3 & - & x_4 & + & x_5 & = & 0 \\ x_1 & - & 2x_2 & & & - & 3x_4 & + & 4x_5 & = & a \end{array} \quad . \tag{2.43}$$

We start by converting this system of equations into the compact matrix notation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We no longer mention the variables $\boldsymbol{x}$ explicitly and build the *augmented matrix*

augmented matrix

$$\left[\begin{array}{rrrrr|r} -2 & 4 & -2 & -1 & 4 & -3 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ 1 & -2 & 1 & -1 & 1 & 0 \\ 1 & -2 & 0 & -3 & 4 & a \end{array}\right] \begin{array}{l} \\ \\ \text{Swap with } R_3 \\ \\ \text{Swap with } R_1 \end{array}$$

where we used the vertical line to separate the left-hand-side from the right-hand-side in (2.43). We use $\rightsquigarrow$ to indicate a transformation of the left-hand-side into the right-hand-side using elementary transformations.

Swapping rows 1 and 3 leads to

$$\left[\begin{array}{rrrrr|r} 1 & -2 & 1 & -1 & 1 & 0 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ -2 & 4 & -2 & -1 & 4 & -3 \\ 1 & -2 & 0 & -3 & 4 & a \end{array}\right] \begin{array}{l} \\ -4R_1 \\ +2R_1 \\ -R_1 \end{array}$$

When we now apply the indicated transformations (e.g., subtract Row 1 four times from Row 2), we obtain

The augmented matrix $[\boldsymbol{A}\,|\,\boldsymbol{b}]$ compactly represents the system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

$$\left[\begin{array}{rrrrr|r} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & -1 & -2 & 3 & a \end{array}\right] -R_2 - R_3$$

$$\rightsquigarrow \left[\begin{array}{rrrrr|r} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & 0 & 0 & 0 & a+1 \end{array}\right] \begin{array}{l} \\ \cdot(-1) \\ \cdot(-\frac{1}{3}) \\ \\ \end{array}$$

$$\rightsquigarrow \left[\begin{array}{rrrrr|r} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 3 & -2 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & a+1 \end{array}\right]$$

This (augmented) matrix is in a convenient form, the *row-echelon form (REF)*. Reverting this compact notation back into the explicit notation with the variables we seek, we obtain

row-echelon form (REF)

$$\begin{array}{rcrcrcrcr} x_1 & - & 2x_2 & + & x_3 & - & x_4 & + & x_5 & = & 0 \\ & & & & x_3 & - & x_4 & + & 3x_5 & = & -2 \\ & & & & & & x_4 & - & 2x_5 & = & 1 \\ & & & & & & & & 0 & = & a+1 \end{array} \quad . \tag{2.44}$$

Only for $a = -1$ this system can be solved. A *particular solution* is

particular solution

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix} . \tag{2.45}$$

The *general solution*, which captures the set of all possible solutions, is

general solution

$$\left\{ \boldsymbol{x} \in \mathbb{R}^5 : \boldsymbol{x} = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 2 \\ 0 \\ -1 \\ 2 \\ 1 \end{bmatrix} , \quad \lambda_1, \lambda_2 \in \mathbb{R} \right\} . \tag{2.46}$$

In the following, we will detail a constructive way to obtain a particular and general solution of a system of linear equations.

Remark (Pivots and Staircase Structure). The leading coefficient of a row (first non-zero number from the left) is called the *pivot* and is always strictly to the right of the pivot of the row above it. Therefore, any equation system in row echelon form always has a "staircase" structure. ♢

pivot

row echelon form

Definition 2.5 (Row Echelon Form)**.** A matrix is in *row echelon form* (REF) if

- All rows that contain only zeros are at the bottom of the matrix; correspondingly, all rows that contain at least one non-zero element are on top of rows that contain only zeros.
- Looking at non-zero rows only, the first non-zero number from the left (also called the *pivot* or the *leading coefficient*) is always strictly to the right of the pivot of the row above it.

pivot
leading coefficient
In other books, it is sometimes required that the pivot is 1.

Remark (Basic and Free Variables). The variables corresponding to the pivots in the row-echelon form are called *basic variables*, the other variables are *free variables*. For example, in (2.44), x_1, x_3, x_4 are basic variables, whereas x_2, x_5 are free variables. ♢

basic variables
free variables

Remark (Obtaining a Particular Solution). The row echelon form makes our lives easier when we need to determine a particular solution. To do this, we express the right-hand side of the equation system using the pivot columns, such that $\boldsymbol{b} = \sum_{i=1}^{P} \lambda_i \boldsymbol{p}_i$, where $\boldsymbol{p}_i,\ i = 1, \ldots, P$, are the pivot columns. The λ_i are determined easiest if we start with the most-right pivot column and work our way to the left.

In the above example, we would try to find $\lambda_1, \lambda_2, \lambda_3$ such that

$$\lambda_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \lambda_3 \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \\ 1 \\ 0 \end{bmatrix} . \tag{2.47}$$

From here, we find relatively directly that $\lambda_3 = 1, \lambda_2 = -1, \lambda_1 = 2$. When we put everything together, we must not forget the non-pivot columns for which we set the coefficients implicitly to 0. Therefore, we get the particular solution $\boldsymbol{x} = [2, 0, -1, 1, 0]^\top$. ♢

reduced row echelon form

Remark (Reduced Row Echelon Form). An equation system is in *reduced row echelon form* (also: *row-reduced echelon form* or *row canonical form*) if

- It is in row echelon form.
- Every pivot is 1.
- The pivot is the only non-zero entry in its column.

♢

The reduced row echelon form will play an important role later in Section 2.3.3 because it allows us to determine the general solution of a system of linear equations in a straightforward way.

Gaussian elimination

Remark (Gaussian Elimination). *Gaussian elimination* is an algorithm that performs elementary transformations to bring a system of linear equations into reduced row echelon form. ♢

Example 2.7 (Reduced Row Echelon Form)
Verify that the following matrix is in reduced row echelon form (the pivots are in **bold**):

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & 3 & 0 & 0 & 3 \\ 0 & 0 & \mathbf{1} & 0 & 9 \\ 0 & 0 & 0 & \mathbf{1} & -4 \end{bmatrix} \tag{2.48}$$

The key idea for finding the solutions of $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$ is to look at the *non-pivot columns*, which we will need to express as a (linear) combination of the pivot columns. The reduced row echelon form makes this relatively straightforward, and we express the non-pivot columns in terms of sums and multiples of the pivot columns that are on their left: The second column is 3 times the first column (we can ignore the pivot columns on the right of the second column). Therefore, to obtain $\mathbf{0}$, we need to subtract the second column from three times the first column. Now, we look at the fifth column, which is our second non-pivot column. The fifth column can be expressed as 3 times the first pivot column, 9 times the second pivot column, and -4 times the third pivot column. We need to keep track of the indices of the pivot columns and translate this into 3 times the first column, 0 times the second column (which is a non-pivot column), 9 times the third pivot column (which is our second pivot column), and -4 times the fourth column (which is the third pivot column). Then we need to subtract the fifth column to obtain $\mathbf{0}$. In the end, we are still solving a homogeneous equation system.

To summarize, all solutions of $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}, \boldsymbol{x} \in \mathbb{R}^5$ are given by

$$\left\{ \boldsymbol{x} \in \mathbb{R}^5 : \boldsymbol{x} = \lambda_1 \begin{bmatrix} 3 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ 0 \\ 9 \\ -4 \\ -1 \end{bmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{R} \right\} . \tag{2.49}$$

2.3.3 The Minus-1 Trick

In the following, we introduce a practical trick for reading out the solutions $\boldsymbol{x}$ of a homogeneous system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$, where $\boldsymbol{A} \in \mathbb{R}^{k \times n}, \boldsymbol{x} \in \mathbb{R}^n$.

To start, we assume that $\boldsymbol{A}$ is in reduced row echelon form without any rows that just contain zeros, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} 0 & \cdots & 0 & \mathbf{1} & * & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * \\ \vdots & & \vdots & 0 & 0 & \cdots & 0 & \mathbf{1} & * & \cdots & * & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & 0 & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & 0 & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \mathbf{1} & * & \cdots & * \end{bmatrix}, \tag{2.50}$$

where $*$ can be an arbitrary real number, with the constraints that the first non-zero entry per row must be 1 and all other entries in the corresponding column must be 0. The columns $j_1, \ldots, j_k$ with the pivots (marked in **bold**) are the standard unit vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k \in \mathbb{R}^k$. We extend this matrix to an $n \times n$-matrix $\tilde{\boldsymbol{A}}$ by adding $n - k$ rows of the form

$$\begin{bmatrix} 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \end{bmatrix} \tag{2.51}$$

so that the diagonal of the augmented matrix $\tilde{\boldsymbol{A}}$ contains either 1 or -1. Then, the columns of $\tilde{\boldsymbol{A}}$, which contain the -1 as pivots are solutions of the homogeneous equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. To be more precise, these columns form a basis (Section 2.6.1) of the solution space of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$, which we will later call the *kernel* or *null space* (see Section 2.7.3).

kernel
null space

Example 2.8 (Minus-1 Trick)
Let us revisit the matrix in (2.48), which is already in REF:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 3 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & -4 \end{bmatrix}. \tag{2.52}$$

We now augment this matrix to a 5×5 matrix by adding rows of the form (2.51) at the places where the pivots on the diagonal are missing and obtain

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} 1 & 3 & 0 & 0 & 3 \\ 0 & -\mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 0 & -\mathbf{1} \end{bmatrix} \tag{2.53}$$

From this form, we can immediately read out the solutions of $\boldsymbol{Ax} = \boldsymbol{0}$ by taking the columns of $\tilde{\boldsymbol{A}}$, which contain -1 on the diagonal:

$$\left\{ \boldsymbol{x} \in \mathbb{R}^5 : \boldsymbol{x} = \lambda_1 \begin{bmatrix} 3 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ 0 \\ 9 \\ -4 \\ -1 \end{bmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{R} \right\}, \tag{2.54}$$

which is identical to the solution in (2.49) that we obtained by "insight".

Calculating the Inverse

To compute the inverse $\boldsymbol{A}^{-1}$ of $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, we need to find a matrix $\boldsymbol{X}$ that satisfies $\boldsymbol{AX} = \boldsymbol{I}_n$. Then, $\boldsymbol{X} = \boldsymbol{A}^{-1}$. We can write this down as a set of simultaneous linear equations $\boldsymbol{AX} = \boldsymbol{I}_n$, where we solve for $\boldsymbol{X} = [\boldsymbol{x}_1 | \cdots | \boldsymbol{x}_n]$. We use the augmented matrix notation for a compact representation of this set of systems of linear equations and obtain

$$\left[\boldsymbol{A} | \boldsymbol{I}_n\right] \quad \rightsquigarrow \cdots \rightsquigarrow \quad \left[\boldsymbol{I}_n | \boldsymbol{A}^{-1}\right]. \tag{2.55}$$

This means that if we bring the augmented equation system into reduced row echelon form, we can read out the inverse on the right-hand side of the equation system. Hence, determining the inverse of a matrix is equivalent to solving systems of linear equations.

Example 2.9 (Calculating an Inverse Matrix by Gaussian Elimination)
To determine the inverse of

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.56}$$

we write down the augmented matrix

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

and use Gaussian elimination to bring it into reduced row echelon form

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & -1 & 2 & -2 & 2 \\ 0 & 1 & 0 & 0 & 1 & -1 & 2 & -2 \\ 0 & 0 & 1 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & -1 & 2 \end{array}\right],$$

such that the desired inverse is given as its right-hand side:

$$\boldsymbol{A}^{-1} = \begin{bmatrix} -1 & 2 & -2 & 2 \\ 1 & -1 & 2 & -2 \\ 1 & -1 & 1 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}. \tag{2.57}$$

2.3.4 Algorithms for Solving a System of Linear Equations

In the following, we briefly discuss approaches to solving a system of linear equations of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

In special cases, we may be able to determine the inverse $\boldsymbol{A}^{-1}$, such that the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is given as $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. However, this is only possible if $\boldsymbol{A}$ is a square matrix and invertible, which is often not the case. Otherwise, under mild assumptions (i.e., $\boldsymbol{A}$ needs to have linearly independent columns) we can use the transformation

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \iff \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top\boldsymbol{b} \iff \boldsymbol{x} = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b} \tag{2.58}$$

Moore-Penrose pseudo-inverse

and use the *Moore-Penrose pseudo-inverse* $(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top$ to determine the solution (2.58) that solves $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, which also corresponds to the minimum norm least-squares solution. A disadvantage of this approach is that it requires many computations for the matrix-matrix product and computing the inverse of $\boldsymbol{A}^\top\boldsymbol{A}$. Moreover, for reasons of numerical precision it is generally not recommended to compute the inverse or pseudo-inverse. In the following, we therefore briefly discuss alternative approaches to solving systems of linear equations.

Gaussian elimination plays an important role when computing determinants (Section 4.1), checking whether a set of vectors is linearly independent (Section 2.5), computing the inverse of a matrix (Section 2.2.2), computing the rank of a matrix (Section 2.6.2) and a basis of a vector space (Section 2.6.1). We will discuss all these topics later on. Gaussian elimination is an intuitive and constructive way to solve a system of linear equations with thousands of variables. However, for systems with millions of variables, it is impractical as the required number of arithmetic operations scales cubically in the number of simultaneous equations.

In practice, systems of many linear equations are solved indirectly, by either stationary iterative methods, such as the Richardson method, the Jacobi method, the Gauß-Seidel method, or the successive over-relaxation method, or Krylov subspace methods, such as conjugate gradients, generalized minimal residual, or biconjugate gradients.

Let $\boldsymbol{x}_*$ be a solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. The key idea of these iterative methods

is to set up an iteration of the form

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{A}\boldsymbol{x}^{(k)} \tag{2.59}$$

that reduces the residual error $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}_*\|$ in every iteration and finally converges to $\boldsymbol{x}_*$. We will introduce norms $\|\cdot\|$, which allow us to compute similarities between vectors, in Section 3.1.

2.4 Vector Spaces

Thus far, we have looked at systems of linear equations and how to solve them. We saw that systems of linear equations can be compactly represented using matrix-vector notations. In the following, we will have a closer look at vector spaces, i.e., a structured space in which vectors live.

In the beginning of this chapter, we informally characterized vectors as objects that can be added together and multiplied by a scalar, and they remain objects of the same type (see page 17). Now, we are ready to formalize this, and we will start by introducing the concept of a group, which is a set of elements and an operation defined on these elements that keeps some structure of the set intact.

2.4.1 Groups

Groups play an important role in computer science. Besides providing a fundamental framework for operations on sets, they are heavily used in cryptography, coding theory and graphics.

Definition 2.6 (Group)**.** Consider a set $\mathcal{G}$ and an operation $\otimes : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ defined on $\mathcal{G}$.

Then $G := (\mathcal{G}, \otimes)$ is called a *group* if the following hold: group

1. *Closure* of $\mathcal{G}$ under $\otimes$: $\forall x, y \in \mathcal{G} : x \otimes y \in \mathcal{G}$ Closure
2. *Associativity:* $\forall x, y, z \in \mathcal{G} : (x \otimes y) \otimes z = x \otimes (y \otimes z)$ Associativity:
3. *Neutral element:* $\exists e \in \mathcal{G} \, \forall x \in \mathcal{G} : x \otimes e = x$ and $e \otimes x = x$ Neutral element:
4. *Inverse element:* $\forall x \in \mathcal{G} \, \exists y \in \mathcal{G} : x \otimes y = e$ and $y \otimes x = e$. We often write x^{-1} to denote the inverse element of x. Inverse element:

If additionally $\forall x, y \in \mathcal{G} : x \otimes y = y \otimes x$ then $G = (\mathcal{G}, \otimes)$ is an *Abelian group* (commutative). Abelian group

Example 2.10 (Groups)
Let us have a look at some examples of sets with associated operations and see whether they are groups.

- $(\mathbb{Z}, +)$ is a group.

$\mathbb{N}_0 := \mathbb{N} \cup \{0\}$

- $(\mathbb{N}_0, +)$ is not a group: Although $(\mathbb{N}_0, +)$ possesses a neutral element (0), the inverse elements are missing.
- $(\mathbb{Z}, \cdot)$ is not a group: Although $(\mathbb{Z}, \cdot)$ contains a neutral element (1), the inverse elements for any $z \in \mathbb{Z}, z \neq \pm 1$, are missing.
- $(\mathbb{R}, \cdot)$ is not a group since 0 does not possess an inverse element.
- $(\mathbb{R}\backslash\{0\})$ is Abelian.
- $(\mathbb{R}^n, +), (\mathbb{Z}^n, +), n \in \mathbb{N}$ are Abelian if $+$ is defined componentwise, i.e.,

$$(x_1, \cdots, x_n) + (y_1, \cdots, y_n) = (x_1 + y_1, \cdots, x_n + y_n). \tag{2.60}$$

  Then, $(x_1, \cdots, x_n)^{-1} := (-x_1, \cdots, -x_n)$ is the inverse element and $e = (0, \cdots, 0)$ is the neutral element.
- $(\mathbb{R}^{m \times n}, +)$, the set of $m \times n$-matrices is Abelian (with componentwise addition as defined in (2.60)).
- Let us have a closer look at $(\mathbb{R}^{n \times n}, \cdot)$, i.e., the set of $n \times n$-matrices with matrix multiplication as defined in (2.12).
  - Closure and associativity follow directly from the definition of matrix multiplication.
  - Neutral element: The identity matrix $\boldsymbol{I}_n$ is the neutral element with respect to matrix multiplication "$\cdot$" in $(\mathbb{R}^{n \times n}, \cdot)$.
  - Inverse element: If the inverse exists then $\boldsymbol{A}^{-1}$ is the inverse element of $\boldsymbol{A} \in \mathbb{R}^{n \times n}$.

If $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ then $\boldsymbol{I}_n$ is only a right neutral element, such that $\boldsymbol{A}\boldsymbol{I}_n = \boldsymbol{A}$. The corresponding left-neutral element would be $\boldsymbol{I}_m$ since $\boldsymbol{I}_m\boldsymbol{A} = \boldsymbol{A}$.

Remark. The inverse element is defined with respect to the operation $\otimes$ and does not necessarily mean $\frac{1}{x}$. ♢

Definition 2.7 (General Linear Group)**.** The set of regular (invertible) matrices $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a group with respect to matrix multiplication as defined in (2.12) and is called *general linear group* $GL(n, \mathbb{R})$. However, since matrix multiplication is not commutative, the group is not Abelian.

general linear group

2.4.2 Vector Spaces

When we discussed groups, we looked at sets $\mathcal{G}$ and inner operations on $\mathcal{G}$, i.e., mappings $\mathcal{G} \times \mathcal{G} \to \mathcal{G}$ that only operate on elements in $\mathcal{G}$. In the following, we will consider sets that in addition to an inner operation $+$ also contain an outer operation $\cdot$, the multiplication of a vector $\boldsymbol{x} \in \mathcal{G}$ by a scalar $\lambda \in \mathbb{R}$.

vector space

Definition 2.8 (Vector space)**.** A real-valued *vector space* $V = (\mathcal{V}, +, \cdot)$ is a set $\mathcal{V}$ with two operations

$$+ : \mathcal{V} \times \mathcal{V} \to \mathcal{V} \tag{2.61}$$

$$\cdot : \mathbb{R} \times \mathcal{V} \to \mathcal{V} \tag{2.62}$$

where

1. $(\mathcal{V}, +)$ is an Abelian group
2. Distributivity:
   1. $\forall \lambda \in \mathbb{R}, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{V} : \lambda \cdot (\boldsymbol{x} + \boldsymbol{y}) = \lambda \cdot \boldsymbol{x} + \lambda \cdot \boldsymbol{y}$
   2. $\forall \lambda, \psi \in \mathbb{R}, \boldsymbol{x} \in \mathcal{V} : (\lambda + \psi) \cdot \boldsymbol{x} = \lambda \cdot \boldsymbol{x} + \psi \cdot \boldsymbol{x}$
3. Associativity (outer operation): $\forall \lambda, \psi \in \mathbb{R}, \boldsymbol{x} \in \mathcal{V} : \lambda \cdot (\psi \cdot \boldsymbol{x}) = (\lambda\psi) \cdot \boldsymbol{x}$
4. Neutral element with respect to the outer operation: $\forall \boldsymbol{x} \in \mathcal{V} : 1 \cdot \boldsymbol{x} = \boldsymbol{x}$

The elements $\boldsymbol{x} \in V$ are called *vectors*. The neutral element of $(\mathcal{V}, +)$ is the zero vector $\mathbf{0} = [0, \dots, 0]^\top$, and the inner operation $+$ is called *vector addition*. The elements $\lambda \in \mathbb{R}$ are called *scalars* and the outer operation $\cdot$ is a *multiplication by scalars*. Note that a scalar product is something different, and we will get to this in Section 3.2.

vectors
vector addition
scalars
multiplication by scalars

Remark. A "vector multiplication" $\boldsymbol{ab}$, $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, is not defined. Theoretically, we could define an element-wise multiplication, such that $\boldsymbol{c} = \boldsymbol{ab}$ with $c_j = a_j b_j$. This "array multiplication" is common to many programming languages but makes mathematically limited sense using the standard rules for matrix multiplication: By treating vectors as $n \times 1$ matrices (which we usually do), we can use the matrix multiplication as defined in (2.12). However, then the dimensions of the vectors do not match. Only the following multiplications for vectors are defined: $\boldsymbol{ab}^\top \in \mathbb{R}^{n \times n}$ (outer product), $\boldsymbol{a}^\top \boldsymbol{b} \in \mathbb{R}$ (inner/scalar/dot product). ♢

Example 2.11 (Vector Spaces)
Let us have a look at some important examples.

- $\mathcal{V} = \mathbb{R}^n, n \in \mathbb{N}$ is a vector space with operations defined as follows:
  - Addition: $\boldsymbol{x} + \boldsymbol{y} = (x_1, \dots, x_n) + (y_1, \dots, y_n) = (x_1 + y_1, \dots, x_n + y_n)$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$
  - Multiplication by scalars: $\lambda \boldsymbol{x} = \lambda(x_1, \dots, x_n) = (\lambda x_1, \dots, \lambda x_n)$ for all $\lambda \in \mathbb{R}, \boldsymbol{x} \in \mathbb{R}^n$
- $\mathcal{V} = \mathbb{R}^{m \times n}, m, n \in \mathbb{N}$ is a vector space with
  - Addition: $\boldsymbol{A} + \boldsymbol{B} = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$ is defined elementwise for all $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{V}$
  - Multiplication by scalars: $\lambda \boldsymbol{A} = \begin{bmatrix} \lambda a_{11} & \cdots & \lambda a_{1n} \\ \vdots & & \vdots \\ \lambda a_{m1} & \cdots & \lambda a_{mn} \end{bmatrix}$ as defined in Section 2.2. Remember that $\mathbb{R}^{m \times n}$ is equivalent to $\mathbb{R}^{mn}$.
- $\mathcal{V} = \mathbb{C}$, with the standard definition of addition of complex numbers.

Remark. In the following, we will denote a vector space $(\mathcal{V}, +, \cdot)$ by V when $+$ and $\cdot$ are the standard vector addition and scalar multiplication. Moreover, we will use the notation $\boldsymbol{x} \in V$ for vectors in $\mathcal{V}$ to simplify notation. ♢

Remark. The vector spaces $\mathbb{R}^n, \mathbb{R}^{n\times 1}, \mathbb{R}^{1\times n}$ are only different in the way we write vectors. In the following, we will not make a distinction between $\mathbb{R}^n$ and $\mathbb{R}^{n\times 1}$, which allows us to write n-tuples as *column vectors*

column vectors

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.63}$$

This simplifies the notation regarding vector space operations. However, we do distinguish between $\mathbb{R}^{n\times 1}$ and $\mathbb{R}^{1\times n}$ (the *row vectors*) to avoid confusion with matrix multiplication. By default we write $\boldsymbol{x}$ to denote a column vector, and a row vector is denoted by $\boldsymbol{x}^\top$, the *transpose* of $\boldsymbol{x}$. ♢

row vectors

transpose

2.4.3 Vector Subspaces

In the following, we will introduce vector subspaces. Intuitively, they are sets contained in the original vector space with the property that when we perform vector space operations on elements within this subspace, we will never leave it. In this sense, they are "closed".

Definition 2.9 (Vector Subspace)**.** Let $V = (\mathcal{V}, +, \cdot)$ be a vector space and $\mathcal{U} \subseteq \mathcal{V}, \mathcal{U} \neq \emptyset$. Then $U = (\mathcal{U}, +, \cdot)$ is called *vector subspace* of V (or *linear subspace*) if U is a vector space with the vector space operations $+$ and $\cdot$ restricted to $\mathcal{U} \times \mathcal{U}$ and $\mathbb{R} \times \mathcal{U}$. We write $U \subseteq V$ to denote a subspace U of V.

vector subspace

linear subspace

If $\mathcal{U} \subseteq \mathcal{V}$ and V is a vector space, then U naturally inherits many properties directly from V because they are true for all $\boldsymbol{x} \in \mathcal{V}$, and in particular for all $\boldsymbol{x} \in \mathcal{U} \subseteq \mathcal{V}$. This includes the Abelian group properties, the distributivity, the associativity and the neutral element. To determine whether $(\mathcal{U}, +, \cdot)$ is a subspace of V we still do need to show

1. $\mathcal{U} \neq \emptyset$, in particular: $\mathbf{0} \in \mathcal{U}$
2. Closure of U:
   1. With respect to the outer operation: $\forall \lambda \in \mathbb{R} \, \forall \boldsymbol{x} \in \mathcal{U} : \lambda \boldsymbol{x} \in \mathcal{U}$.
   2. With respect to the inner operation: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{U} : \boldsymbol{x} + \boldsymbol{y} \in \mathcal{U}$.

Example 2.12 (Vector Subspaces)
Let us have a look at some subspaces.

- For every vector space V the trivial subspaces are V itself and $\{\mathbf{0}\}$.

- Only example D in Figure 2.5 is a subspace of $\mathbb{R}^2$ (with the usual inner/outer operations). In A and C, the closure property is violated; B does not contain $\mathbf{0}$.
- The solution set of a homogeneous linear equation system $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$ with n unknowns $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$ is a subspace of $\mathbb{R}^n$.
- The solution of an inhomogeneous equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{b} \neq \mathbf{0}$ is not a subspace of $\mathbb{R}^n$.
- The intersection of arbitrarily many subspaces is a subspace itself.

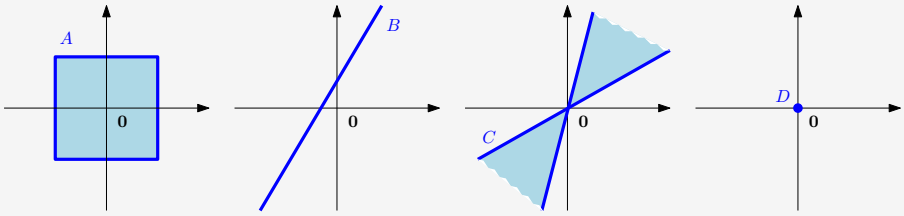


Figure 2.5 Not all subsets of $\mathbb{R}^2$ are subspaces. In A and C, the closure property is violated; B does not contain $\mathbf{0}$. Only D is a subspace.

Remark. Every subspace $U \subseteq (\mathbb{R}^n, +, \cdot)$ is the solution space of a homogeneous linear equation system $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$. ♢

2.5 Linear Independence

So far, we looked at vector spaces and some of their properties, e.g., closure. Now, we will look at what we can do with vectors (elements of the vector space). In particular, we can add vectors together and multiply them with scalars. The closure property guarantees that we end up with another vector in the same vector space. Let us formalize this:

Definition 2.10 (Linear Combination)**.** Consider a vector space V and a finite number of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in V$. Then, every $\boldsymbol{v} \in V$ of the form

$$\boldsymbol{v} = \lambda_1 \boldsymbol{x}_1 + \cdots + \lambda_k \boldsymbol{x}_k = \sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i \in V \tag{2.64}$$

with $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$ is a *linear combination* of the vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$. linear combination

The $\mathbf{0}$-vector can always be written as the linear combination of k vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ because $\mathbf{0} = \sum_{i=1}^{k} 0\boldsymbol{x}_i$ is always true. In the following, we are interested in non-trivial linear combinations of a set of vectors to represent $\mathbf{0}$, i.e., linear combinations of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ where not all coefficients λ_i in (2.64) are 0.

Definition 2.11 (Linear (In)dependence)**.** Let us consider a vector space V with $k \in \mathbb{N}$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in V$. If there is a non-trivial linear combination, such that $\mathbf{0} = \sum_{i=1}^{k} \lambda_i \boldsymbol{x}_i$ with at least one $\lambda_i \neq 0$, the vectors

$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are *linearly dependent*. If only the trivial solution exists, i.e., $\lambda_1 = \ldots = \lambda_k = 0$ the vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are *linearly independent*.

linearly dependent

linearly independent

Linear independence is one of the most important concepts in linear algebra. Intuitively, a set of linearly independent vectors are vectors that have no redundancy, i.e., if we remove any of those vectors from the set, we will lose something. Throughout the next sections, we will formalize this intuition more.

Example 2.13 (Linearly Dependent Vectors)
A geographic example may help to clarify the concept of linear independence. A person in Nairobi (Kenya) describing where Kigali (Rwanda) is might say "You can get to Kigali by first going $506\,\mathrm{km}$ Northwest to Kampala (Uganda) and then $374\,\mathrm{km}$ Southwest.". This is sufficient information to describe the location of Kigali because the geographic coordinate system may be considered a two-dimensional vector space (ignoring altitude and the Earth's surface). The person may add "It is about $751\,\mathrm{km}$ West of here." Although this last statement is true, it is not necessary to find Kigali given the previous information (see Figure 2.6 for an illustration).

In this example, we make crude approximations to cardinal directions.

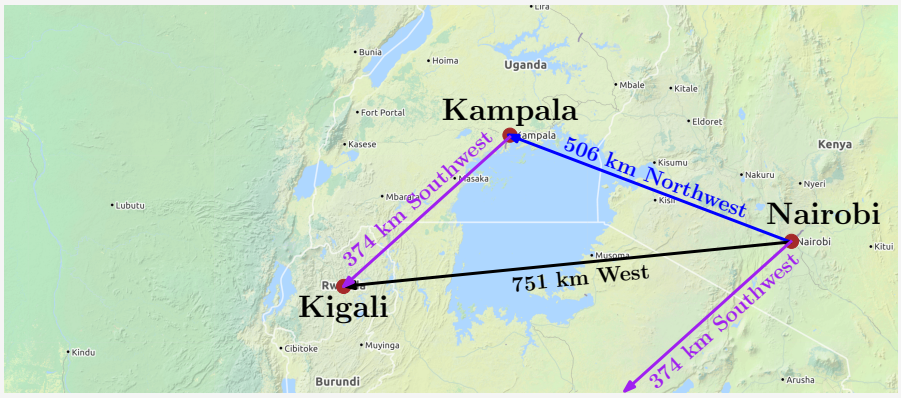


Figure 2.6 Geographic example (with crude approximations to cardinal directions) of linearly dependent vectors in a two-dimensional space (plane).

In this example, the "$506\,\mathrm{km}$ Northwest" vector (blue) and the "$374\,\mathrm{km}$ Southwest" vector (purple) are linearly independent. This means the Southwest vector cannot be described in terms of the Northwest vector, and vice versa. However, the third "$751\,\mathrm{km}$ West" vector (black) is a linear combination of the other two vectors, and it makes the set of vectors linearly dependent.

Remark. The following properties are useful to find out whether vectors are linearly independent.

- k vectors are either linearly dependent or linearly independent. There is no third option.

- If at least one of the vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ is $\mathbf{0}$ then they are linearly dependent. The same holds if two vectors are identical.
- The vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k : \boldsymbol{x}_i \neq \mathbf{0}, i = 1, \ldots, k\}$, $k \geqslant 2$, are linearly dependent if and only if (at least) one of them is a linear combination of the others. In particular, if one vector is a multiple of another vector, i.e., $\boldsymbol{x}_i = \lambda \boldsymbol{x}_j, \ \lambda \in \mathbb{R}$ then the set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k : \boldsymbol{x}_i \neq \mathbf{0}, i = 1, \ldots, k\}$ is linearly dependent.
- A practical way of checking whether vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in V$ are linearly independent is to use Gaussian elimination: Write all vectors as columns of a matrix $\boldsymbol{A}$ and perform Gaussian elimination until the matrix is in row echelon form (the reduced row echelon form is not necessary here).
  - The pivot columns indicate the vectors, which are linearly independent of the vectors on the left. Note that there is an ordering of vectors when the matrix is built.
  - The non-pivot columns can be expressed as linear combinations of the pivot columns on their left. For instance, the row echelon form

$$\begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{2.65}$$

    tells us that the first and third column are pivot columns. The second column is a non-pivot column because it is 3 times the first column.

  All column vectors are linearly independent if and only if all columns are pivot columns. If there is at least one non-pivot column, the columns (and, therefore, the corresponding vectors) are linearly dependent.

♢

Example 2.14
Consider $\mathbb{R}^4$ with

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 2 \\ -3 \\ 4 \end{bmatrix}, \quad \boldsymbol{x}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \end{bmatrix}, \quad \boldsymbol{x}_3 = \begin{bmatrix} -1 \\ -2 \\ 1 \\ 1 \end{bmatrix}. \tag{2.66}$$

To check whether they are linearly dependent, we follow the general approach and solve

$$\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 + \lambda_3 \boldsymbol{x}_3 = \lambda_1 \begin{bmatrix} 1 \\ 2 \\ -3 \\ 4 \end{bmatrix} + \lambda_2 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \end{bmatrix} + \lambda_3 \begin{bmatrix} -1 \\ -2 \\ 1 \\ 1 \end{bmatrix} = \mathbf{0} \tag{2.67}$$

for $\lambda_1, \ldots, \lambda_3$. We write the vectors $\boldsymbol{x}_i$, $i = 1, 2, 3$, as the columns of a matrix and apply elementary row operations until we identify the pivot columns:

$$\begin{bmatrix} 1 & 1 & -1 \\ 2 & 1 & -2 \\ -3 & 0 & 1 \\ 4 & 2 & 1 \end{bmatrix} \rightsquigarrow \cdots \rightsquigarrow \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.68}$$

Here, every column of the matrix is a pivot column. Therefore, there is no non-trivial solution, and we require $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 0$ to solve the equation system. Hence, the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ are linearly independent.

Remark. Consider a vector space V with k linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ and m linear combinations

$$\begin{aligned} \boldsymbol{x}_1 &= \sum_{i=1}^{k} \lambda_{i1} \boldsymbol{b}_i \,, \\ &\vdots \\ \boldsymbol{x}_m &= \sum_{i=1}^{k} \lambda_{im} \boldsymbol{b}_i \,. \end{aligned} \tag{2.69}$$

Defining $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k]$ as the matrix whose columns are the linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$, we can write

$$\boldsymbol{x}_j = \boldsymbol{B}\boldsymbol{\lambda}_j \,, \quad \boldsymbol{\lambda}_j = \begin{bmatrix} \lambda_{1j} \\ \vdots \\ \lambda_{kj} \end{bmatrix} \,, \quad j = 1, \ldots, m \,, \tag{2.70}$$

in a more compact form.

We want to test whether $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are linearly independent. For this purpose, we follow the general approach of testing when $\sum_{j=1}^{m} \psi_j \boldsymbol{x}_j = \boldsymbol{0}$. With (2.70), we obtain

$$\sum_{j=1}^{m} \psi_j \boldsymbol{x}_j = \sum_{j=1}^{m} \psi_j \boldsymbol{B}\boldsymbol{\lambda}_j = \boldsymbol{B} \sum_{j=1}^{m} \psi_j \boldsymbol{\lambda}_j \,. \tag{2.71}$$

This means that $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ are linearly independent if and only if the column vectors $\{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m\}$ are linearly independent.

♢

Remark. In a vector space V, m linear combinations of k vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are linearly dependent if $m > k$. ♢

Example 2.15
Consider a set of linearly independent vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_4 \in \mathbb{R}^n$ and

$$\begin{array}{rcrcrcrcl} \boldsymbol{x}_1 &=& \boldsymbol{b}_1 &-& 2\boldsymbol{b}_2 &+& \boldsymbol{b}_3 &-& \boldsymbol{b}_4 \\ \boldsymbol{x}_2 &=& -4\boldsymbol{b}_1 &-& 2\boldsymbol{b}_2 & & &+& 4\boldsymbol{b}_4 \\ \boldsymbol{x}_3 &=& 2\boldsymbol{b}_1 &+& 3\boldsymbol{b}_2 &-& \boldsymbol{b}_3 &-& 3\boldsymbol{b}_4 \\ \boldsymbol{x}_4 &=& 17\boldsymbol{b}_1 &-& 10\boldsymbol{b}_2 &+& 11\boldsymbol{b}_3 &+& \boldsymbol{b}_4 \end{array} \tag{2.72}$$

Are the vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4 \in \mathbb{R}^n$ linearly independent? To answer this question, we investigate whether the column vectors

$$\left\{ \begin{bmatrix} 1 \\ -2 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -4 \\ -2 \\ 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ -1 \\ -3 \end{bmatrix}, \begin{bmatrix} 17 \\ -10 \\ 11 \\ 1 \end{bmatrix} \right\} \tag{2.73}$$

are linearly independent. The reduced row echelon form of the corresponding linear equation system with coefficient matrix

$$\boldsymbol{A} = \begin{bmatrix} 1 & -4 & 2 & 17 \\ -2 & -2 & 3 & -10 \\ 1 & 0 & -1 & 11 \\ -1 & 4 & -3 & 1 \end{bmatrix} \tag{2.74}$$

is given as

$$\begin{bmatrix} 1 & 0 & 0 & -7 \\ 0 & 1 & 0 & -15 \\ 0 & 0 & 1 & -18 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.75}$$

We see that the corresponding linear equation system is non-trivially solvable: The last column is not a pivot column, and $\boldsymbol{x}_4 = -7\boldsymbol{x}_1 - 15\boldsymbol{x}_2 - 18\boldsymbol{x}_3$. Therefore, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ are linearly dependent as $\boldsymbol{x}_4$ can be expressed as a linear combination of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_3$.

2.6 Basis and Rank

In a vector space V, we are particularly interested in sets of vectors A that possess the property that any vector $\boldsymbol{v} \in V$ can be obtained by a linear combination of vectors in A. These vectors are special vectors, and in the following, we will characterize them.

2.6.1 Generating Set and Basis

Definition 2.12 (Generating Set and Span)**.** Consider a vector space $V = (\mathcal{V}, +, \cdot)$ and set of vectors $\mathcal{A} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\} \subseteq \mathcal{V}$. If every vector $\boldsymbol{v} \in$

$\mathcal{V}$ can be expressed as a linear combination of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$, $\mathcal{A}$ is called a *generating set* of V. The set of all linear combinations of vectors in $\mathcal{A}$ is called the *span* of $\mathcal{A}$. If $\mathcal{A}$ spans the vector space V, we write $V = \operatorname{span}[\mathcal{A}]$ or $V = \operatorname{span}[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k]$.

generating set
span

Generating sets are sets of vectors that span vector (sub)spaces, i.e., every vector can be represented as a linear combination of the vectors in the generating set. Now, we will be more specific and characterize the smallest generating set that spans a vector (sub)space.

Definition 2.13 (Basis)**.** Consider a vector space $V = (\mathcal{V}, +, \cdot)$ and $\mathcal{A} \subseteq \mathcal{V}$. A generating set $\mathcal{A}$ of V is called *minimal* if there exists no smaller set $\tilde{\mathcal{A}} \subseteq \mathcal{A} \subseteq \mathcal{V}$ that spans V. Every linearly independent generating set of V is minimal and is called a *basis* of V.

minimal
basis

Let $V = (\mathcal{V}, +, \cdot)$ be a vector space and $\mathcal{B} \subseteq \mathcal{V}, \mathcal{B} \neq \emptyset$. Then, the following statements are equivalent:

A basis is a minimal generating set and a maximal linearly independent set of vectors.

- $\mathcal{B}$ is a basis of V
- $\mathcal{B}$ is a minimal generating set
- $\mathcal{B}$ is a maximal linearly independent set of vectors in V, i.e., adding any other vector to this set will make it linearly dependent.
- Every vector $\boldsymbol{x} \in V$ is a linear combination of vectors from $\mathcal{B}$, and every linear combination is unique, i.e., with

$$\boldsymbol{x} = \sum_{i=1}^{k} \lambda_i \boldsymbol{b}_i = \sum_{i=1}^{k} \psi_i \boldsymbol{b}_i \tag{2.76}$$

and $\lambda_i, \psi_i \in \mathbb{R}$, $\boldsymbol{b}_i \in \mathcal{B}$ it follows that $\lambda_i = \psi_i,\ i = 1, \ldots, k$.

Example 2.16

canonical/standard basis

- In $\mathbb{R}^3$, the *canonical/standard basis* is

$$\mathcal{B} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}. \tag{2.77}$$

- Different bases in $\mathbb{R}^3$ are

$$\mathcal{B}_1 = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}, \mathcal{B}_2 = \left\{ \begin{bmatrix} 0.5 \\ 0.8 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 1.8 \\ 0.3 \\ 0.3 \end{bmatrix}, \begin{bmatrix} -2.2 \\ -1.3 \\ 3.5 \end{bmatrix} \right\}. \tag{2.78}$$

- The set

$$\mathcal{A} = \left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ -4 \end{bmatrix} \right\} \tag{2.79}$$

is linearly independent, but not a generating set (and no basis) of $\mathbb{R}^4$: For instance, the vector $[1, 0, 0, 0]^\top$ cannot be obtained by a linear combination of elements in $\mathcal{A}$.

Remark. Every vector space V possesses a basis $\mathcal{B}$. The examples above show that there can be many bases of a vector space V, i.e., there is no unique basis. However, all bases possess the same number of elements, the *basis vectors*. ♢

basis vectors

We only consider finite-dimensional vector spaces V. In this case, the *dimension* of V is the number of basis vectors, and we write $\dim(V)$. If $U \subseteq V$ is a subspace of V then $\dim(U) \leqslant \dim(V)$ and $\dim(U) = \dim(V)$ if and only if $U = V$. Intuitively, the dimension of a vector space can be thought of as the number of independent directions in this vector space.

The dimension of a vector space corresponds to the number of basis vectors.

dimension

Remark. A basis of a subspace $U = \mathrm{span}[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m] \subseteq \mathbb{R}^n$ can be found by executing the following steps:

1. Write the spanning vectors as columns of a matrix $\boldsymbol{A}$
2. Determine the row echelon form of $\boldsymbol{A}$.
3. The spanning vectors associated with the pivot columns are a basis of U.

♢

Example 2.17 (Determining a Basis)
For a vector subspace $U \subseteq \mathbb{R}^5$, spanned by the vectors

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \\ -1 \\ -1 \end{bmatrix}, \quad \boldsymbol{x}_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \\ 2 \\ -2 \end{bmatrix}, \quad \boldsymbol{x}_3 = \begin{bmatrix} 3 \\ -4 \\ 3 \\ 5 \\ -3 \end{bmatrix}, \quad \boldsymbol{x}_4 = \begin{bmatrix} -1 \\ 8 \\ -5 \\ -6 \\ 1 \end{bmatrix} \in \mathbb{R}^5, \tag{2.80}$$

we are interested in finding out which vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ are a basis for U. For this, we need to check whether $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ are linearly independent. Therefore, we need to solve

$$\sum_{i=1}^{4} \lambda_i \boldsymbol{x}_i = \boldsymbol{0}\,, \tag{2.81}$$

which leads to a homogeneous equation system with matrix

$$\big[\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\big] = \begin{bmatrix} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{bmatrix}. \tag{2.82}$$

With the basic transformation rules for systems of linear equations, we obtain the reduced row echelon form

$$\begin{bmatrix} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{bmatrix} \rightsquigarrow \cdots \rightsquigarrow \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

From this reduced-row echelon form we see that $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_4$ belong to the pivot columns, and, therefore, are linearly independent (because the linear equation system $\lambda_1\boldsymbol{x}_1 + \lambda_2\boldsymbol{x}_2 + \lambda_4\boldsymbol{x}_4 = \boldsymbol{0}$ can only be solved with $\lambda_1 = \lambda_2 = \lambda_4 = 0$). Therefore, $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_4\}$ is a basis of U.

2.6.2 Rank

The number of linearly independent columns of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ equals the number of linearly independent rows and is called the *rank* of $\boldsymbol{A}$ and is denoted by $\mathrm{rk}(\boldsymbol{A})$.

rank

Remark. The rank of a matrix has some important properties:

- $\mathrm{rk}(\boldsymbol{A}) = \mathrm{rk}(\boldsymbol{A}^\top)$, i.e., the column rank equals the row rank.
- The columns of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ span a subspace $U \subseteq \mathbb{R}^m$ with $\dim(U) = \mathrm{rk}(\boldsymbol{A})$. Later, we will call this subspace the *image* or *range*. A basis of U can be found by applying Gaussian elimination to $\boldsymbol{A}$ to identify the pivot columns.
- The rows of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ span a subspace $W \subseteq \mathbb{R}^n$ with $\dim(W) = \mathrm{rk}(\boldsymbol{A})$. A basis of W can be found by applying Gaussian elimination to $\boldsymbol{A}^\top$.
- For all $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ holds: $\boldsymbol{A}$ is regular (invertible) if and only if $\mathrm{rk}(\boldsymbol{A}) = n$.
- For all $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and all $\boldsymbol{b} \in \mathbb{R}^m$ it holds that the linear equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can be solved if and only if $\mathrm{rk}(\boldsymbol{A}) = \mathrm{rk}(\boldsymbol{A}|\boldsymbol{b})$, where $\boldsymbol{A}|\boldsymbol{b}$ denotes the augmented system.
- For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ the subspace of solutions for $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ possesses dimension $n - \mathrm{rk}(\boldsymbol{A})$. Later, we will call this subspace the *kernel* or the *null space*.

kernel
null space

- A matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ has *full rank* if its rank equals the largest possible rank for a matrix of the same dimensions. This means that the rank of a full-rank matrix is the lesser of the number of rows and columns, i.e., $\mathrm{rk}(\boldsymbol{A}) = \min(m, n)$. A matrix is said to be *rank deficient* if it does not have full rank.

full rank

rank deficient

♢

Example 2.18 (Rank)

- $\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$. $\boldsymbol{A}$ possesses two linearly independent rows (and columns). Therefore, $\mathrm{rk}(\boldsymbol{A}) = 2$.
- $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ We use Gaussian elimination to determine the rank:

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \rightsquigarrow \cdots \rightsquigarrow \begin{bmatrix} 1 & 2 & 1 \\ 0 & -1 & 3 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.83}$$

  Here, we see that the number of linearly independent rows and columns is 2, such that $\mathrm{rk}(\boldsymbol{A}) = 2$.

2.7 Linear Mappings

In the following, we will study mappings on vector spaces that preserve their structure. In the beginning of the chapter, we said that vectors are objects that can be added together and multiplied by a scalar, and the resulting object is still a vector. This property we wish to preserve when applying the mapping: Consider two real vector spaces V, W. A mapping $\Phi : V \to W$ preserves the structure of the vector space if

$$\Phi(\boldsymbol{x} + \boldsymbol{y}) = \Phi(\boldsymbol{x}) + \Phi(\boldsymbol{y}) \tag{2.84}$$
$$\Phi(\lambda\boldsymbol{x}) = \lambda\Phi(\boldsymbol{x}) \tag{2.85}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in V$ and $\lambda \in \mathbb{R}$. We can summarize this in the following definition:

Definition 2.14 (Linear Mapping)**.** For vector spaces V, W, a mapping $\Phi : V \to W$ is called a *linear mapping* (or *vector space homomorphism/ linear transformation*) if

linear mapping
vector space homomorphism
linear transformation

$$\forall \boldsymbol{x}, \boldsymbol{y} \in V \, \forall \lambda, \psi \in \mathbb{R} : \Phi(\lambda\boldsymbol{x} + \psi\boldsymbol{y}) = \lambda\Phi(\boldsymbol{x}) + \psi\Phi(\boldsymbol{y}) \,. \tag{2.86}$$

Before we continue, we will briefly introduce special mappings.

Definition 2.15 (Injective, Surjective, Bijective)**.** Consider a mapping $\Phi : \mathcal{V} \to \mathcal{W}$, where $\mathcal{V}, \mathcal{W}$ can be arbitrary sets. Then Φ is called

injective
surjective
bijective

- *injective* if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{V} : \Phi(\boldsymbol{x}) = \Phi(\boldsymbol{y}) \implies \boldsymbol{x} = \boldsymbol{y}$.
- *surjective* if $\Phi(\mathcal{V}) = \mathcal{W}$.
- *bijective* if it is injective and surjective.

If Φ is injective then it can also be "undone", i.e., there exists a mapping $\Psi : \mathcal{W} \to \mathcal{V}$ so that $\Psi \circ \Phi(\boldsymbol{x}) = \boldsymbol{x}$. If Φ is surjective then every element in $\mathcal{W}$ can be "reached" from $\mathcal{V}$ using Φ.

With these definitions, we introduce the following special cases of linear mappings between vector spaces V and W:

Isomorphism
Endomorphism
Automorphism
identity mapping

- *Isomorphism*: $\Phi : V \to W$ linear and bijective
- *Endomorphism*: $\Phi : V \to V$ linear
- *Automorphism*: $\Phi : V \to V$ linear and bijective
- We define $\mathrm{id}_V : V \to V$, $\boldsymbol{x} \mapsto \boldsymbol{x}$ as the *identity mapping* in V.

Example 2.19 (Homomorphism)
The mapping $\Phi : \mathbb{R}^2 \to \mathbb{C},\ \Phi(\boldsymbol{x}) = x_1 + ix_2$, is a homomorphism:

$$\begin{aligned}\Phi\left(\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}y_1\\y_2\end{bmatrix}\right) &= (x_1 + y_1) + i(x_2 + y_2) = x_1 + ix_2 + y_1 + iy_2 \\ &= \Phi\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) + \Phi\left(\begin{bmatrix}y_1\\y_2\end{bmatrix}\right) \\ \Phi\left(\lambda\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) &= \lambda x_1 + \lambda i x_2 = \lambda(x_1 + ix_2) = \lambda\Phi\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right).\end{aligned} \tag{2.87}$$

This also justifies why complex numbers can be represented as tuples in $\mathbb{R}^2$: There is a bijective linear mapping that converts the elementwise addition of tuples in $\mathbb{R}^2$ into the set of complex numbers with the corresponding addition. Note that we only showed linearity, but not the bijection.

Theorem 2.16. *Finite-dimensional vector spaces V and W are isomorphic if and only if* $\dim(V) = \dim(W)$.

Theorem 2.16 states that there exists a linear, bijective mapping between two vector spaces of the same dimension. Intuitively, this means that vector spaces of the same dimension are kind of the same thing as they can be transformed into each other without incurring any loss.

Theorem 2.16 also gives us the justification to treat $\mathbb{R}^{m\times n}$ (the vector space of $m \times n$-matrices) and $\mathbb{R}^{mn}$ (the vector space of vectors of length mn) the same as their dimensions are mn, and there exists a linear, bijective mapping that transforms one into the other.

Remark. Consider vector spaces V, W, X. Then:

- For linear mappings $\Phi : V \to W$ and $\Psi : W \to X$ the mapping $\Psi \circ \Phi : V \to X$ is also linear.
- If $\Phi : V \to W$ is an isomorphism then $\Phi^{-1} : W \to V$ is an isomorphism, too.
- If $\Phi : V \to W,\ \Psi : V \to W$ are linear then $\Phi + \Psi$ and $\lambda\Phi,\ \lambda \in \mathbb{R}$, are linear, too.

♢

2.7.1 Matrix Representation of Linear Mappings

Any n-dimensional vector space is isomorphic to $\mathbb{R}^n$ (Theorem 2.16). We consider a basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ of an n-dimensional vector space V. In the following, the order of the basis vectors will be important. Therefore, we write

$$B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \tag{2.88}$$

and call this n-tuple an *ordered basis* of V.

ordered basis

Remark (Notation). We are at the point where notation gets a bit tricky. Therefore, we summarize some parts here. $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ is an ordered basis, $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ is an (unordered) basis, and $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ is a matrix whose columns are the vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$. ♢

Definition 2.17 (Coordinates)**.** Consider a vector space V and an ordered basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of V. For any $\boldsymbol{x} \in V$ we obtain a unique representation (linear combination)

$$\boldsymbol{x} = \alpha_1 \boldsymbol{b}_1 + \ldots + \alpha_n \boldsymbol{b}_n \tag{2.89}$$

of $\boldsymbol{x}$ with respect to B. Then $\alpha_1, \ldots, \alpha_n$ are the *coordinates* of $\boldsymbol{x}$ with respect to B, and the vector

coordinates

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \in \mathbb{R}^n \tag{2.90}$$

is the *coordinate vector/coordinate representation* of $\boldsymbol{x}$ with respect to the ordered basis B.

coordinate vector
coordinate representation

Remark. Intuitively, the basis vectors can be thought of as being equipped with units (including common units such as "kilograms" or "seconds"). Let us have a look at a geometric vector $\boldsymbol{x} \in \mathbb{R}^2$ with coordinates $[2, 3]^\top$ with respect to the standard basis $\boldsymbol{e}_1, \boldsymbol{e}_2$ in $\mathbb{R}^2$. This means, we can write $\boldsymbol{x} = 2\boldsymbol{e}_1 + 3\boldsymbol{e}_2$. However, we do not have to choose the standard basis to represent this vector. If we use the basis vectors $\boldsymbol{b}_1 = [1, -1]^\top, \boldsymbol{b}_2 = [1, 1]^\top$ we will obtain the coordinates $\frac{1}{2}[-1, 5]^\top$ to represent the same vector (see Figure 2.7). ♢

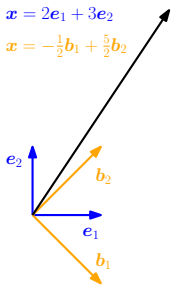


Figure 2.7 Different coordinate representations of a vector $\boldsymbol{x}$, depending on the choice of basis.

Remark. For an n-dimensional vector space V and an ordered basis B of V, the mapping $\Phi : \mathbb{R}^n \to V$, $\Phi(\boldsymbol{e}_i) = \boldsymbol{b}_i$, $i = 1, \ldots, n$, is linear (and because of Theorem 2.16 an isomorphism), where $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$ is the standard basis of $\mathbb{R}^n$.

♢

Now we are ready to make an explicit connection between matrices and linear mappings between finite-dimensional vector spaces.

Definition 2.18 (Transformation matrix)**.** Consider vector spaces V, W with corresponding (ordered) bases $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ and $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m)$. Moreover, we consider a linear mapping $\Phi : V \to W$. For $j \in \{1, \ldots, n\}$

$$\Phi(\boldsymbol{b}_j) = \alpha_{1j}\boldsymbol{c}_1 + \cdots + \alpha_{mj}\boldsymbol{c}_m = \sum_{i=1}^{m} \alpha_{ij}\boldsymbol{c}_i \tag{2.91}$$

is the unique representation of $\Phi(\boldsymbol{b}_j)$ with respect to C. Then, we call the $m \times n$-matrix $\boldsymbol{A}_\Phi$ whose elements are given by

$$A_\Phi(i, j) = \alpha_{ij} \tag{2.92}$$

transformation matrix

the *transformation matrix* of Φ (with respect to the ordered bases B of V and C of W).

The coordinates of $\Phi(\boldsymbol{b}_j)$ with respect to the ordered basis C of W are the j-th column of $\boldsymbol{A}_\Phi$. Consider (finite-dimensional) vector spaces V, W with ordered bases B, C and a linear mapping $\Phi : V \to W$ with transformation matrix $\boldsymbol{A}_\Phi$. If $\hat{\boldsymbol{x}}$ is the coordinate vector of $\boldsymbol{x} \in V$ with respect to B and $\hat{\boldsymbol{y}}$ the coordinate vector of $\boldsymbol{y} = \Phi(\boldsymbol{x}) \in W$ with respect to C, then

$$\hat{\boldsymbol{y}} = \boldsymbol{A}_\Phi \hat{\boldsymbol{x}}\,. \tag{2.93}$$

This means that the transformation matrix can be used to map coordinates with respect to an ordered basis in V to coordinates with respect to an ordered basis in W.

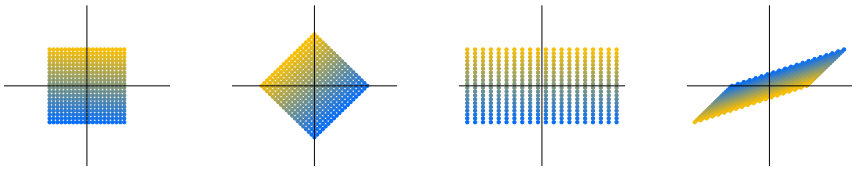
Example 2.20 (Transformation Matrix)
Consider a homomorphism $\Phi : V \to W$ and ordered bases $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_3)$ of V and $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_4)$ of W. With

$$\begin{aligned}
\Phi(\boldsymbol{b}_1) &= \boldsymbol{c}_1 - \boldsymbol{c}_2 + 3\boldsymbol{c}_3 - \boldsymbol{c}_4 \\
\Phi(\boldsymbol{b}_2) &= 2\boldsymbol{c}_1 + \boldsymbol{c}_2 + 7\boldsymbol{c}_3 + 2\boldsymbol{c}_4 \\
\Phi(\boldsymbol{b}_3) &= 3\boldsymbol{c}_2 + \boldsymbol{c}_3 + 4\boldsymbol{c}_4
\end{aligned} \tag{2.94}$$

the transformation matrix $\boldsymbol{A}_\Phi$ with respect to B and C satisfies $\Phi(\boldsymbol{b}_k) = \sum_{i=1}^{4} \alpha_{ik}\boldsymbol{c}_i$ for $k = 1, \ldots, 3$ and is given as

$$\boldsymbol{A}_\Phi = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3] = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 1 & 3 \\ 3 & 7 & 1 \\ -1 & 2 & 4 \end{bmatrix}, \tag{2.95}$$

where the $\boldsymbol{\alpha}_j,\ j = 1, 2, 3$, are the coordinate vectors of $\Phi(\boldsymbol{b}_j)$ with respect to C.



(a) Original data. (b) Rotation by $45°$. (c) Stretch along the horizontal axis. (d) General linear mapping.

Figure 2.8 Three examples of linear transformations of the vectors shown as dots in (a). (b) Rotation by $45°$; (c) Stretching of the horizontal coordinates by 2; (d) Combination of reflection, rotation and stretching.

Example 2.21 (Linear Transformations of Vectors)
We consider three linear transformations of a set of vectors in $\mathbb{R}^2$ with the transformation matrices

$$\boldsymbol{A}_1 = \begin{bmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{A}_3 = \frac{1}{2}\begin{bmatrix} 3 & -1 \\ 1 & -1 \end{bmatrix}. \tag{2.96}$$

Figure 2.8 gives three examples of linear transformations of a set of vectors. Figure 2.8(a) shows 400 vectors in $\mathbb{R}^2$, each of which is represented by a dot at the corresponding (x_1, x_2)-coordinates. The vectors are arranged in a square. When we use matrix $\boldsymbol{A}_1$ in (2.96) to linearly transform each of these vectors, we obtain the rotated square in Figure 2.8(b). If we apply the linear mapping represented by $\boldsymbol{A}_2$, we obtain the rectangle in Figure 2.8(c) where each x_1-coordinate is stretched by 2. Figure 2.8(d) shows the original square from Figure 2.8(a) when linearly transformed using $\boldsymbol{A}_3$, which is a combination of a reflection, a rotation and a stretch.

2.7.2 Basis Change

In the following, we will have a closer look at how transformation matrices of a linear mapping $\Phi : V \to W$ change if we change the bases in V and W. Consider two ordered bases

$$B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n), \quad \tilde{B} = (\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n) \tag{2.97}$$

of V and two ordered bases

$$C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m), \quad \tilde{C} = (\tilde{\boldsymbol{c}}_1, \ldots, \tilde{\boldsymbol{c}}_m) \tag{2.98}$$

of W. Moreover, $\boldsymbol{A}_\Phi \in \mathbb{R}^{m \times n}$ is the transformation matrix of the linear mapping $\Phi : V \to W$ with respect to the bases B and C, and $\tilde{\boldsymbol{A}}_\Phi \in \mathbb{R}^{m \times n}$ is the corresponding transformation mapping with respect to $\tilde{B}$ and $\tilde{C}$. In the following, we will investigate how $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ are related, i.e., how/whether we can transform $\boldsymbol{A}_\Phi$ into $\tilde{\boldsymbol{A}}_\Phi$ if we choose to perform a basis change from B, C to $\tilde{B}, \tilde{C}$.

Remark. We effectively get different coordinate representations of the identity mapping id_V. In the context of Figure 2.7, this would mean to map coordinates with respect to $\boldsymbol{e}_1, \boldsymbol{e}_2$ onto coordinates with respect to $\boldsymbol{b}_1, \boldsymbol{b}_2$ without changing the vector $\boldsymbol{x}$. By changing the basis and correspondingly the representation of vectors, the transformation matrix with respect to this new basis can have a particularly simple form that allows for straightforward computation. ♢

Example 2.22 (Basis Change)
Consider a transformation matrix

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{2.99}$$

with respect to the canonical basis in $\mathbb{R}^2$. If we define a new basis

$$B = (\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}) \tag{2.100}$$

we obtain a diagonal transformation matrix

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.101}$$

with respect to B, which is easier to work with than $\boldsymbol{A}$.

In the following, we will look at mappings that transform coordinate vectors with respect to one basis into coordinate vectors with respect to a different basis. We will state our main result first and then provide an explanation.

Theorem 2.19 (Basis Change)**.** *For a linear mapping $\Phi : V \to W$, ordered bases*

$$B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n), \quad \tilde{B} = (\tilde{\boldsymbol{b}}_1, \ldots, \tilde{\boldsymbol{b}}_n) \tag{2.102}$$

of V and

$$C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m), \quad \tilde{C} = (\tilde{\boldsymbol{c}}_1, \ldots, \tilde{\boldsymbol{c}}_m) \tag{2.103}$$

of W, and a transformation matrix $\boldsymbol{A}_\Phi$ of Φ with respect to B and C, the corresponding transformation matrix $\tilde{\boldsymbol{A}}_\Phi$ with respect to the bases $\tilde{B}$ and $\tilde{C}$ is given as

$$\tilde{\boldsymbol{A}}_\Phi = \boldsymbol{T}^{-1} \boldsymbol{A}_\Phi \boldsymbol{S} \,. \tag{2.104}$$

Here, $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ is the transformation matrix of id_V that maps coordinates with respect to $\tilde{B}$ onto coordinates with respect to B, and $\boldsymbol{T} \in \mathbb{R}^{m \times m}$ is the transformation matrix of id_W that maps coordinates with respect to $\tilde{C}$ onto coordinates with respect to C.

Proof Following Drumm and Weil (2001) we can write the vectors of the new basis $\tilde{B}$ of V as a linear combination of the basis vectors of B, such that

$$\tilde{\boldsymbol{b}}_j = s_{1j}\boldsymbol{b}_1 + \cdots + s_{nj}\boldsymbol{b}_n = \sum_{i=1}^{n} s_{ij}\boldsymbol{b}_i\,, \quad j = 1,\ldots,n\,. \tag{2.105}$$

Similarly, we write the new basis vectors $\tilde{C}$ of W as a linear combination of the basis vectors of C, which yields

$$\tilde{\boldsymbol{c}}_k = t_{1k}\boldsymbol{c}_1 + \cdots + t_{mk}\boldsymbol{c}_m = \sum_{l=1}^{m} t_{lk}\boldsymbol{c}_l\,, \quad k = 1,\ldots,m\,. \tag{2.106}$$

We define $\boldsymbol{S} = ((s_{ij})) \in \mathbb{R}^{n\times n}$ as the transformation matrix that maps coordinates with respect to $\tilde{B}$ onto coordinates with respect to B and $\boldsymbol{T} = ((t_{lk})) \in \mathbb{R}^{m\times m}$ as the transformation matrix that maps coordinates with respect to $\tilde{C}$ onto coordinates with respect to C. In particular, the jth column of $\boldsymbol{S}$ is the coordinate representation of $\tilde{\boldsymbol{b}}_j$ with respect to B and the kth column of $\boldsymbol{T}$ is the coordinate representation of $\tilde{\boldsymbol{c}}_k$ with respect to C. Note that both $\boldsymbol{S}$ and $\boldsymbol{T}$ are regular.

We are going to look at $\Phi(\tilde{\boldsymbol{b}}_j)$ from two perspectives. First, applying the mapping Φ, we get that for all $j = 1,\ldots,n$

$$\Phi(\tilde{\boldsymbol{b}}_j) = \sum_{k=1}^{m} \underbrace{\tilde{a}_{kj}\tilde{\boldsymbol{c}}_k}_{\in W} \overset{(2.106)}{=} \sum_{k=1}^{m} \tilde{a}_{kj} \sum_{l=1}^{m} t_{lk}\boldsymbol{c}_l = \sum_{l=1}^{m} \left(\sum_{k=1}^{m} t_{lk}\tilde{a}_{kj} \right) \boldsymbol{c}_l\,, \tag{2.107}$$

where we first expressed the new basis vectors $\tilde{\boldsymbol{c}}_k \in W$ as linear combinations of the basis vectors $\boldsymbol{c}_l \in W$ and then swapped the order of summation.

Alternatively, when we express the $\tilde{\boldsymbol{b}}_j \in V$ as linear combinations of $\boldsymbol{b}_j \in V$, we arrive at

$$\Phi(\tilde{\boldsymbol{b}}_j) \overset{(2.105)}{=} \Phi\left(\sum_{i=1}^{n} s_{ij}\boldsymbol{b}_i \right) = \sum_{i=1}^{n} s_{ij}\Phi(\boldsymbol{b}_i) = \sum_{i=1}^{n} s_{ij} \sum_{l=1}^{m} a_{li}\boldsymbol{c}_l \tag{2.108a}$$

$$= \sum_{l=1}^{m} \left(\sum_{i=1}^{n} a_{li}s_{ij} \right) \boldsymbol{c}_l\,, \quad j = 1,\ldots,n\,, \tag{2.108b}$$

where we exploited the linearity of Φ. Comparing (2.107) and (2.108b), it follows for all $j = 1,\ldots,n$ and $l = 1,\ldots,m$ that

$$\sum_{k=1}^{m} t_{lk}\tilde{a}_{kj} = \sum_{i=1}^{n} a_{li}s_{ij} \tag{2.109}$$

and, therefore,

$$\boldsymbol{T}\tilde{\boldsymbol{A}}_\Phi = \boldsymbol{A}_\Phi\boldsymbol{S} \in \mathbb{R}^{m\times n}\,, \tag{2.110}$$

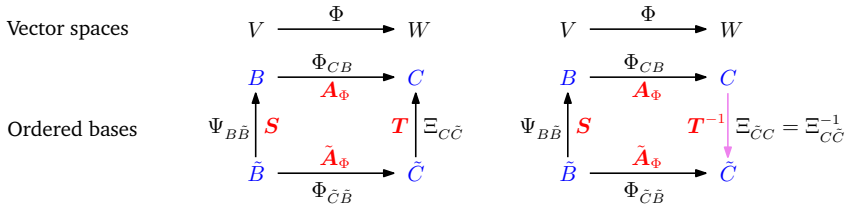


Figure 2.9 For a homomorphism $\Phi : V \to W$ and ordered bases $B, \tilde{B}$ of V and $C, \tilde{C}$ of W (marked in blue), we can express the mapping $\Phi_{\tilde{C}\tilde{B}}$ with respect to the bases $\tilde{B}, \tilde{C}$ equivalently as a composition of the homomorphisms $\Phi_{\tilde{C}\tilde{B}} = \Xi_{\tilde{C}C} \circ \Phi_{CB} \circ \Psi_{B\tilde{B}}$ with respect to the bases in the subscripts. The corresponding transformation matrices are in red.

such that

$$\tilde{\boldsymbol{A}}_\Phi = \boldsymbol{T}^{-1} \boldsymbol{A}_\Phi \boldsymbol{S}\,, \tag{2.111}$$

which proves Theorem 2.19. □

Theorem 2.19 tells us that with a basis change in V (B is replaced with $\tilde{B}$) and W (C is replaced with $\tilde{C}$) the transformation matrix $\boldsymbol{A}_\Phi$ of a linear mapping $\Phi : V \to W$ is replaced by an equivalent matrix $\tilde{\boldsymbol{A}}_\Phi$ with

$$\tilde{\boldsymbol{A}}_\Phi = \boldsymbol{T}^{-1} \boldsymbol{A}_\Phi \boldsymbol{S}. \tag{2.112}$$

Figure 2.9 illustrates this relation: Consider a homomorphism $\Phi : V \to W$ and ordered bases $B, \tilde{B}$ of V and $C, \tilde{C}$ of W. The mapping Φ_{CB} is an instantiation of Φ and maps basis vectors of B onto linear combinations of basis vectors of C. Assuming, we know the transformation matrix $\boldsymbol{A}_\Phi$ of Φ_{CB} with respect to the ordered bases B, C. When we perform a basis change from B to $\tilde{B}$ in V and from C to $\tilde{C}$ in W, we can determine the corresponding transformation matrix $\tilde{\boldsymbol{A}}_\Phi$ as follows: First, we find the matrix representation of the linear mapping $\Psi_{B\tilde{B}} : V \to V$ that maps coordinates with respect to the new basis $\tilde{B}$ onto the (unique) coordinates with respect to the "old" basis B (in V). Then, we use the transformation matrix $\boldsymbol{A}_\Phi$ of $\Phi_{CB} : V \to W$ to map these coordinates onto the coordinates with respect to C in W. Finally, we use a linear mapping $\Xi_{\tilde{C}C} : W \to W$ to map the coordinates with respect to C onto coordinates with respect to $\tilde{C}$. Therefore, we can express the linear mapping $\Phi_{\tilde{C}\tilde{B}}$ as a composition of linear mappings that involve the "old" basis:

$$\Phi_{\tilde{C}\tilde{B}} = \Xi_{\tilde{C}C} \circ \Phi_{CB} \circ \Psi_{B\tilde{B}} = \Xi^{-1}_{C\tilde{C}} \circ \Phi_{CB} \circ \Psi_{B\tilde{B}}\,. \tag{2.113}$$

Concretely, we use $\Psi_{B\tilde{B}} = \mathrm{id}_V$ and $\Xi_{C\tilde{C}} = \mathrm{id}_W$, i.e., the identity mappings that map vectors onto themselves, but with respect to a different basis.

equivalent

Definition 2.20 (Equivalence)**.** Two matrices $\boldsymbol{A}, \tilde{\boldsymbol{A}} \in \mathbb{R}^{m \times n}$ are *equivalent* if there exist regular matrices $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{T} \in \mathbb{R}^{m \times m}$, such that $\tilde{\boldsymbol{A}} = \boldsymbol{T}^{-1} \boldsymbol{A} \boldsymbol{S}$.

similar

Definition 2.21 (Similarity)**.** Two matrices $\boldsymbol{A}, \tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ are *similar* if there exists a regular matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ with $\tilde{\boldsymbol{A}} = \boldsymbol{S}^{-1} \boldsymbol{A} \boldsymbol{S}$

Remark. Similar matrices are always equivalent. However, equivalent matrices are not necessarily similar. ♢

Remark. Consider vector spaces V, W, X. From the remark on page 48 we already know that for linear mappings $\Phi : V \to W$ and $\Psi : W \to X$ the mapping $\Psi \circ \Phi : V \to X$ is also linear. With transformation matrices $\boldsymbol{A}_\Phi$ and $\boldsymbol{A}_\Psi$ of the corresponding mappings, the overall transformation matrix is $\boldsymbol{A}_{\Psi\circ\Phi} = \boldsymbol{A}_\Psi \boldsymbol{A}_\Phi$. ♢

In light of this remark, we can look at basis changes from the perspective of composing linear mappings:

- $\boldsymbol{A}_\Phi$ is the transformation matrix of a linear mapping $\Phi_{CB} : V \to W$ with respect to the bases B, C.
- $\tilde{\boldsymbol{A}}_\Phi$ is the transformation matrix of the linear mapping $\Phi_{\tilde{C}\tilde{B}} : V \to W$ with respect to the bases $\tilde{B}, \tilde{C}$.
- $\boldsymbol{S}$ is the transformation matrix of a linear mapping $\Psi_{B\tilde{B}} : V \to V$ (automorphism) that represents $\tilde{B}$ in terms of B. Normally, $\Psi = \mathrm{id}_V$ is the identity mapping in V.
- $\boldsymbol{T}$ is the transformation matrix of a linear mapping $\Xi_{C\tilde{C}} : W \to W$ (automorphism) that represents $\tilde{C}$ in terms of C. Normally, $\Xi = \mathrm{id}_W$ is the identity mapping in W.

If we (informally) write down the transformations just in terms of bases then $\boldsymbol{A}_\Phi : B \to C$, $\tilde{\boldsymbol{A}}_\Phi : \tilde{B} \to \tilde{C}$, $\boldsymbol{S} : \tilde{B} \to B$, $\boldsymbol{T} : \tilde{C} \to C$ and $\boldsymbol{T}^{-1} : C \to \tilde{C}$, and

$$\tilde{B} \to \tilde{C} = \tilde{B} \to B \to C \to \tilde{C} \tag{2.114}$$

$$\tilde{\boldsymbol{A}}_\Phi = \boldsymbol{T}^{-1} \boldsymbol{A}_\Phi \boldsymbol{S} . \tag{2.115}$$

Note that the execution order in (2.115) is from right to left because vectors are multiplied at the right-hand side so that $\boldsymbol{x} \mapsto \boldsymbol{S}\boldsymbol{x} \mapsto \boldsymbol{A}_\Phi(\boldsymbol{S}\boldsymbol{x}) \mapsto \boldsymbol{T}^{-1}\big(\boldsymbol{A}_\Phi(\boldsymbol{S}\boldsymbol{x})\big) = \tilde{\boldsymbol{A}}_\Phi \boldsymbol{x}$.

Example 2.23 (Basis Change)
Consider a linear mapping $\Phi : \mathbb{R}^3 \to \mathbb{R}^4$ whose transformation matrix is

$$\boldsymbol{A}_\Phi = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 1 & 3 \\ 3 & 7 & 1 \\ -1 & 2 & 4 \end{bmatrix} \tag{2.116}$$

with respect to the standard bases

$$B = \Big(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Big), \quad C = \Big(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \Big). \tag{2.117}$$

We seek the transformation matrix $\tilde{\boldsymbol{A}}_\Phi$ of Φ with respect to the new bases

$$\tilde{B} = (\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}) \in \mathbb{R}^3, \quad \tilde{C} = (\begin{bmatrix}1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\1\end{bmatrix}). \quad (2.118)$$

Then,

$$\boldsymbol{S} = \begin{bmatrix}1 & 0 & 1\\ 1 & 1 & 0\\ 0 & 1 & 1\end{bmatrix}, \qquad \boldsymbol{T} = \begin{bmatrix}1 & 1 & 0 & 1\\ 1 & 0 & 1 & 0\\ 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad (2.119)$$

where the ith column of $\boldsymbol{S}$ is the coordinate representation of $\tilde{\boldsymbol{b}}_i$ in terms of the basis vectors of B. Similarly, the jth column of $\boldsymbol{T}$ is the coordinate representation of $\tilde{\boldsymbol{c}}_j$ in terms of the basis vectors of C.

Since B is the standard basis, the coordinate representation is straightforward to find. For a general basis B we would need to solve a linear equation system to find the λ_i such that $\sum_{i=1}^{3} \lambda_i \boldsymbol{b}_i = \tilde{\boldsymbol{b}}_j$, $j = 1, \ldots, 3$.

Therefore, we obtain

$$\tilde{\boldsymbol{A}}_\Phi = \boldsymbol{T}^{-1}\boldsymbol{A}_\Phi\boldsymbol{S} = \frac{1}{2}\begin{bmatrix}1 & 1 & -1 & -1\\ 1 & -1 & 1 & -1\\ -1 & 1 & 1 & 1\\ 0 & 0 & 0 & 2\end{bmatrix}\begin{bmatrix}3 & 2 & 1\\ 0 & 4 & 2\\ 10 & 8 & 4\\ 1 & 6 & 3\end{bmatrix} \quad (2.120a)$$

$$= \begin{bmatrix}-4 & -4 & -2\\ 6 & 0 & 0\\ 4 & 8 & 4\\ 1 & 6 & 3\end{bmatrix}. \quad (2.120b)$$

In Chapter 4, we will be able to exploit the concept of a basis change to find a basis with respect to which the transformation matrix of an endomorphism has a particularly simple (diagonal) form. In Chapter 10, we will look at a data compression problem and find a convenient basis onto which we can project the data while minimizing the compression loss.

2.7.3 Image and Kernel

The image and kernel of a linear mapping are vector subspaces with certain important properties. In the following, we will characterize them more carefully.

Definition 2.22 (Image and Kernel)**.**
For $\Phi : V \to W$, we define the *kernel/null space*

kernel
null space

$$\ker(\Phi) := \Phi^{-1}(\boldsymbol{0}_W) = \{\boldsymbol{v} \in V : \Phi(\boldsymbol{v}) = \boldsymbol{0}_W\} \quad (2.121)$$

and the *image/range*

image
range

$$\mathrm{Im}(\Phi) := \Phi(V) = \{\boldsymbol{w} \in W | \exists \boldsymbol{v} \in V : \Phi(\boldsymbol{v}) = \boldsymbol{w}\}. \quad (2.122)$$

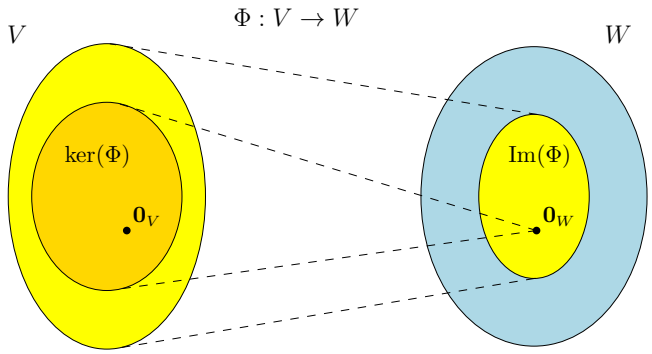


Figure 2.10 Kernel and Image of a linear mapping $\Phi : V \to W$.

We also call V and W also the *domain* and *codomain* of Φ, respectively.

domain
codomain

Intuitively, the kernel is the set of vectors in $\boldsymbol{v} \in V$ that Φ maps onto the neutral element $\mathbf{0}_W \in W$. The image is the set of vectors $\boldsymbol{w} \in W$ that can be "reached" by Φ from any vector in V. An illustration is given in Figure 2.10.

Remark. Consider a linear mapping $\Phi : V \to W$, where V, W are vector spaces.

- It always holds that $\Phi(\mathbf{0}_V) = \mathbf{0}_W$ and, therefore, $\mathbf{0}_V \in \ker(\Phi)$. In particular, the null space is never empty.
- $\mathrm{Im}(\Phi) \subseteq W$ is a subspace of W, and $\ker(\Phi) \subseteq V$ is a subspace of V.
- Φ is injective (one-to-one) if and only if $\ker(\Phi) = \{\mathbf{0}\}$

♢

Remark (Null Space and Column Space). Let us consider $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a linear mapping $\Phi : \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$.

- For $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$, where $\boldsymbol{a}_i$ are the columns of $\boldsymbol{A}$, we obtain

$$\mathrm{Im}(\Phi) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^n\} = \left\{\sum_{i=1}^{n} x_i \boldsymbol{a}_i : x_1, \ldots, x_n \in \mathbb{R}\right\} \tag{2.123a}$$

$$= \mathrm{span}[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n] \subseteq \mathbb{R}^m\,, \tag{2.123b}$$

  i.e., the image is the span of the columns of $\boldsymbol{A}$, also called the *column space*. Therefore, the column space (image) is a subspace of $\mathbb{R}^m$, where m is the "height" of the matrix.

column space

- $\mathrm{rk}(\boldsymbol{A}) = \dim(\mathrm{Im}(\Phi))$
- The kernel/null space $\ker(\Phi)$ is the general solution to the linear homogeneous equation system $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$ and captures all possible linear combinations of the elements in $\mathbb{R}^n$ that produce $\mathbf{0} \in \mathbb{R}^m$.
- The kernel is a subspace of $\mathbb{R}^n$, where n is the "width" of the matrix.
- The kernel focuses on the relationship among the columns, and we can use it to determine whether/how we can express a column as a linear combination of other columns.

- The purpose of the kernel is to determine whether a solution of the system of linear equations is unique and, if not, to capture all possible solutions.

♢

Example 2.24 (Image and Kernel of a Linear Mapping)
The mapping

$$\Phi : \mathbb{R}^4 \to \mathbb{R}^2, \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \mapsto \begin{bmatrix} 1 & 2 & -1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 + 2x_2 - x_3 \\ x_1 + x_4 \end{bmatrix} \tag{2.124}$$

$$= x_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} 2 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.125}$$

is linear. To determine $\mathrm{Im}(\Phi)$ we can take the span of the columns of the transformation matrix and obtain

$$\mathrm{Im}(\Phi) = \mathrm{span}[\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}]. \tag{2.126}$$

To compute the kernel (null space) of Φ, we need to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$, i.e., we need to solve a homogeneous equation system. To do this, we use Gaussian elimination to transform $\boldsymbol{A}$ into reduced row echelon form:

$$\begin{bmatrix} 1 & 2 & -1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \cdots \rightsquigarrow \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}. \tag{2.127}$$

This matrix is in reduced row echelon form, and we can use the Minus-1 Trick to compute a basis of the kernel (see Section 2.3.3). Alternatively, we can express the non-pivot columns (columns 3 and 4) as linear combinations of the pivot-columns (columns 1 and 2). The third column $\boldsymbol{a}_3$ is equivalent to $-\frac{1}{2}$ times the second column $\boldsymbol{a}_2$. Therefore, $\boldsymbol{0} = \boldsymbol{a}_3 + \frac{1}{2}\boldsymbol{a}_2$. In the same way, we see that $\boldsymbol{a}_4 = \boldsymbol{a}_1 - \frac{1}{2}\boldsymbol{a}_2$ and, therefore, $\boldsymbol{0} = \boldsymbol{a}_1 - \frac{1}{2}\boldsymbol{a}_2 - \boldsymbol{a}_4$. Overall, this gives us the kernel (null space) as

$$\ker(\Phi) = \mathrm{span}[\begin{bmatrix} 0 \\ \frac{1}{2} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ \frac{1}{2} \\ 0 \\ 1 \end{bmatrix}]. \tag{2.128}$$

Theorem 2.23 (Rank-Nullity Theorem)**.** *For vector spaces V, W and a linear mapping $\Phi : V \to W$ it holds that*

$$\dim(\ker(\Phi)) + \dim(\mathrm{Im}(\Phi)) = \dim(V). \tag{2.129}$$

2.8 Affine Spaces

In the following, we will have a closer look at spaces that are offset from the origin, i.e., spaces that are no longer vector subspaces. Moreover, we will briefly discuss properties of mappings between these affine spaces, which resemble linear mappings.

2.8.1 Affine Subspaces

Definition 2.24 (Affine Subspace)**.** Let V be a vector space, $\boldsymbol{x}_0 \in V$ and $U \subseteq V$ a subspace. Then the subset

$$L = \boldsymbol{x}_0 + U := \{\boldsymbol{x}_0 + \boldsymbol{u} : \boldsymbol{u} \in U\} \tag{2.130a}$$
$$= \{\boldsymbol{v} \in V | \exists \boldsymbol{u} \in U : \boldsymbol{v} = \boldsymbol{x}_0 + \boldsymbol{u}\} \subseteq V \tag{2.130b}$$

is called *affine subspace* or *linear manifold* of V. U is called *direction* or *direction space*, and $\boldsymbol{x}_0$ is called *support point*. In Chapter 12, we refer to such a subspace as a *hyperplane*.

affine subspace
linear manifold
direction
direction space
support point
hyperplane

Note that the definition of an affine subspace excludes $\mathbf{0}$ if $\boldsymbol{x}_0 \notin U$. Therefore, an affine subspace is not a (linear) subspace (vector subspace) of V for $\boldsymbol{x}_0 \notin U$.

Examples of affine subspaces are points, lines and planes in $\mathbb{R}^3$, which do not (necessarily) go through the origin.

Remark. Consider two affine subspaces $L = \boldsymbol{x}_0 + U$ and $\tilde{L} = \tilde{\boldsymbol{x}}_0 + \tilde{U}$ of a vector space V. Then, $L \subseteq \tilde{L}$ if and only if $U \subseteq \tilde{U}$ and $\boldsymbol{x}_0 - \tilde{\boldsymbol{x}}_0 \in \tilde{U}$.

Affine subspaces are often described by *parameters*: Consider a k-dimensional affine space $L = \boldsymbol{x}_0 + U$ of V. If $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k)$ is an ordered basis of U, then every element $\boldsymbol{x} \in L$ can be uniquely described as

parameters

$$\boldsymbol{x} = \boldsymbol{x}_0 + \lambda_1 \boldsymbol{b}_1 + \ldots + \lambda_k \boldsymbol{b}_k \,, \tag{2.131}$$

where $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$. This representation is called *parametric equation* of L with directional vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ and *parameters* $\lambda_1, \ldots, \lambda_k$. ♢

parametric equation
parameters

Example 2.25 (Affine Subspaces)

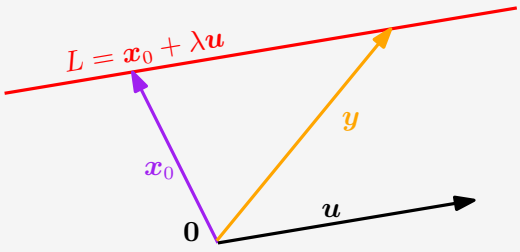


Figure 2.11 Vectors $\boldsymbol{y}$ on a line lie in an affine subspace L with support point $\boldsymbol{x}_0$ and direction $\boldsymbol{u}$.

lines
- One-dimensional affine subspaces are called *lines* and can be written as $\boldsymbol{y} = \boldsymbol{x}_0 + \lambda \boldsymbol{x}_1$, where $\lambda \in \mathbb{R}$, where $U = \text{span}[\boldsymbol{x}_1] \subseteq \mathbb{R}^n$ is a one-dimensional subspace of $\mathbb{R}^n$. This means, a line is defined by a support point $\boldsymbol{x}_0$ and a vector $\boldsymbol{x}_1$ that defines the direction. See Figure 2.11 for an illustration.

planes
- Two-dimensional affine subspaces of $\mathbb{R}^n$ are called *planes*. The parametric equation for planes is $\boldsymbol{y} = \boldsymbol{x}_0 + \lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2$, where $\lambda_1, \lambda_2 \in \mathbb{R}$ and $U = [\boldsymbol{x}_1, \boldsymbol{x}_2] \subseteq \mathbb{R}^n$. This means, a plane is defined by a support point $\boldsymbol{x}_0$ and two linearly independent vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$ that span the direction space.

hyperplanes
- In $\mathbb{R}^n$, the $(n-1)$-dimensional affine subspaces are called *hyperplanes*, and the corresponding parametric equation is $\boldsymbol{y} = \boldsymbol{x}_0 + \sum_{i=1}^{n-1} \lambda_i \boldsymbol{x}_i$, where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$ form a basis of an $(n-1)$-dimensional subspace U of $\mathbb{R}^n$. This means, a hyperplane is defined by a support point $\boldsymbol{x}_0$ and $(n-1)$ linearly independent vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$ that span the direction space. In $\mathbb{R}^2$, a line is also a hyperplane. In $\mathbb{R}^3$, a plane is also a hyperplane.

Remark (Inhomogeneous linear equation systems and affine subspaces). For $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ the solution of the linear equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is either the empty set or an affine subspace of $\mathbb{R}^n$ of dimension $n - \text{rk}(\boldsymbol{A})$. In particular, the solution of the linear equation $\lambda_1 \boldsymbol{x}_1 + \ldots + \lambda_n \boldsymbol{x}_n = \boldsymbol{b}$, where $(\lambda_1, \ldots, \lambda_n) \neq (0, \ldots, 0)$, is a hyperplane in $\mathbb{R}^n$.

In $\mathbb{R}^n$, every k-dimensional affine subspace is the solution of a linear inhomogeneous equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$ and $\text{rk}(\boldsymbol{A}) = n - k$. Recall that for homogeneous equation systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ the solution was a vector subspace, which we can also think of as a special affine space with support point $\boldsymbol{x}_0 = \boldsymbol{0}$. ♢

2.8.2 Affine Mappings

Similar to linear mappings between vector spaces, which we discussed in Section 2.7, we can define affine mappings between two affine spaces. Linear and affine mappings are closely related. Therefore, many properties that we already know from linear mappings, e.g., that the composition of linear mappings is a linear mapping, also hold for affine mappings.

Definition 2.25 (Affine mapping)**.** For two vector spaces V, W and a linear mapping $\Phi : V \to W$ and $\boldsymbol{a} \in W$ the mapping

$$\phi : V \to W \tag{2.132}$$
$$\boldsymbol{x} \mapsto \boldsymbol{a} + \Phi(\boldsymbol{x}) \tag{2.133}$$

affine mapping 1449 is an *affine mapping* from V to W. The vector $\boldsymbol{a}$ is called the *translation* translation vector 1450 *vector* of ϕ.

- Every affine mapping $\phi : V \to W$ is also the composition of a linear mapping $\Phi : V \to W$ and a translation $\tau : W \to W$ in W, such that $\phi = \tau \circ \Phi$. The mappings Φ and τ are uniquely determined.
- The composition $\phi' \circ \phi$ of affine mappings $\phi : V \to W$, $\phi' : W \to X$ is affine.
- Affine mappings keep the geometric structure invariant. They also preserve the dimension and parallelism.

Exercises

2.1 We consider $(\mathbb{R}\backslash\{-1\}, \star)$ where

$$a \star b := ab + a + b, \qquad a, b \in \mathbb{R}\backslash\{-1\} \tag{2.134}$$

1. Show that $(\mathbb{R}\backslash\{-1\}, \star)$ is an Abelian group
2. Solve

$$3 \star x \star x = 15$$

in the Abelian group $(\mathbb{R}\backslash\{-1\}, \star)$, where $\star$ is defined in (2.134).

2.2 Let n be in $\mathbb{N} \setminus \{0\}$. Let k, x be in $\mathbb{Z}$. We define the congruence class $\bar{k}$ of the integer k as the set

$$\begin{aligned}\bar{k} &= \{x \in \mathbb{Z} \mid x - k = 0 \pmod{n}\} \\ &= \{x \in \mathbb{Z} \mid (\exists a \in \mathbb{Z}) \colon (x - k = n \cdot a)\}\,.\end{aligned}$$

We now define $\mathbb{Z}/n\mathbb{Z}$ (sometimes written $\mathbb{Z}_n$) as the set of all congruence classes modulo n. Euclidean division implies that this set is a finite set containing n elements:

$$\mathbb{Z}_n = \{\bar{0}, \bar{1}, \ldots, \overline{n-1}\}$$

For all $\bar{a}, \bar{b} \in \mathbb{Z}_n$, we define

$$\bar{a} \oplus \bar{b} := \overline{a + b}$$

1. Show that $(\mathbb{Z}_n, \oplus)$ is a group. Is it Abelian?
2. We now define another operation $\otimes$ for all $\bar{a}$ and $\bar{b}$ in $\mathbb{Z}_n$ as

$$\bar{a} \otimes \bar{b} = \overline{a \times b} \tag{2.135}$$

where $a \times b$ represents the usual multiplication in $\mathbb{Z}$.
Let $n = 5$. Draw the times table of the elements of $\mathbb{Z}_5 \setminus \{\bar{0}\}$ under $\otimes$, i.e., calculate the products $\bar{a} \otimes \bar{b}$ for all $\bar{a}$ and $\bar{b}$ in $\mathbb{Z}_5 \setminus \{\bar{0}\}$.
Hence, show that $\mathbb{Z}_5 \setminus \{\bar{0}\}$ is closed under $\otimes$ and possesses a neutral element for $\otimes$. Display the inverse of all elements in $\mathbb{Z}_5 \setminus \{\bar{0}\}$ under $\otimes$.
Conclude that $(\mathbb{Z}_5 \setminus \{\bar{0}\}, \otimes)$ is an Abelian group.
3. Show that $(\mathbb{Z}_8 \setminus \{\bar{0}\}, \otimes)$ is not a group.

4. We recall that Bézout theorem states that two integers a and b are relatively prime (i.e., $gcd(a, b) = 1$) if and only if there exist two integers u and v such that $au + bv = 1$. Show that $(\mathbb{Z}_n \setminus \{\overline{0}\}, \otimes)$ is a group if and only if $n \in \mathbb{N} \setminus \{0\}$ is prime.

2.3 Consider the set $\mathcal{G}$ of 3×3 matrices defined as:

$$\mathcal{G} = \left\{ \begin{bmatrix} 1 & x & z \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \,\middle|\, x, y, z \in \mathbb{R} \right\} \tag{2.136}$$

We define $\cdot$ as the standard matrix multiplication.
Is $(\mathcal{G}, \cdot)$ a group? If yes, is it Abelian? Justify your answer.

2.4 Compute the following matrix products:

1.
$$\begin{bmatrix} 1 & 2 \\ 4 & 5 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

2.
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

3.
$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

4.
$$\begin{bmatrix} 1 & 2 & 1 & 2 \\ 4 & 1 & -1 & -4 \end{bmatrix} \begin{bmatrix} 0 & 3 \\ 1 & -1 \\ 2 & 1 \\ 5 & 2 \end{bmatrix}$$

5.
$$\begin{bmatrix} 0 & 3 \\ 1 & -1 \\ 2 & 1 \\ 5 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 2 \\ 4 & 1 & -1 & -4 \end{bmatrix}$$

2.5 Find the set $\mathcal{S}$ of all solutions in $\boldsymbol{x}$ of the following inhomogeneous linear systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A}$ and $\boldsymbol{b}$ are defined below:

1.
$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$$

2.

$$\boldsymbol{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$$

3. Using Gaussian elimination find all solutions of the inhomogeneous equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

2.6 Find all solutions in $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^3$ of the equation system $\boldsymbol{A}\boldsymbol{x} = 12\boldsymbol{x}$, where

$$\boldsymbol{A} = \begin{bmatrix} 6 & 4 & 3 \\ 6 & 0 & 9 \\ 0 & 8 & 0 \end{bmatrix}$$

and $\sum_{i=1}^{3} x_i = 1$.

2.7 Determine the inverse of the following matrices if possible:

1.

$$\boldsymbol{A} = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

2.

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

2.8 Which of the following sets are subspaces of $\mathbb{R}^3$?

1. $A = \{(\lambda, \lambda + \mu^3, \lambda - \mu^3) \mid \lambda, \mu \in \mathbb{R}\}$
2. $B = \{(\lambda^2, -\lambda^2, 0) \mid \lambda \in \mathbb{R}\}$
3. Let γ be in $\mathbb{R}$.
   $C = \{(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3 \mid \xi_1 - 2\xi_2 + 3\xi_3 = \gamma\}$
4. $D = \{(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3 \mid \xi_2 \in \mathbb{Z}\}$

2.9 Are the following vectors linearly independent?

1.

$$\boldsymbol{x}_1 = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}, \quad \boldsymbol{x}_2 \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}, \quad \boldsymbol{x}_3 \begin{bmatrix} 3 \\ -3 \\ 8 \end{bmatrix}$$

2.

$$\boldsymbol{x}_1 = \begin{bmatrix}1\\2\\1\\0\\0\end{bmatrix}, \quad \boldsymbol{x}_2 = \begin{bmatrix}1\\1\\0\\1\\1\end{bmatrix}, \quad \boldsymbol{x}_3 = \begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix}$$

2.10 Write

$$\boldsymbol{y} = \begin{bmatrix}1\\-2\\5\end{bmatrix}$$

as linear combination of

$$\boldsymbol{x}_1 = \begin{bmatrix}1\\1\\1\end{bmatrix}, \quad \boldsymbol{x}_2 = \begin{bmatrix}1\\2\\3\end{bmatrix}, \quad \boldsymbol{x}_3 = \begin{bmatrix}2\\-1\\1\end{bmatrix}$$

2.11 1. Consider two subspaces of $\mathbb{R}^4$:

$$U_1 = \text{span}[\begin{bmatrix}1\\1\\-3\\1\end{bmatrix}, \begin{bmatrix}2\\-1\\0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\\-1\\1\end{bmatrix}], \quad U_2 = \text{span}[\begin{bmatrix}-1\\-2\\2\\1\end{bmatrix}, \begin{bmatrix}2\\-2\\0\\0\end{bmatrix}, \begin{bmatrix}-3\\6\\-2\\-1\end{bmatrix}].$$

1487 Determine a basis of $U_1 \cap U_2$.

2. Consider two subspaces U_1 and U_2, where U_1 is the solution space of the homogeneous equation system $\boldsymbol{A}_1\boldsymbol{x} = \boldsymbol{0}$ and U_2 is the solution space of the homogeneous equation system $\boldsymbol{A}_2\boldsymbol{x} = \boldsymbol{0}$ with

$$\boldsymbol{A}_1 = \begin{bmatrix}1 & 0 & 1\\1 & -2 & -1\\2 & 1 & 3\\1 & 0 & 1\end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix}3 & -3 & 0\\1 & 2 & 3\\7 & -5 & 2\\3 & -1 & 2\end{bmatrix}.$$

1488 1. Determine the dimension of U_1, U_2
1489 2. Determine bases of U_1 and U_2
1490 3. Determine a basis of $U_1 \cap U_2$

2.12 Consider two subspaces U_1 and U_2, where U_1 is spanned by the columns of $\boldsymbol{A}_1$ and U_2 is spanned by the columns of $\boldsymbol{A}_2$ with

$$\boldsymbol{A}_1 = \begin{bmatrix}1 & 0 & 1\\1 & -2 & -1\\2 & 1 & 3\\1 & 0 & 1\end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix}3 & -3 & 0\\1 & 2 & 3\\7 & -5 & 2\\3 & -1 & 2\end{bmatrix}.$$

1491 1. Determine the dimension of U_1, U_2
1492 2. Determine bases of U_1 and U_2
1493 3. Determine a basis of $U_1 \cap U_2$

1494 2.13 Let $F = \{(x, y, z) \in \mathbb{R}^3 \mid x+y-z = 0\}$ and $G = \{(a-b, a+b, a-3b) \mid a, b \in \mathbb{R}\}$.

1495 1. Show that F and G are subspaces of $\mathbb{R}^3$.
1496 2. Calculate $F \cap G$ without resorting to any basis vector.

3. Find one basis for F and one for G, calculate $F \cap G$ using the basis vectors previously found and check your result with the previous question.

2.14 Are the following mappings linear?

1. Let $a, b \in \mathbb{R}$.

$$\begin{aligned}\Phi : L^1([a,b]) &\to \mathbb{R} \\ f &\mapsto \Phi(f) = \int_a^b f(x)dx\,,\end{aligned}$$

where $L^1([a,b])$ denotes the set of integrable function on $[a,b]$.

2.

$$\begin{aligned}\Phi : C^1 &\to C^0 \\ f &\mapsto \Phi(f) = f'\,.\end{aligned}$$

where for $k \geqslant 1$, C^k denotes the set of k times continuously differentiable functions, and C^0 denotes the set of continuous functions.

3.

$$\begin{aligned}\Phi : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \Phi(x) = \cos(x)\end{aligned}$$

4.

$$\begin{aligned}\Phi : \mathbb{R}^3 &\to \mathbb{R}^2 \\ \boldsymbol{x} &\mapsto \begin{bmatrix}1 & 2 & 3\\ 1 & 4 & 3\end{bmatrix}\boldsymbol{x}\end{aligned}$$

5. Let θ be in $[0, 2\pi[$.

$$\begin{aligned}\Phi : \mathbb{R}^2 &\to \mathbb{R}^2 \\ \boldsymbol{x} &\mapsto \begin{bmatrix}\cos(\theta) & \sin(\theta)\\ -\sin(\theta) & \cos(\theta)\end{bmatrix}\boldsymbol{x}\end{aligned}$$

2.15 Consider the linear mapping

$$\begin{aligned}\Phi : \mathbb{R}^3 &\to \mathbb{R}^4 \\ \Phi\left(\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}\right) &= \begin{bmatrix}3x_1 + 2x_2 + x_3\\ x_1 + x_2 + x_3\\ x_1 - 3x_2\\ 2x_1 + 3x_2 + x_3\end{bmatrix}\end{aligned}$$

- Find the transformation matrix $\boldsymbol{A}_\Phi$
- Determine $\mathrm{rk}(\boldsymbol{A}_\Phi)$
- Compute kernel and image of Φ. What is $\dim(\ker(\Phi))$ and $\dim(\mathrm{Im}(\Phi))$?

2.16 Let E be a vector space. Let f and g be two endomorphisms on E such that $f \circ g = \mathrm{id}_E$ (i.e. $f \circ g$ is the identity isomorphism). Show that $\ker f = \ker(g \circ f)$, $\mathrm{Im}g = \mathrm{Im}(g \circ f)$ and that $\ker(f) \cap \mathrm{Im}(g) = \{\mathbf{0}_E\}$.

2.17 Consider an endomorphism $\Phi : \mathbb{R}^3 \to \mathbb{R}^3$ whose transformation matrix (with respect to the standard basis in $\mathbb{R}^3$) is

$$\boldsymbol{A}_\Phi = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

1509 1. Determine $\ker(\Phi)$ and $\mathrm{Im}(\Phi)$.
2. Determine the transformation matrix $\tilde{\boldsymbol{A}}_\Phi$ with respect to the basis

$$B = (\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}),$$

1510 i.e., perform a basis change toward the new basis B.

2.18 Let us consider four vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}'_1, \boldsymbol{b}'_2$ of $\mathbb{R}^2$ expressed in the standard basis of $\mathbb{R}^2$ as

$$\boldsymbol{b}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \boldsymbol{b}'_1 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad \boldsymbol{b}'_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2.137}$$

1511 and let us define $B = (\boldsymbol{b}_1, \boldsymbol{b}_2)$ and $B' = (\boldsymbol{b}'_1, \boldsymbol{b}'_2)$.

1512 1. Show that B and B' are two bases of $\mathbb{R}^2$ and draw those basis vectors.
1513 2. Compute the matrix $\boldsymbol{P}_1$ which performs a basis change from B' to B.

1514 2.19 We consider three vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ of $\mathbb{R}^3$ defined in the standard basis of $\mathbb{R}$
1515 as

$$\boldsymbol{c}_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \quad \boldsymbol{c}_2 = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}, \quad \boldsymbol{c}_3 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{2.138}$$

1516 and we define $C = (\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$.

1517 1. Show that C is a basis of $\mathbb{R}^3$.
1518 2. Let us call $C' = (\boldsymbol{c}'_1, \boldsymbol{c}'_2, \boldsymbol{c}'_3)$ the standard basis of $\mathbb{R}^3$. Explicit the matrix
1519 $\boldsymbol{P}_2$ that performs the basis change from C to C'.

2.20 Let us consider $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}'_1, \boldsymbol{b}'_2$, 4 vectors of $\mathbb{R}^2$ expressed in the standard basis of $\mathbb{R}^2$ as

$$\boldsymbol{b}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \boldsymbol{b}'_1 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad \boldsymbol{b}'_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2.139}$$

1520 and let us define two ordered bases $B = (\boldsymbol{b}_1, \boldsymbol{b}_2)$ and $B' = (\boldsymbol{b}'_1, \boldsymbol{b}'_2)$ of $\mathbb{R}^2$.

1521 1. Show that B and B' are two bases of $\mathbb{R}^2$ and draw those basis vectors.
1522 2. Compute the matrix $\boldsymbol{P}_1$ that performs a basis change from B' to B.
3. We consider $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$, 3 vectors of $\mathbb{R}^3$ defined in the standard basis of $\mathbb{R}$ as

$$\boldsymbol{c}_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \quad \boldsymbol{c}_2 = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}, \quad \boldsymbol{c}_3 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{2.140}$$

1523 and we define $C = (\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$.

1. Show that C is a basis of $\mathbb{R}^3$ using determinants
2. Let us call $C' = (\boldsymbol{c}'_1, \boldsymbol{c}'_2, \boldsymbol{c}'_3)$ the standard basis of $\mathbb{R}^3$. Determine the matrix $\boldsymbol{P}_2$ that performs the basis change from C to C'.

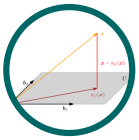
4. We consider a homomorphism $\Phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$, such that

$$\begin{aligned} \Phi(\boldsymbol{b}_1 + \boldsymbol{b}_2) &= \boldsymbol{c}_2 + \boldsymbol{c}_3 \\ \Phi(\boldsymbol{b}_1 - \boldsymbol{b}_2) &= 2\boldsymbol{c}_1 - \boldsymbol{c}_2 + 3\boldsymbol{c}_3 \end{aligned} \tag{2.141}$$

   where $B = (\boldsymbol{b}_1, \boldsymbol{b}_2)$ and $C = (\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$ are ordered bases of $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively.
   Determine the transformation matrix $\boldsymbol{A}_\Phi$ of Φ with respect to the ordered bases B and C.
5. Determine $\boldsymbol{A}'$, the transformation matrix of Φ with respect to the bases $\boldsymbol{B}'$ and $\boldsymbol{C}'$.
6. Let us consider the vector $\boldsymbol{x} \in \mathbb{R}^2$ whose coordinates in B' are $[2, 3]^\top$. In other words, $\boldsymbol{x} = 2\boldsymbol{b}'_1 + 3\boldsymbol{b}'_3$.
   1. Calculate the coordinates of $\boldsymbol{x}$ in B.
   2. Based on that, compute the coordinates of $\Phi(\boldsymbol{x})$ expressed in C.
   3. Then, write $\Phi(\boldsymbol{x})$ in terms of $\boldsymbol{c}'_1, \boldsymbol{c}'_2, \boldsymbol{c}'_3$.
   4. Use the representation of $\boldsymbol{x}$ in B' and the matrix $\boldsymbol{A}'$ to find this result directly.

3

Analytic Geometry



In Chapter 2, we studied vectors, vector spaces and linear mappings at a general but abstract level. In this chapter, we will add some geometric interpretation and intuition to all of these concepts. In particular, we will look at geometric vectors, compute their lengths and distances or angles between two vectors. To be able to do this, we equip the vector space with an inner product that induces the geometry of the vector space. Inner products and their corresponding norms and metrics capture the intuitive notions of similarity and distances, which we use to develop the Support Vector Machine in Chapter 12. We will then use the concepts of lengths and angles between vectors to discuss orthogonal projections, which will play a central role when we discuss principal component analysis in Chapter 10 and regression via maximum likelihood estimation in Chapter 9. Figure 3.1 gives an overview of how concepts in this chapter are related and how they are connected to other chapters of the book.

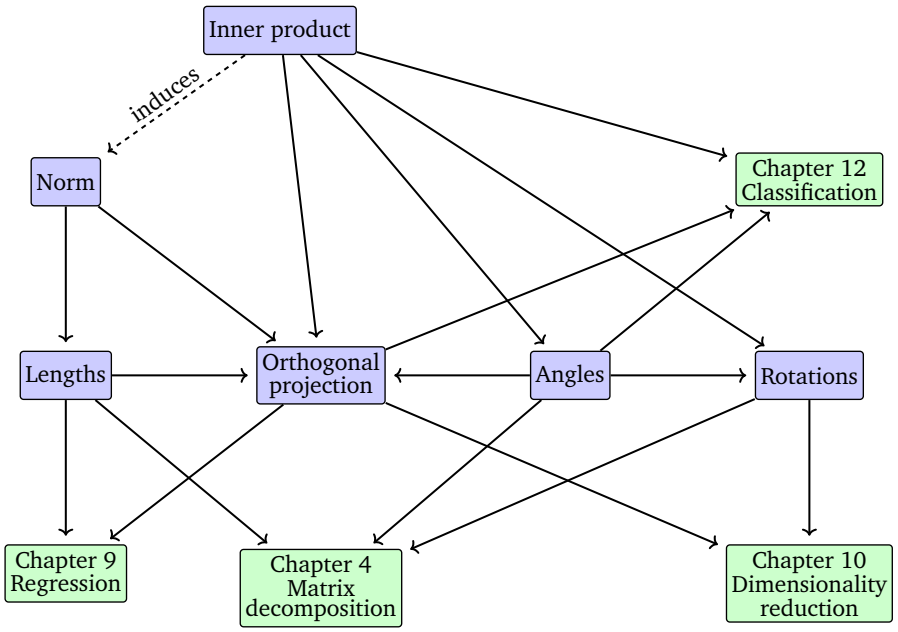


Figure 3.1 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

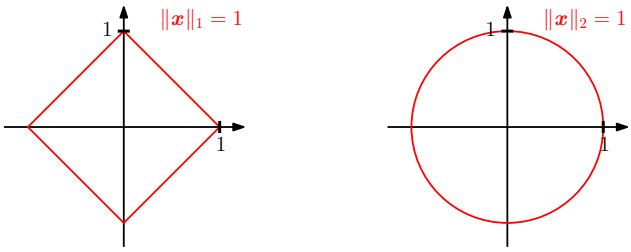


Figure 3.3 For different norms, the red lines indicate the set of vectors with norm 1. Left: Manhattan norm; Right: Euclidean distance.

3.1 Norms

When we think of geometric vectors, i.e., directed line segments that start at the origin, then intuitively the length of a vector is the distance of the "end" of this directed line segment from the origin. In the following, we will discuss the notion of the length of vectors using the concept of a norm.

Definition 3.1 (Norm)**.** A *norm* on a vector space V is a function norm

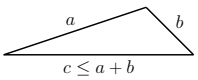
$$\|\cdot\| : V \to \mathbb{R}\,, \tag{3.1}$$
$$\boldsymbol{x} \mapsto \|\boldsymbol{x}\|\,, \tag{3.2}$$

which assigns each vector $\boldsymbol{x}$ its *length* $\|\boldsymbol{x}\| \in \mathbb{R}$, such that for all $\lambda \in \mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{y} \in V$ the following hold: length

- Absolutely homogeneous: $\|\lambda\boldsymbol{x}\| = |\lambda|\|\boldsymbol{x}\|$
- *Triangle inequality*: $\|\boldsymbol{x} + \boldsymbol{y}\| \leqslant \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ Triangle inequality
- *Positive definite*: $\|\boldsymbol{x}\| \geqslant 0$ and $\|\boldsymbol{x}\| = 0 \iff \boldsymbol{x} = \mathbf{0}$. Positive definite

In geometric terms, the triangle inequality states that for any triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side; see Figure 3.2 for an illustration.

Figure 3.2 Triangle inequality.



Recall that for a vector $\boldsymbol{x} \in \mathbb{R}^n$ we denote the elements of the vector using a subscript, that is x_i is the i^{th} element of the vector $\boldsymbol{x}$.

Example 3.1 (Manhattan Norm)
The *Manhattan norm* on $\mathbb{R}^n$ is defined for $\boldsymbol{x} \in \mathbb{R}^n$ as Manhattan norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|\,, \tag{3.3}$$

where $|\cdot|$ is the absolute value. The left panel of Figure 3.3 indicates all vectors $\boldsymbol{x} \in \mathbb{R}^2$ with $\|\boldsymbol{x}\|_1 = 1$. The Manhattan norm is also called ℓ_1 *norm*. ℓ_1 norm

Example 3.2 (Euclidean Norm)
The length of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is given by

$$\|\boldsymbol{x}\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\boldsymbol{x}^\top \boldsymbol{x}}\,, \tag{3.4}$$

Euclidean distance
Euclidean norm
ℓ_2 norm

which computes the *Euclidean distance* of $\boldsymbol{x}$ from the origin. This norm is called the *Euclidean norm*. The right panel of Figure 3.3 shows all vectors $\boldsymbol{x} \in \mathbb{R}^2$ with $\|\boldsymbol{x}\|_2 = 1$. The Euclidean norm is also called ℓ_2 *norm*.

Remark. Throughout this book, we will use the Euclidean norm (3.4) by default if not stated otherwise. ♢

Remark (Inner Products and Norms). Every inner product induces a norm, but there are norms (like the ℓ_1 norm) without a corresponding inner product. For an inner product vector space $(V, \langle \cdot, \cdot \rangle)$ the induced norm $\| \cdot \|$ satisfies the *Cauchy-Schwarz inequality*

Cauchy-Schwarz inequality

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leqslant \|\boldsymbol{x}\| \|\boldsymbol{y}\|\,. \tag{3.5}$$

♢

3.2 Inner Products

Inner products allow for the introduction of intuitive geometrical concepts, such as the length of a vector and the angle or distance between two vectors. A major purpose of inner products is to determine whether vectors are orthogonal to each other.

3.2.1 Dot Product

We may already be familiar with a particular type of inner product, the *scalar product/dot product* in $\mathbb{R}^n$, which is given by

scalar product
dot product

$$\boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i\,. \tag{3.6}$$

We will refer to the particular inner product above as the dot product in this book. However, inner products are more general concepts with specific properties, which we will now introduce.

3.2.2 General Inner Products

Recall the linear mapping from Section 2.7, where we can rearrange the mapping with respect to addition and multiplication with a scalar. A *bilinear*

bilinear mapping

mapping Ω is a mapping with two arguments, and it is linear in each argument, i.e., when we look at a vector space V then it holds that for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V,\ \lambda \in \mathbb{R}$

$$\Omega(\lambda \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z}) = \lambda \Omega(\boldsymbol{x}, \boldsymbol{z}) + \Omega(\boldsymbol{y}, \boldsymbol{z}) \tag{3.7}$$

$$\Omega(\boldsymbol{x}, \lambda \boldsymbol{y} + \boldsymbol{z}) = \lambda \Omega(\boldsymbol{x}, \boldsymbol{y}) + \Omega(\boldsymbol{x}, \boldsymbol{z})\,. \tag{3.8}$$

Here, (3.7) asserts that Ω is linear in the first argument, and (3.8) asserts that Ω is linear in the second argument.

Definition 3.2. Let V be a vector space and $\Omega : V \times V \to \mathbb{R}$ be a bilinear mapping that takes two vectors and maps them onto a real number. Then

- Ω is called *symmetric* if $\Omega(\boldsymbol{x}, \boldsymbol{y}) = \Omega(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in V$, i.e., the order of the arguments does not matter. symmetric
- Ω is called *positive definite* if positive definite

$$\forall \boldsymbol{x} \in V \setminus \{\boldsymbol{0}\} : \Omega(\boldsymbol{x}, \boldsymbol{x}) > 0\,, \quad \Omega(\boldsymbol{0}, \boldsymbol{0}) = 0 \tag{3.9}$$

Definition 3.3. Let V be a vector space and $\Omega : V \times V \to \mathbb{R}$ be a bilinear mapping that takes two vectors and maps them onto a real number. Then

- A positive definite, symmetric bilinear mapping $\Omega : V \times V \to \mathbb{R}$ is called an *inner product* on V. We typically write $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ instead of $\Omega(\boldsymbol{x}, \boldsymbol{y})$. inner product
- The pair $(V, \langle \cdot, \cdot \rangle)$ is called an *inner product space* or (real) *vector space with inner product*. If we use the dot product defined in (3.6), we call $(V, \langle \cdot, \cdot \rangle)$ a *Euclidean vector space*. inner product space; vector space with inner product; Euclidean vector space

We will refer to the spaces above as inner product spaces in this book.

Example 3.3 (Inner Product that is not the Dot Product)
Consider $V = \mathbb{R}^2$. If we define

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := x_1 y_1 - (x_1 y_2 + x_2 y_1) + 2 x_2 y_2 \tag{3.10}$$

then $\langle \cdot, \cdot \rangle$ is an inner product but different from the dot product. The proof will be an exercise.

3.2.3 Symmetric, Positive Definite Matrices

Symmetric, positive definite matrices play an important role in machine learning, and they are defined via the inner product.

Consider an n-dimensional vector space V with an inner product $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ (see Definition 3.3) and an ordered basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of V. Recall from Section 2.6.1 that any vectors $\boldsymbol{x}, \boldsymbol{y} \in V$ can be written as linear combinations of the basis vectors so that $\boldsymbol{x} = \sum_{i=1}^{n} \psi_i \boldsymbol{b}_i \in V$ and

$\boldsymbol{y} = \sum_{j=1}^{n} \lambda_j \boldsymbol{b}_j \in V$ for suitable $\psi_i, \lambda_j \in \mathbb{R}$. Due to the bilinearity of the inner product it holds that for all $\boldsymbol{x}, \boldsymbol{y} \in V$ that

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \left\langle \sum_{i=1}^{n} \psi_i \boldsymbol{b}_i, \sum_{j=1}^{n} \lambda_j \boldsymbol{b}_j \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \psi_i \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle \lambda_j = \hat{\boldsymbol{x}}^\top \boldsymbol{A} \hat{\boldsymbol{y}}\,, \quad (3.11)$$

where $A_{ij} := \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ and $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$ are the coordinates of $\boldsymbol{x}$ and $\boldsymbol{y}$ with respect to the basis B. This implies that the inner product $\langle \cdot, \cdot \rangle$ is uniquely determined through $\boldsymbol{A}$. The symmetry of the inner product also means that $\boldsymbol{A}$ is symmetric. Furthermore, the positive definiteness of the inner product implies that

$$\forall \boldsymbol{x} \in V \backslash \{\boldsymbol{0}\} : \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} > 0\,. \quad (3.12)$$

Definition 3.4 (Symmetric, positive definite matrix)**.** A symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ that satisfies (3.12) is called *symmetric, positive definite* or just *positive definite*. If only $\geqslant$ holds in (3.12) then $\boldsymbol{A}$ is called *symmetric, positive semi-definite*.

symmetric, positive definite
positive definite
symmetric, positive semi-definite

Example 3.4 (Symmetric, Positive Definite Matrices)
Consider the following matrices:

$$\boldsymbol{A}_1 = \begin{bmatrix} 9 & 6 \\ 6 & 5 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 9 & 6 \\ 6 & 3 \end{bmatrix} \quad (3.13)$$

Then, $\boldsymbol{A}_1$ is positive definite because it is symmetric and

$$\boldsymbol{x}^\top \boldsymbol{A}_1 \boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 9 & 6 \\ 6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (3.14a)$$

$$= 9x_1^2 + 12x_1x_2 + 5x_2^2 = (3x_1 + 2x_2)^2 + x_2^2 > 0 \quad (3.14b)$$

for all $\boldsymbol{x} \in V \setminus \{\boldsymbol{0}\}$. However, $\boldsymbol{A}_2$ is symmetric but not positive definite because $\boldsymbol{x}^\top \boldsymbol{A}_2 \boldsymbol{x} = 9x_1^2 + 12x_1x_2 + 3x_2^2 = (3x_1 + 2x_2)^2 - x_2^2$ can be smaller than 0, e.g., for $\boldsymbol{x} = [2, -3]^\top$.

If $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric, positive definite then

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \hat{\boldsymbol{x}}^\top \boldsymbol{A} \hat{\boldsymbol{y}} \quad (3.15)$$

defines an inner product with respect to an ordered basis B where $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are the coordinate representations of $\boldsymbol{x}, \boldsymbol{y} \in V$ with respect to B.

Theorem 3.5. *For a real-valued, finite-dimensional vector space V and an ordered basis B of V it holds that $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ is an inner product if and only if there exists a symmetric, positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with*

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \hat{\boldsymbol{x}}^\top \boldsymbol{A} \hat{\boldsymbol{y}}\,. \quad (3.16)$$

The following properties hold if $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric and positive definite:

- The null space (kernel) of $\boldsymbol{A}$ consists only of $\mathbf{0}$ because $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \mathbf{0}$. This implies that $\boldsymbol{A}\boldsymbol{x} \neq \mathbf{0}$ if $\boldsymbol{x} \neq \mathbf{0}$.
- The diagonal elements a_{ii} of $\boldsymbol{A}$ are positive because $a_{ii} = \boldsymbol{e}_i^\top \boldsymbol{A}\boldsymbol{e}_i > 0$, where $\boldsymbol{e}_i$ is the ith vector of the standard basis in $\mathbb{R}^n$.

In Section 4.3, we will return to symmetric, positive definite matrices in the context of matrix decompositions.

3.3 Lengths and Distances

In Section 3.1, we already discussed norms that we can use to compute the length of a vector. Inner products and norms are closely related in the sense that any inner product induces a norm

Inner products induce norms.

$$\|\boldsymbol{x}\| := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \tag{3.17}$$

in a natural way, such that we can compute lengths of vectors using the inner product. However, not every norm is induced by an inner product. The Manhattan norm (3.3) is an example of a norm that is not induced by an inner product. In the following, we will focus on norms that are induced by inner products and introduce geometric concepts, such as lengths, distances and angles.

Example 3.5 (Lengths of Vectors using Inner Products)
In geometry, we are often interested in lengths of vectors. We can now use an inner product to compute them using (3.17). Let us take $\boldsymbol{x} = [1, 1]^\top \in \mathbb{R}^2$. If we use the dot product as the inner product, with (3.17) we obtain

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^\top \boldsymbol{x}} = \sqrt{1^2 + 1^2} = \sqrt{2} \tag{3.18}$$

as the length of $\boldsymbol{x}$. Let us now choose a different inner product:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x}^\top \begin{bmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{bmatrix} \boldsymbol{y} = x_1 y_1 - \frac{1}{2}(x_1 y_2 + x_2 y_1) + x_2 y_2 \,. \tag{3.19}$$

If we compute the norm of a vector, then this inner product returns smaller values than the dot product if x_1 and x_2 have the same sign (and $x_1 x_2 > 0$), otherwise it returns greater values than the dot product. With this inner product we obtain

$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle = x_1^2 - x_1 x_2 + x_2^2 = 1 - 1 + 1 = 1 \implies \|\boldsymbol{x}\| = \sqrt{1} = 1 \,, \tag{3.20}$$

such that $\boldsymbol{x}$ is "shorter" with this inner product than with the dot product.

Definition 3.6 (Distance and Metric)**.** Consider an inner product space $(V, \langle \cdot, \cdot \rangle)$. Then

$$d(\boldsymbol{x}, \boldsymbol{y}) := \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle} \tag{3.21}$$

distance
Euclidean distance

is called *distance* of $\boldsymbol{x}, \boldsymbol{y} \in V$. If we use the dot product as the inner product, then the distance is called *Euclidean distance*. The mapping

$$d : V \times V \to \mathbb{R} \tag{3.22}$$
$$(\boldsymbol{x}, \boldsymbol{y}) \mapsto d(\boldsymbol{x}, \boldsymbol{y}) \tag{3.23}$$

metric

is called *metric*.

Remark. Similar to the length of a vector, the distance between vectors does not require an inner product: a norm is sufficient. If we have a norm induced by an inner product, the distance may vary depending on the choice of the inner product. ♢

A metric d satisfies:

positive definite
1. d is *positive definite*, i.e., $d(\boldsymbol{x}, \boldsymbol{y}) \geqslant 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in V$ and $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \iff \boldsymbol{x} = \boldsymbol{y}$

symmetric
2. d is *symmetric*, i.e., $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in V$.

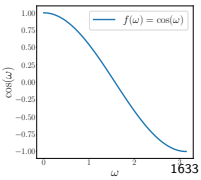
Triangular inequality
3. *Triangular inequality*: $d(\boldsymbol{x}, \boldsymbol{z}) \leqslant d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$.

3.4 Angles and Orthogonality

The Cauchy-Schwarz inequality (3.5) allows us to define angles ω in inner product spaces between two vectors $\boldsymbol{x}, \boldsymbol{y}$. Assume that $\boldsymbol{x} \neq \boldsymbol{0}, \boldsymbol{y} \neq \boldsymbol{0}$. Then

$$-1 \leqslant \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \, \|\boldsymbol{y}\|} \leqslant 1 \,. \tag{3.24}$$

Figure 3.4 When restricted to $[0, \pi]$ then $f(\omega) = \cos(\omega)$ returns a unique number in the interval $[-1, 1]$.



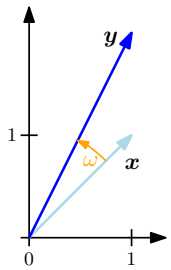
Therefore, there exists a unique $\omega \in [0, \pi]$ with

$$\cos \omega = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \, \|\boldsymbol{y}\|} \,, \tag{3.25}$$

angle

see Figure 3.4 for an illustration. The number ω is the *angle* between the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Intuitively, the angle between two vectors tells us how similar their orientations are. For example, using the dot product, the angle between $\boldsymbol{x}$ and $\boldsymbol{y} = 4\boldsymbol{x}$, i.e., $\boldsymbol{y}$ is a scaled version of $\boldsymbol{x}$, is 0: Their orientation is the same.

Figure 3.5 The angle ω between two vectors $\boldsymbol{x}, \boldsymbol{y}$ is computed using the inner product.



Example 3.6 (Angle between Vectors)
Let us compute the angle between $\boldsymbol{x} = [1, 1]^\top \in \mathbb{R}^2$ and $\boldsymbol{y} = [1, 2]^\top \in \mathbb{R}^2$, see Figure 3.5, where we use the dot product as the inner product. Then we get

$$\cos \omega = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle \, \langle \boldsymbol{y}, \boldsymbol{y} \rangle}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\sqrt{\boldsymbol{x}^\top \boldsymbol{x} \boldsymbol{y}^\top \boldsymbol{y}}} = \frac{3}{\sqrt{10}} \,, \tag{3.26}$$

and the angle between the two vectors is $\arccos(\frac{3}{\sqrt{10}}) \approx 0.32\,\text{rad}$, which corresponds to about $18°$.

The inner product also allows us to characterize vectors that are orthogonal.

Definition 3.7 (Orthogonality)**.** Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are *orthogonal* if and only if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, and we write $\boldsymbol{x} \perp \boldsymbol{y}$. If additionally $\|\boldsymbol{x}\| = 1 = \|\boldsymbol{y}\|$, i.e., the vectors are unit vectors, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are *orthonormal*. orthogonal orthonormal

An implication of this definition is that the $\mathbf{0}$-vector is orthogonal to every vector in the vector space.

Remark. Orthogonality is the generalization of the concept of perpendicularity to bilinear forms that do not have to be the dot product. In our context, geometrically, we can think of orthogonal vectors as having a right angle with respect to a specific inner product. ♢

Example 3.7 (Orthogonal Vectors)

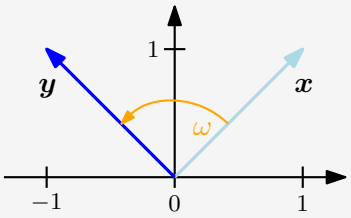


Figure 3.6 The angle ω between two vectors $\boldsymbol{x}, \boldsymbol{y}$ can change depending on the inner product.

Consider two vectors $\boldsymbol{x} = [1, 1]^\top, \boldsymbol{y} = [-1, 1]^\top \in \mathbb{R}^2$, see Figure 3.6. We are interested in determining the angle ω between them using two different inner products. Using the dot product as inner product yields an angle ω between $\boldsymbol{x}$ and $\boldsymbol{y}$ of $90°$, such that $\boldsymbol{x} \perp \boldsymbol{y}$. However, if we choose the inner product

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{y}\,, \tag{3.27}$$

we get that the angle ω between $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by

$$\cos\omega = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} = -\frac{1}{3} \implies \omega \approx 1.91\,\text{rad} \approx 109.5°\,, \tag{3.28}$$

and $\boldsymbol{x}$ and $\boldsymbol{y}$ are not orthogonal. Therefore, vectors that are orthogonal with respect to one inner product do not have to be orthogonal with respect to a different inner product.

Definition 3.8 (Orthogonal Matrix)**.** A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is an *orthogonal matrix* if and only if its columns are orthonormal so that orthogonal matrix

$$\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{I} = \boldsymbol{A}^\top \boldsymbol{A}\,, \tag{3.29}$$

which implies that

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^\top , \tag{3.30}$$

It is convention to call these matrices "orthogonal" but a more precise description would be "orthonormal".

i.e., the inverse is obtained by simply transposing the matrix.

Remark. Transformations by orthogonal matrices are special because the length of a vector $\boldsymbol{x}$ is not changed when transforming it using an orthogonal matrix $\boldsymbol{A}$. For the dot product we obtain

$$\|\boldsymbol{A}\boldsymbol{x}\|^2 = (\boldsymbol{A}\boldsymbol{x})^\top(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^\top\boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}^\top\boldsymbol{x} = \|\boldsymbol{x}\|^2 . \tag{3.31}$$

Transformations with orthogonal matrices preserve distances and angles.

Moreover, the angle between any two vectors $\boldsymbol{x}, \boldsymbol{y}$, as measured by their inner product, is also unchanged when transforming both of them using an orthogonal matrix $\boldsymbol{A}$. Assuming the dot product as the inner product, the angle of the images $\boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{A}\boldsymbol{y}$ is given as

$$\cos\omega = \frac{(\boldsymbol{A}\boldsymbol{x})^\top(\boldsymbol{A}\boldsymbol{y})}{\|\boldsymbol{A}\boldsymbol{x}\|\,\|\boldsymbol{A}\boldsymbol{y}\|} = \frac{\boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{y}}{\sqrt{\boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x}\boldsymbol{y}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{y}}} = \frac{\boldsymbol{x}^\top\boldsymbol{y}}{\|\boldsymbol{x}\|\,\|\boldsymbol{y}\|} , \tag{3.32}$$

which gives exactly the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$. This means that orthogonal matrices $\boldsymbol{A}$ with $\boldsymbol{A}^\top = \boldsymbol{A}^{-1}$ preserve both angles and distances. ♢

3.5 Orthonormal Basis

In Section 2.6.1, we characterized properties of basis vectors and found that in an n-dimensional vector space, we need n basis vectors, i.e., n vectors that are linearly independent. In Sections 3.3 and 3.4, we used inner products to compute the length of vectors and the angle between vectors. In the following, we will discuss the special case where the basis vectors are orthogonal to each other and where the length of each basis vector is 1. We will call this basis then an orthonormal basis.

Let us introduce this more formally.

Definition 3.9 (Orthonormal basis)**.** Consider an n-dimensional vector space V and a basis $\{\boldsymbol{b}_1, \dots, \boldsymbol{b}_n\}$ of V. If

$$\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = 0 \quad \text{for } i \neq j \tag{3.33}$$

$$\langle \boldsymbol{b}_i, \boldsymbol{b}_i \rangle = 1 \tag{3.34}$$

orthonormal basis
ONB
orthogonal basis

for all $i, j = 1, \dots, n$ then the basis is called an *orthonormal basis* (*ONB*). If only (3.33) is satisfied then the basis is called an *orthogonal basis*.

Note that (3.34) implies that every basis vector has length/norm 1. The Gram-Schmidt process (Strang, 2003) is a constructive way to iteratively build an orthonormal basis $\{\boldsymbol{b}_1, \dots, \boldsymbol{b}_n\}$ given a set $\{\tilde{\boldsymbol{b}}_1, \dots, \tilde{\boldsymbol{b}}_n\}$ of non-orthogonal and unnormalized basis vectors.

Example 3.8 (Orthonormal basis)
The canonical/standard basis for a Euclidean vector space $\mathbb{R}^n$ is an orthonormal basis, where the inner product is the dot product of vectors.

In $\mathbb{R}^2$, the vectors

$$\boldsymbol{b}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, \quad \boldsymbol{b}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \tag{3.35}$$

form an orthonormal basis since $\boldsymbol{b}_1^\top \boldsymbol{b}_2 = 0$ and $\|\boldsymbol{b}_1\| = 1 = \|\boldsymbol{b}_2\|$.

We will exploit the concept of an orthonormal basis in Chapter 12 and Chapter 10 when we discuss Support Vector Machines and Principal Component Analysis.

3.6 Inner Product of Functions

Thus far, we looked at properties of inner products to compute lengths, angles and distances. We focused on inner products of finite-dimensional vectors.

In the following, we will look at an example of inner products of a different type of vectors: inner products of functions.

The inner products we discussed so far were defined for vectors with a finite number of entries. We can think of a vector $\boldsymbol{x} \in \mathbb{R}^n$ as function with n function values. The concept of an inner product can be generalized to vectors with an infinite number of entries (countably infinite) and also continuous-valued functions (uncountably infinite). Then, the sum over individual components of vectors, see (3.6) for example, turns into an integral.

An inner product of two functions $u : \mathbb{R} \to \mathbb{R}$ and $v : \mathbb{R} \to \mathbb{R}$ can be defined as the definite integral

$$\langle u, v\rangle := \int_a^b u(x)v(x)dx \tag{3.36}$$

for lower and upper limits $a, b < \infty$, respectively. As with our usual inner product, we can define norms and orthogonality by looking at the inner product. If (3.36) evaluates to 0, the functions u and v are orthogonal. To make the above inner product mathematically precise, we need to take care of measures, and the definition of integrals. Furthermore, unlike inner product on finite-dimensional vectors, inner products on functions may diverge (have infinite value). Some careful definitions need to be observed, which requires a foray into real and functional analysis which we do not cover in this book.

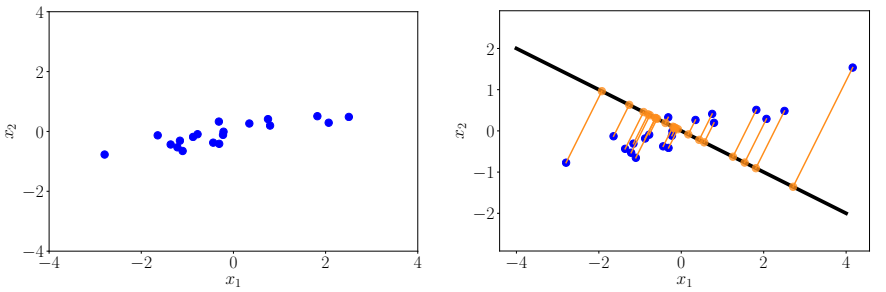


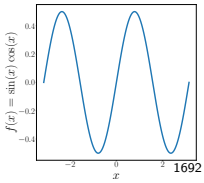
Figure 3.8 Orthogonal projection of a two-dimensional data set onto a one-dimensional subspace.

(a) Original dataset.

(b) Original data (blue) and their corresponding orthogonal projections (orange) onto a lower-dimensional subspace (straight line).

Example 3.9 (Inner Product of Functions)
If we choose $u = \sin(x)$ and $v = \cos(x)$, the integrand $f(x) = u(x)v(x)$ of (3.36), is shown in Figure 3.7. We see that this function is odd, i.e., $f(-x) = -f(x)$. Therefore, the integral with limits $a = -\pi, b = \pi$ of this product evaluates to 0. Therefore, sin and cos are orthogonal functions.

Figure 3.7 $f(x) = \sin(x)\cos(x)$.



Remark. It also holds that the collection of functions

$$\{1, \cos(x), \cos(2x), \cos(3x), \dots\} \tag{3.37}$$

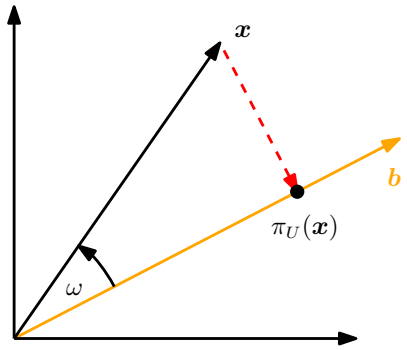
is orthogonal if we integrate from $-\pi$ to π, i.e., any pair of functions are orthogonal to each other. ♢

In Chapter 6, we will have a look at a second type of unconventional inner products: the inner product of random variables.

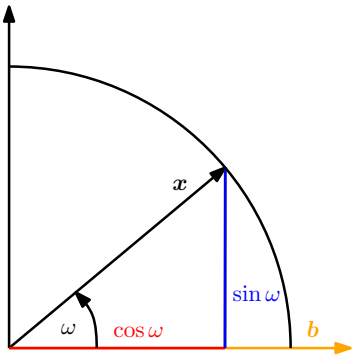
3.7 Orthogonal Projections

Projections are an important class of linear transformations (besides rotations and reflections). Projections play an important role in graphics, coding theory, statistics and machine learning. In machine learning, we often deal with data that is high-dimensional. High-dimensional data is often hard to analyze or visualize. However, high-dimensional data quite often possesses the property that only a few dimensions contain most information, and most other dimensions are not essential to describe key properties of the data. When we compress or visualize high-dimensional data we will lose information. To minimize this compression loss, we ideally find the most informative dimensions in the data. Then, we can project the original high-dimensional data onto a lower-dimensional feature space and work in this lower-dimensional space to learn more about the dataset and extract patterns. For example, machine learning algorithms, such as Principal Component Analysis (PCA) by Pearson (1901b);

"Feature" is a commonly used word for "data representation".



(a) Projection of $\boldsymbol{x} \in \mathbb{R}^2$ onto a subspace U with basis vector $\boldsymbol{b}$.



(b) Projection of a two-dimensional vector $\boldsymbol{x}$ with $\|\boldsymbol{x}\| = 1$ onto a one-dimensional subspace spanned by $\boldsymbol{b}$.

Figure 3.9 Examples of projections onto one-dimensional subspaces.

Hotelling (1933) and Deep Neural Networks (e.g., deep auto-encoders Deng et al. (2010)), heavily exploit the idea of dimensionality reduction. In the following, we will focus on orthogonal projections, which we will use in Chapter 10 for linear dimensionality reduction and in Chapter 12 for classification. Even linear regression, which we discuss in Chapter 9, can be interpreted using orthogonal projections. For a given lower-dimensional subspace, orthogonal projections of high-dimensional data retain as much information as possible and minimize the difference/error between the original data and the corresponding projection. An illustration of such an orthogonal projection is given in Figure 3.8.

Before we detail how to obtain these projections, let us define what a projection actually is.

Definition 3.10 (Projection)**.** Let V be a vector space and $W \subseteq V$ a subspace of V. A linear mapping $\pi : V \to W$ is called a *projection* if $\pi^2 = \pi \circ \pi = \pi$. projection

Remark (Projection matrix). Since linear mappings can be expressed by transformation matrices (see Section 2.7), the definition above applies equally to a special kind of transformation matrices, the *projection matrices* $\boldsymbol{P}_\pi$, which exhibit the property that $\boldsymbol{P}_\pi^2 = \boldsymbol{P}_\pi$. projection matrices

♢

In the following, we will derive orthogonal projections of vectors in the inner product space $(\mathbb{R}^n, \langle \cdot, \cdot \rangle)$ onto subspaces. We will start with one-dimensional subspaces, which are also called *lines*. If not mentioned otherwise, we assume the dot product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \boldsymbol{y}$ as the inner product. lines

3.7.1 Projection onto 1-Dimensional Subspaces (Lines)

Assume we are given a line (1-dimensional subspace) through the origin with basis vector $\boldsymbol{b} \in \mathbb{R}^n$. The line is a one-dimensional subspace $U \subseteq \mathbb{R}^n$

spanned by $\boldsymbol{b}$. When we project $\boldsymbol{x} \in \mathbb{R}^n$ onto U, we want to find the point $\pi_U(\boldsymbol{x}) \in U$ that is closest to $\boldsymbol{x}$. Using geometric arguments, let us characterize some properties of the projection $\pi_U(\boldsymbol{x})$ (Fig. 3.9 serves as an illustration):

- The projection $\pi_U(\boldsymbol{x})$ is closest to $\boldsymbol{x}$, where "closest" implies that the distance $\|\boldsymbol{x} - \pi_U(\boldsymbol{x})\|$ is minimal. It follows that the segment $\pi_U(\boldsymbol{x}) - \boldsymbol{x}$ from $\pi_U(\boldsymbol{x})$ to $\boldsymbol{x}$ is orthogonal to U and, therefore, the basis $\boldsymbol{b}$ of U. The orthogonality condition yields $\langle \pi_U(\boldsymbol{x}) - \boldsymbol{x}, \boldsymbol{b} \rangle = 0$ since angles between vectors are defined by means of the inner product.
- The projection $\pi_U(\boldsymbol{x})$ of $\boldsymbol{x}$ onto U must be an element of U and, therefore, a multiple of the basis vector $\boldsymbol{b}$ that spans U. Hence, $\pi_U(\boldsymbol{x}) = \lambda \boldsymbol{b}$, for some $\lambda \in \mathbb{R}$.

λ is then the coordinate of $\pi_U(\boldsymbol{x})$ with respect to $\boldsymbol{b}$.

In the following three steps, we determine the coordinate λ, the projection $\pi_U(\boldsymbol{x}) \in U$ and the projection matrix $\boldsymbol{P}_\pi$ that maps arbitrary $\boldsymbol{x} \in \mathbb{R}^n$ onto U.

1. Finding the coordinate λ. The orthogonality condition yields

$$\langle \boldsymbol{x} - \pi_U(\boldsymbol{x}), \boldsymbol{b} \rangle = 0 \tag{3.38}$$

$$\overset{\pi_U(\boldsymbol{x}) = \lambda \boldsymbol{b}}{\iff} \langle \boldsymbol{x} - \lambda \boldsymbol{b}, \boldsymbol{b} \rangle = 0 \,. \tag{3.39}$$

With a general inner product, we get $\lambda = \langle \boldsymbol{x}, \boldsymbol{b} \rangle$ if $\|\boldsymbol{b}\| = 1$.

We can now exploit the bilinearity of the inner product and arrive at

$$\langle \boldsymbol{x}, \boldsymbol{b} \rangle - \lambda \langle \boldsymbol{b}, \boldsymbol{b} \rangle = 0 \tag{3.40}$$

$$\iff \lambda = \frac{\langle \boldsymbol{x}, \boldsymbol{b} \rangle}{\langle \boldsymbol{b}, \boldsymbol{b} \rangle} = \frac{\langle \boldsymbol{x}, \boldsymbol{b} \rangle}{\|\boldsymbol{b}\|^2} \tag{3.41}$$

If we choose $\langle \cdot, \cdot \rangle$ to be the dot product, we obtain

$$\lambda = \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\boldsymbol{b}^\top \boldsymbol{b}} = \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|\boldsymbol{b}\|^2} \tag{3.42}$$

If $\|\boldsymbol{b}\| = 1$, then the coordinate λ of the projection is given by $\boldsymbol{b}^\top \boldsymbol{x}$.

2. Finding the projection point $\pi_U(\boldsymbol{x}) \in U$. Since $\pi_U(\boldsymbol{x}) = \lambda \boldsymbol{b}$ we immediately obtain with (3.42) that

$$\pi_U(\boldsymbol{x}) = \lambda \boldsymbol{b} = \frac{\langle \boldsymbol{x}, \boldsymbol{b} \rangle}{\|\boldsymbol{b}\|^2} \boldsymbol{b} = \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|\boldsymbol{b}\|^2} \boldsymbol{b} \,, \tag{3.43}$$

where the last equality holds for the dot product only. We can also compute the length of $\pi_U(\boldsymbol{x})$ by means of Definition 3.1 as

$$\|\pi_U(\boldsymbol{x})\| = \|\lambda \boldsymbol{b}\| = |\lambda| \, \|\boldsymbol{b}\| \,. \tag{3.44}$$

This means that our projection is of length $|\lambda|$ times the length of $\boldsymbol{b}$. This also adds the intuition that λ is the coordinate of $\pi_U(\boldsymbol{x})$ with respect to the basis vector $\boldsymbol{b}$ that spans our one-dimensional subspace U.

If we use the dot product as an inner product we get

$$\|\pi_U(\boldsymbol{x})\| \overset{(3.43)}{=} \frac{|\boldsymbol{b}^\top \boldsymbol{x}|}{\|\boldsymbol{b}\|^2}\|\boldsymbol{b}\| \overset{(3.25)}{=} |\cos\omega|\,\|\boldsymbol{x}\|\,\|\boldsymbol{b}\|\frac{\|\boldsymbol{b}\|}{\|\boldsymbol{b}\|^2} = |\cos\omega|\,\|\boldsymbol{x}\|\,. \tag{3.45}$$

Here, ω is the angle between $\boldsymbol{x}$ and $\boldsymbol{b}$. This equation should be familiar from trigonometry: If $\|\boldsymbol{x}\| = 1$ then $\boldsymbol{x}$ lies on the unit circle. It follows that the projection onto the horizontal axis spanned by $\boldsymbol{b}$ is exactly $\cos\omega$, and the length of the corresponding vector $\pi_U(\boldsymbol{x}) = |\cos\omega|$. An illustration is given in Figure 3.9.

The horizontal axis is a one-dimensional subspace.

3. Finding the projection matrix $\boldsymbol{P}_\pi$. We know that a projection is a linear mapping (see Definition 3.10). Therefore, there exists a projection matrix $\boldsymbol{P}_\pi$, such that $\pi_U(\boldsymbol{x}) = \boldsymbol{P}_\pi \boldsymbol{x}$. With the dot product as inner product and

$$\pi_U(\boldsymbol{x}) = \lambda\boldsymbol{b} = \boldsymbol{b}\lambda = \boldsymbol{b}\frac{\boldsymbol{b}^\top\boldsymbol{x}}{\|\boldsymbol{b}\|^2} = \frac{\boldsymbol{b}\boldsymbol{b}^\top}{\|\boldsymbol{b}\|^2}\boldsymbol{x} \tag{3.46}$$

we immediately see that

$$\boldsymbol{P}_\pi = \frac{\boldsymbol{b}\boldsymbol{b}^\top}{\|\boldsymbol{b}\|^2}. \tag{3.47}$$

Note that $\boldsymbol{b}\boldsymbol{b}^\top$ is a symmetric matrix (with rank 1) and $\|\boldsymbol{b}\|^2 = \langle\boldsymbol{b},\boldsymbol{b}\rangle$ is a scalar.

Projection matrices are always symmetric.

The projection matrix $\boldsymbol{P}_\pi$ projects any vector $\boldsymbol{x} \in \mathbb{R}^n$ onto the line through the origin with direction $\boldsymbol{b}$ (equivalently, the subspace U spanned by $\boldsymbol{b}$).

Remark. The projection $\pi_U(\boldsymbol{x}) \in \mathbb{R}^n$ is still an n-dimensional vector and not a scalar. However, we no longer require n coordinates to represent the projection, but only a single one if we want to express it with respect to the basis vector $\boldsymbol{b}$ that spans the subspace U: λ. ♢

Example 3.10 (Projection onto a Line)

Find the projection matrix $\boldsymbol{P}_\pi$ onto the line through the origin spanned by $\boldsymbol{b} = \begin{bmatrix}1 & 2 & 2\end{bmatrix}^\top$. $\boldsymbol{b}$ is a direction and a basis of the one-dimensional subspace (line through origin).

With (3.47), we obtain

$$\boldsymbol{P}_\pi = \frac{\boldsymbol{b}\boldsymbol{b}^\top}{\boldsymbol{b}^\top\boldsymbol{b}} = \frac{1}{9}\begin{bmatrix}1\\2\\2\end{bmatrix}\begin{bmatrix}1 & 2 & 2\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 2 & 2\\2 & 4 & 4\\2 & 4 & 4\end{bmatrix}. \tag{3.48}$$

Let us now choose a particular $\boldsymbol{x}$ and see whether it lies in the subspace

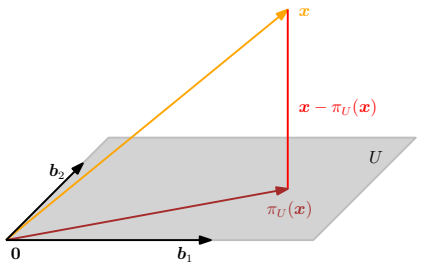


Figure 3.10 Projection onto a two-dimensional subspace U with basis $\boldsymbol{b}_1, \boldsymbol{b}_2$. The projection $\pi_U(\boldsymbol{x})$ of $\boldsymbol{x} \in \mathbb{R}^3$ onto U can be expressed as a linear combination of $\boldsymbol{b}_1, \boldsymbol{b}_2$ and the displacement vector $\boldsymbol{x} - \pi_U(\boldsymbol{x})$ is orthogonal to both $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$.

spanned by $\boldsymbol{b}$. For $\boldsymbol{x} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^\top$, the projection is

$$\pi_U(\boldsymbol{x}) = \boldsymbol{P}_\pi \boldsymbol{x} = \frac{1}{9}\begin{bmatrix}1 & 2 & 2\\ 2 & 4 & 4\\ 2 & 4 & 4\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}5\\10\\10\end{bmatrix} \in \operatorname{span}\left[\begin{bmatrix}1\\2\\2\end{bmatrix}\right]. \tag{3.49}$$

Note that the application of $\boldsymbol{P}_\pi$ to $\pi_U(\boldsymbol{x})$ does not change anything, i.e., $\boldsymbol{P}_\pi \pi_U(\boldsymbol{x}) = \pi_U(\boldsymbol{x})$. This is expected because according to Definition 3.10 we know that a projection matrix $\boldsymbol{P}_\pi$ satisfies $\boldsymbol{P}_\pi^2 \boldsymbol{x} = \boldsymbol{P}_\pi \boldsymbol{x}$ for all $\boldsymbol{x}$.

With the results from Chapter 4 we can show that $\pi_U(\boldsymbol{x})$ is also an eigenvector of $\boldsymbol{P}_\pi$, and the corresponding eigenvalue is 1.

If U is given by a set of spanning vectors, which are not a basis, make sure you determine a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ before proceeding.

The basis vectors form the columns of $\boldsymbol{B} \in \mathbb{R}^{n \times m}$, where $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m]$.

3.7.2 Projection onto General Subspaces

In the following, we look at orthogonal projections of vectors $\boldsymbol{x} \in \mathbb{R}^n$ onto higher-dimensional subspaces $U \subseteq \mathbb{R}^n$ with $\dim(U) = m \geqslant 1$. An illustration is given in Figure 3.10.

Assume that $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ is an ordered basis of U. Any projection $\pi_U(\boldsymbol{x})$ onto U is necessarily an element of U. Therefore, they can be represented as linear combinations of the basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ of U, such that $\pi_U(\boldsymbol{x}) = \sum_{i=1}^m \lambda_i \boldsymbol{b}_i$.

As in the 1D case, we follow a three-step procedure to find the projection $\pi_U(\boldsymbol{x})$ and the projection matrix $\boldsymbol{P}_\pi$:

1. Find the coordinates $\lambda_1, \ldots, \lambda_m$ of the projection (with respect to the basis of U), such that the linear combination

$$\pi_U(\boldsymbol{x}) = \sum_{i=1}^m \lambda_i \boldsymbol{b}_i = \boldsymbol{B}\boldsymbol{\lambda}, \tag{3.50}$$

$$\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m] \in \mathbb{R}^{n \times m}, \ \boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_m]^\top \in \mathbb{R}^m, \tag{3.51}$$

is closest to $\boldsymbol{x} \in \mathbb{R}^n$. As in the 1D case, "closest" means "minimum distance", which implies that the vector connecting $\pi_U(\boldsymbol{x}) \in U$ and $\boldsymbol{x} \in \mathbb{R}^n$ must be orthogonal to all basis vectors of U. Therefore, we obtain m simultaneous conditions (assuming the dot product as the inner product)

$$\langle \boldsymbol{b}_1, \boldsymbol{x} - \pi_U(\boldsymbol{x}) \rangle = \boldsymbol{b}_1^\top (\boldsymbol{x} - \pi_U(\boldsymbol{x})) = 0 \tag{3.52}$$

$$\vdots \tag{3.53}$$

$$\langle \boldsymbol{b}_m, \boldsymbol{x} - \pi_U(\boldsymbol{x}) \rangle = \boldsymbol{b}_m^\top (\boldsymbol{x} - \pi_U(\boldsymbol{x})) = 0 \tag{3.54}$$

which, with $\pi_U(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{\lambda}$, can be written as

$$\boldsymbol{b}_1^\top (\boldsymbol{x} - \boldsymbol{B}\boldsymbol{\lambda}) = 0 \tag{3.55}$$

$$\vdots \tag{3.56}$$

$$\boldsymbol{b}_m^\top (\boldsymbol{x} - \boldsymbol{B}\boldsymbol{\lambda}) = 0 \tag{3.57}$$

such that we obtain a homogeneous linear equation system

$$\begin{bmatrix} \boldsymbol{b}_1^\top \\ \vdots \\ \boldsymbol{b}_m^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{x} - \boldsymbol{B}\boldsymbol{\lambda} \end{bmatrix} = \boldsymbol{0} \iff \boldsymbol{B}^\top (\boldsymbol{x} - \boldsymbol{B}\boldsymbol{\lambda}) = \boldsymbol{0} \tag{3.58}$$

$$\iff \boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{B}^\top \boldsymbol{x}\,. \tag{3.59}$$

The last expression is called *normal equation*. Since $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ are a basis of U and, therefore, linearly independent, $\boldsymbol{B}^\top \boldsymbol{B} \in \mathbb{R}^{m \times m}$ is regular and can be inverted. This allows us to solve for the coefficients/ coordinates

normal equation

$$\boldsymbol{\lambda} = (\boldsymbol{B}^\top \boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{x}\,. \tag{3.60}$$

The matrix $(\boldsymbol{B}^\top \boldsymbol{B})^{-1} \boldsymbol{B}^\top$ is also called the *pseudo-inverse* of $\boldsymbol{B}$, which can be computed for non-square matrices $\boldsymbol{B}$. It only requires that $\boldsymbol{B}^\top \boldsymbol{B}$ is positive definite, which is the case if $\boldsymbol{B}$ is full rank.

pseudo-inverse

2. Find the projection $\pi_U(\boldsymbol{x}) \in U$. We already established that $\pi_U(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{\lambda}$. Therefore, with (3.60)

$$\pi_U(\boldsymbol{x}) = \boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{x}\,. \tag{3.61}$$

3. Find the projection matrix $\boldsymbol{P}_\pi$. From (3.61) we can immediately see that the projection matrix that solves $\boldsymbol{P}_\pi \boldsymbol{x} = \pi_U(\boldsymbol{x})$ must be

$$\boldsymbol{P}_\pi = \boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{B})^{-1} \boldsymbol{B}^\top\,. \tag{3.62}$$

In practical applications (e.g., linear regression), we often add a "jitter term" $\epsilon \boldsymbol{I}$ to $\boldsymbol{B}^\top \boldsymbol{B}$ to guarantee increased numerical stability and positive definiteness. This "ridge" can be rigorously derived using Bayesian inference. See Chapter 9 for details.

Remark. Comparing the solutions for projecting onto a one-dimensional subspace and the general case, we see that the general case includes the 1D case as a special case: If $\dim(U) = 1$ then $\boldsymbol{B}^\top \boldsymbol{B} \in \mathbb{R}$ is a scalar and we can rewrite the projection matrix in (3.62) $\boldsymbol{P}_\pi = \boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{B})^{-1} \boldsymbol{B}^\top$ as $\boldsymbol{P}_\pi = \frac{\boldsymbol{B}\boldsymbol{B}^\top}{\boldsymbol{B}^\top \boldsymbol{B}}$, which is exactly the projection matrix in (3.47). ♢

Example 3.11 (Projection onto a Two-dimensional Subspace)

For a subspace $U = \text{span}[\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}] \subseteq \mathbb{R}^3$ and $\boldsymbol{x} = \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^3$ find the

coordinates $\boldsymbol{\lambda}$ of $\boldsymbol{x}$ in terms of the subspace U, the projection point $\pi_U(\boldsymbol{x})$ and the projection matrix $\boldsymbol{P}_\pi$.

First, we see that the generating set of U is a basis (linear independence) and write the basis vectors of U into a matrix $\boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$.

Second, we compute the matrix $\boldsymbol{B}^\top\boldsymbol{B}$ and the vector $\boldsymbol{B}^\top\boldsymbol{x}$ as

$$\boldsymbol{B}^\top\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}, \quad \boldsymbol{B}^\top\boldsymbol{x} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \end{bmatrix}. \tag{3.63}$$

Third, we solve the normal equation $\boldsymbol{B}^\top\boldsymbol{B}\boldsymbol{\lambda} = \boldsymbol{B}^\top\boldsymbol{x}$ to find $\boldsymbol{\lambda}$:

$$\begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \end{bmatrix} \iff \boldsymbol{\lambda} = \begin{bmatrix} 5 \\ -3 \end{bmatrix}. \tag{3.64}$$

Fourth, the projection $\pi_U(\boldsymbol{x})$ of $\boldsymbol{x}$ onto U, i.e., into the column space of $\boldsymbol{B}$, can be directly computed via

$$\pi_U(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{\lambda} = \begin{bmatrix} 5 \\ 2 \\ -1 \end{bmatrix}. \tag{3.65}$$

projection error

The corresponding *projection error* is the norm of the difference vector between the original vector and its projection onto U, i.e.,

$$\|\boldsymbol{x} - \pi_U(\boldsymbol{x})\| = \left\| \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}^\top \right\| = \sqrt{6}. \tag{3.66}$$

The projection error is also called the *reconstruction error*.

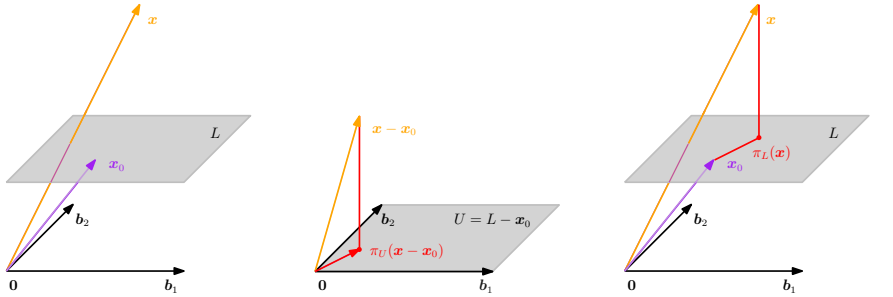
Fifth, the projection matrix (for any $\boldsymbol{x} \in \mathbb{R}^3$) is given by

$$\boldsymbol{P}_\pi = \boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top = \frac{1}{6} \begin{bmatrix} 5 & 2 & -1 \\ 2 & 2 & 2 \\ -1 & 2 & 5 \end{bmatrix}. \tag{3.67}$$

To verify the results, we can (a) check whether the displacement vector $\pi_U(\boldsymbol{x}) - \boldsymbol{x}$ is orthogonal to all basis vectors of U, (b) verify that $\boldsymbol{P}_\pi = \boldsymbol{P}_\pi^2$ (see Definition 3.10).

Remark. The projections $\pi_U(\boldsymbol{x})$ are still vectors in $\mathbb{R}^n$ although they lie in an m-dimensional subspace $U \subseteq \mathbb{R}^n$. However, to represent a projected vector we only need the m coordinates $\lambda_1, \ldots, \lambda_m$ with respect to the basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ of U. ♢

Remark. In vector spaces with general inner products, we have to pay attention when computing angles and distances, which are defined by means of the inner product. ♢



(a) Setting.

(b) Reduce problem to projection π_U onto vector subspace.

(c) Add support point back in to get affine projection π_L.

Figure 3.11 Projection onto an affine space. (a) The original setting; (b) The setting is shifted by $-\boldsymbol{x}_0$, so that $\boldsymbol{x} - \boldsymbol{x}_0$ can be projected onto the direction space U; (c) The projection is translated back to $\boldsymbol{x}_0 + \pi_U(\boldsymbol{x} - \boldsymbol{x}_0)$, which gives the final orthogonal projection $\pi_L(\boldsymbol{x})$.

We can find approximate solutions to unsolvable linear equation systems using projections.

Projections allow us to look at situations where we have a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ without a solution. Recall that this means that $\boldsymbol{b}$ does not lie in the span of $\boldsymbol{A}$, i.e., the vector $\boldsymbol{b}$ does not lie in the subspace spanned by the columns of $\boldsymbol{A}$. Given that the linear equation cannot be solved exactly, we can find an *approximate solution*. The idea is to find the vector in the subspace spanned by the columns of $\boldsymbol{A}$ that is closest to $\boldsymbol{b}$, i.e., we compute the orthogonal projection of $\boldsymbol{b}$ onto the subspace spanned by the columns of $\boldsymbol{A}$. This problem arises often in practice, and the solution is called the *least squares solution* (assuming the dot product as the inner product) of an overdetermined system. This is discussed further in Chapter 9.

least squares solution

Remark. We just looked at projections of vectors $\boldsymbol{x}$ onto a subspace U with basis vectors $\{\boldsymbol{b}_1, \dots, \boldsymbol{b}_k\}$. If this basis is an ONB, i.e., (3.33)–(3.34) are satisfied, the projection equation (3.61) simplifies greatly to

$$\pi_U(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x} \tag{3.68}$$

since $\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}$ with coordinates

$$\boldsymbol{\lambda} = \boldsymbol{B}^\top \boldsymbol{x}\,. \tag{3.69}$$

This means that we no longer have to compute the tedious inverse from (3.61), which saves us much computation time. ♢

3.7.3 Projection onto Affine Subspaces

Thus far, we discussed how to project a vector onto a lower-dimensional subspace U. In the following, we provide a solution to projecting a vector onto an affine subspace.

Consider the setting in Figure 3.11(a). We are given an affine space $L = \boldsymbol{x}_0 + U$ where $\boldsymbol{b}_1, \boldsymbol{b}_2$ are basis vectors of U. To determine the orthogonal projection $\pi_L(\boldsymbol{x})$ of $\boldsymbol{x}$ onto L, we transform the problem into a problem that we know how to solve: the projection onto a vector subspace. In

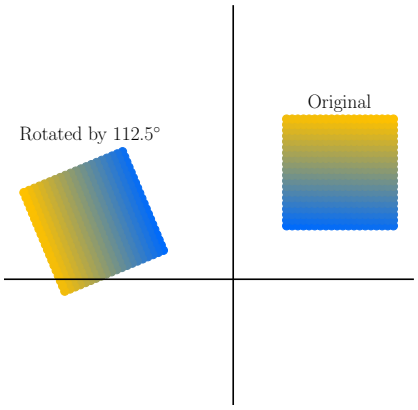


Figure 3.12 A rotation rotates objects in a plane about the origin. If the rotation angle is positive, we rotate counterclockwise.

order to get there, we subtract the support point $\boldsymbol{x}_0$ from $\boldsymbol{x}$ and from L, so that $L - \boldsymbol{x}_0 = U$ is exactly the vector subspace U. We can now use the orthogonal projections onto a subspace we discussed in Section 3.7.2 and obtain the projection $\pi_U(\boldsymbol{x} - \boldsymbol{x}_0)$, which is illustrated in Figure 3.11(b). This projection can now be translated back into L by adding $\boldsymbol{x}_0$, such that we obtain the orthogonal projection onto an affine space L as

$$\pi_L(\boldsymbol{x}) = \boldsymbol{x}_0 + \pi_U(\boldsymbol{x} - \boldsymbol{x}_0)\,, \tag{3.70}$$

where $\pi_U(\cdot)$ is the orthogonal projection onto the subspace U, i.e., the direction space of L, see Figure 3.11(c).

From Figure 3.11 it is also evident that the distance of $\boldsymbol{x}$ from the affine space L is identical to the distance of $\boldsymbol{x} - \boldsymbol{x}_0$ from U, i.e.,

$$\begin{aligned} d(\boldsymbol{x}, L) &= \|\boldsymbol{x} - \pi_L(\boldsymbol{x})\| = \|\boldsymbol{x} - (\boldsymbol{x}_0 + \pi_U(\boldsymbol{x} - \boldsymbol{x}_0))\| && (3.71)\\ &= d(\boldsymbol{x} - \boldsymbol{x}_0, \pi_U(\boldsymbol{x} - \boldsymbol{x}_0))\,. && (3.72) \end{aligned}$$

3.8 Rotations

Length and angle preservation, as discussed in Section 3.4, are the two characteristics of linear mappings with orthogonal transformation matrices. In the following, we will have a closer look at specific orthogonal transformation matrices, which describe rotations.

rotation

A *rotation* is a linear mapping (more specifically, an automorphism of a Euclidean vector space) that rotates a plane by an angle θ about the origin, i.e., the origin is a fixed point. For a positive angle $\theta > 0$, by common convention, we rotate in a counterclockwise direction. An example is shown in Figure 3.12, where the transformation matrix is

$$\boldsymbol{R} = \begin{bmatrix} -0.38 & -0.92 \\ 0.92 & -0.38 \end{bmatrix}\,. \tag{3.73}$$

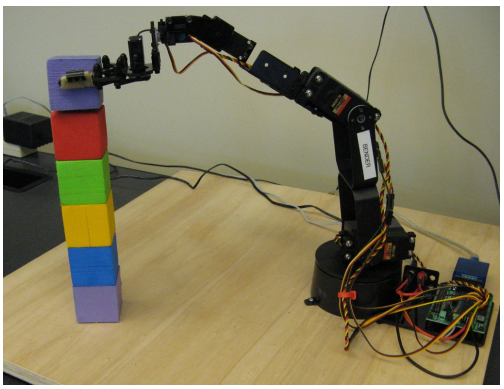


Figure 3.13 The robotic arm needs to rotate its joints in order to pick up objects or to place them correctly. Figure taken from (Deisenroth et al., 2015).

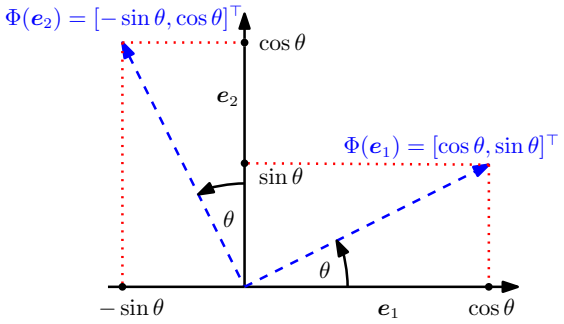


Figure 3.14 Rotation of the standard basis in $\mathbb{R}^2$ by an angle θ.

Important application areas of rotations include computer graphics and robotics. For example, in robotics, it is often important to know how to rotate the joints of a robotic arm in order to pick up or place an object, see Figure 3.13.

3.8.1 Rotations in $\mathbb{R}^2$

Consider the standard basis $\left\{\boldsymbol{e}_1 = \begin{bmatrix}1\\0\end{bmatrix}, \boldsymbol{e}_2 = \begin{bmatrix}0\\1\end{bmatrix}\right\}$ of $\mathbb{R}^2$, which defines the standard coordinate system in $\mathbb{R}^2$. We aim to rotate this coordinate system by an angle θ as illustrated in Figure 3.14. Note that the rotated vectors are still linearly independent and, therefore, are a basis of $\mathbb{R}^2$. This means that the rotation performs a basis change.

Rotations Φ are linear mappings so that we can express them by a *rotation matrix* $\boldsymbol{R}(\theta)$. Trigonometry (see Figure 3.14) allows us to determine the coordinates of the rotated axes (the image of Φ) with respect to the standard basis in $\mathbb{R}^2$. We obtain

rotation matrix

$$\Phi(\boldsymbol{e}_1) = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix}, \quad \Phi(\boldsymbol{e}_2) = \begin{bmatrix}-\sin\theta\\ \cos\theta\end{bmatrix}. \tag{3.74}$$

Therefore, the rotation matrix that performs the basis change into the

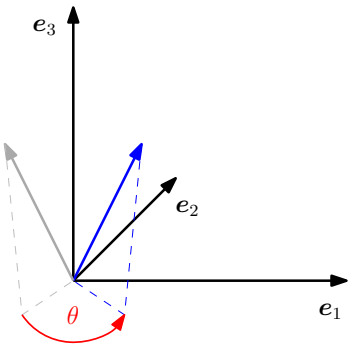


Figure 3.15 Rotation of a vector (gray) in $\mathbb{R}^3$ by an angle θ about the e_3-axis. The rotated vector is shown in blue.

rotated coordinates $\boldsymbol{R}(\theta)$ is given as

$$\boldsymbol{R}(\theta) = \begin{bmatrix}\Phi(\boldsymbol{e}_1) & \Phi(\boldsymbol{e}_2)\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}. \tag{3.75}$$

3.8.2 Rotations in $\mathbb{R}^3$

In contrast to the $\mathbb{R}^2$ case, in $\mathbb{R}^3$ we can rotate any two-dimensional plane about a one-dimensional axis. The easiest way to specify the general rotation matrix is to specify how the images of the standard basis $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ are supposed to be rotated, and making sure these images $\boldsymbol{R}\boldsymbol{e}_1, \boldsymbol{R}\boldsymbol{e}_2, \boldsymbol{R}\boldsymbol{e}_3$ are orthonormal to each other. We can then obtain a general rotation matrix $\boldsymbol{R}$ by combining the images of the standard basis.

To have a meaningful rotation angle we have to define what "counterclockwise" means when we operate in more than two dimensions. We use the convention that a "counterclockwise" (planar) rotation about an axis refers to a rotation about an axis when we look at the axis "head on, from the end toward the origin". In $\mathbb{R}^3$, there are therefore three (planar) rotations about the three standard basis vectors (see Figure 3.15):

- Rotation about the $\boldsymbol{e}_1$-axis

$$\boldsymbol{R}_1(\theta) = \begin{bmatrix}\Phi(\boldsymbol{e}_1) & \Phi(\boldsymbol{e}_2) & \Phi(\boldsymbol{e}_3)\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta\end{bmatrix}. \tag{3.76}$$

  Here, the $\boldsymbol{e}_1$ coordinate is fixed, and the counterclockwise rotation is performed in the $\boldsymbol{e}_2\boldsymbol{e}_3$ plane.
- Rotation about the $\boldsymbol{e}_2$-axis

$$\boldsymbol{R}_2(\theta) = \begin{bmatrix}\cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta\end{bmatrix}. \tag{3.77}$$

  If we rotate the $\boldsymbol{e}_1\boldsymbol{e}_3$ plane about the $\boldsymbol{e}_2$ axis, we need to look at the $\boldsymbol{e}_2$ axis from its "tip" toward the origin.

- Rotation about the $\boldsymbol{e}_3$-axis

$$\boldsymbol{R}_3(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.78}$$

Figure 3.15 illustrates this.

3.8.3 Rotations in n Dimensions

The generalization of rotations from 2D and 3D to n-dimensional Euclidean vector spaces can be intuitively described as fixing $n-2$ dimensions and restrict the rotation to a two-dimensional plane in the n-dimensional space. As in the three-dimensional case we can rotate any plane (two-dimensional subspace of $\mathbb{R}^n$).

Definition 3.11 (Givens Rotation)**.** Let and V be an n-dimensional Euclidean vector space and $\Phi : V \to V$ an automorphism with transformation matrix

$$\boldsymbol{R}_{ij}(\theta) := \begin{bmatrix} \boldsymbol{I}_{i-1} & \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \cos\theta & \boldsymbol{0} & -\sin\theta & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{j-i-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \sin\theta & \boldsymbol{0} & \cos\theta & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} & \boldsymbol{I}_{n-j} \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{3.79}$$

for $1 \leqslant i < j \leqslant n$ and $\theta \in \mathbb{R}$. Then $\boldsymbol{R}_{ij}(\theta)$ is called a *Givens rotation*. Essentially, $\boldsymbol{R}_{ij}(\theta)$ is the identity matrix $\boldsymbol{I}_n$ with Givens rotation

$$r_{ii} = \cos\theta\,, \quad r_{ij} = -\sin\theta\,, \quad r_{ji} = \sin\theta\,, \quad r_{jj} = \cos\theta\,. \tag{3.80}$$

In two dimensions (i.e., $n = 2$), we obtain (3.75) as a special case.

3.8.4 Properties of Rotations

Rotations exhibit a number useful properties:

- Rotations preserve distances, i.e., $\|\boldsymbol{x}-\boldsymbol{y}\| = \|\boldsymbol{R}_\theta(\boldsymbol{x})-\boldsymbol{R}_\theta(\boldsymbol{y})\|$. In other words, rotations leave the distance between any two points unchanged after the transformation.
- Rotations preserve angles, i.e., the angle between $\boldsymbol{R}_\theta\boldsymbol{x}$ and $\boldsymbol{R}_\theta\boldsymbol{y}$ equals the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Rotations in three (or more) dimensions are generally not commutative. Therefore, the order in which rotations are applied is important, even if they rotate about the same point. Only in two dimensions vector rotations are commutative, such that $\boldsymbol{R}(\phi)\boldsymbol{R}(\theta) = \boldsymbol{R}(\theta)\boldsymbol{R}(\phi)$ for all $\phi, \theta \in [0, 2\pi)$, and form an Abelian group (with multiplication) only if they rotate about the same point (e.g., the origin).

3.9 Further Reading

In this chapter, we gave a brief overview of some of the important concepts of analytic geometry, which we will use in later chapters of the book. For a broader and more in-depth overview of some the concepts we presented we refer to the following excellent books: Axler (2015) and Boyd and Vandenberghe (2018).

Inner products allow us to determine specific bases of vector (sub)spaces, where each vector is orthogonal to all others (orthogonal bases) using the Gram-Schmidt method. These bases are important in optimization and numerical algorithms for solving linear equation systems. For instance, Krylov subspace methods, such as Conjugate Gradients or GMRES, minimize residual errors that are orthogonal to each other (Stoer and Burlirsch, 2002).

In machine learning, inner products are important in the context of kernel methods (Schölkopf and Smola, 2002). Kernel methods exploit the fact that many linear algorithms can be expressed purely by inner product computations. Then, the "kernel trick" allows us to compute these inner products implicitly in a (potentially infinite-dimensional) feature space, without even knowing this feature space explicitly. This allowed the "non-linearization" of many algorithms used in machine learning, such as kernel-PCA (Schölkopf et al., 1997) for dimensionality reduction. Gaussian processes (Rasmussen and Williams, 2006) also fall into the category of kernel methods and are the current state-of-the-art in probabilistic regression (fitting curves to data points). The idea of kernels is explored further in Chapter 12.

Projections are often used in computer graphics, e.g., to generate shadows. In optimization, orthogonal projections are often used to (iteratively) minimize residual errors. This also has applications in machine learning, e.g., in linear regression where we want to find a (linear) function that minimizes the residual errors, i.e., the lengths of the orthogonal projections of the data onto the linear function (Bishop, 2006). We will investigate this further in Chapter 9. PCA (Hotelling, 1933; Pearson, 1901b) also uses projections to reduce the dimensionality of high-dimensional data. We will discuss this in more detail in Chapter 10.

Exercises

3.1 Show that $\langle \cdot, \cdot \rangle$ defined for all $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{y} = (y_1, y_2)$ in $\mathbb{R}^2$ by:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := x_1 y_1 - (x_1 y_2 + x_2 y_1) + 2(x_2 y_2)$$

is an inner product.

3.2 Consider $\mathbb{R}^2$ with $\langle \cdot, \cdot \rangle$ defined for all $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^2$ as:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x}^\top \underbrace{\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix}}_{=:\boldsymbol{A}} \boldsymbol{y}$$

Is $\langle \cdot, \cdot \rangle$ an inner product?

3.3 Consider the Euclidean vector space $\mathbb{R}^5$ with the dot product. A subspace $U \subseteq \mathbb{R}^5$ and $\boldsymbol{x} \in \mathbb{R}^5$ are given by

$$U = \operatorname{span}[\begin{bmatrix} 0 \\ -1 \\ 2 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -3 \\ 1 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} -3 \\ 4 \\ 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -3 \\ 5 \\ 0 \\ 7 \end{bmatrix}], \quad \boldsymbol{x} = \begin{bmatrix} -1 \\ -9 \\ -1 \\ 4 \\ 1 \end{bmatrix}$$

1. Determine the orthogonal projection $\pi_U(\boldsymbol{x})$ of $\boldsymbol{x}$ onto U
2. Determine the distance $d(\boldsymbol{x}, U)$

3.4 Consider $\mathbb{R}^3$ with the inner product

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x}^\top \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \boldsymbol{y} .$$

Furthermore, we define $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ as the standard/canonical basis in $\mathbb{R}^3$.

1. Determine the orthogonal projection $\pi_U(\boldsymbol{e}_2)$ of $\boldsymbol{e}_2$ onto

$$U = \operatorname{span}[\boldsymbol{e}_1, \boldsymbol{e}_3] .$$

   Hint: Orthogonality is defined through the inner product.
2. Compute the distance $d(\boldsymbol{e}_2, U)$.
3. Draw the scenario: standard basis vectors and $\pi_U(\boldsymbol{e}_2)$

3.5 Prove the Cauchy-Schwarz inequality $| \langle \boldsymbol{x}, \boldsymbol{y} \rangle | \leqslant \|\boldsymbol{x}\| \, \|\boldsymbol{y}\|$ for $\boldsymbol{x}, \boldsymbol{y} \in V$, where V is a vector space.

4

Matrix Decompositions

In Chapters 2 and 3, we studied ways to manipulate and measure vectors, projections of vectors and linear mappings. Mappings and transformations of vectors can be conveniently described as operations performed on matrices. Moreover, data is often represented in matrix form as well, for example where the rows of the matrix represent different instances of the data (for example people) and the columns describe different features of the data (for example weight, height and socio-economic status). In this chapter we present three aspects of matrices: how to summarize matrices, how matrices can be decomposed, and how these decompositions can be used to consider matrix approximations.

We first consider methods that allow us to describe matrices with just a few numbers that characterize the overall properties of matrices. We will do this in the sections on determinants (Section 4.1) and eigenvalues (Section 4.2 for the important special case of square matrices. These characteristic numbers have important mathematical consequences and allow us to quickly grasp what useful properties a matrix has. From here we will proceed to matrix decomposition methods: An analogy for matrix decomposition is the factoring of numbers, such as the factoring of 21 into prime numbers 7×3. For this reason matrix decomposition is also often referred to as *matrix factorization*. Matrix decompositions are used to interpret a matrix using a different representation using factors of interpretable matrices.

matrix factorization

We will first cover a square-root-like operation for matrices called Cholesky decomposition (Section 4.3) for symmetric, positive definite matrices. From here we will look at two related methods for factorizing matrices into canonical forms. The first one is known as matrix diagonalization (Section 4.4), which allows us to represent the linear mapping using a diagonal transformation matrix if we choose an appropriate basis. The second method, singular value decomposition (Section 4.5), extends this factorization to non-square matrices, and it is considered one of the fundamental concepts in linear algebra. These decomposition are helpful as matrices representing numerical data are often very large and hard to analyze. We conclude the chapter with a systematic overview of the types of matrices and the characteristic properties that distinguish them in form of a matrix taxonomy (Section 4.7).

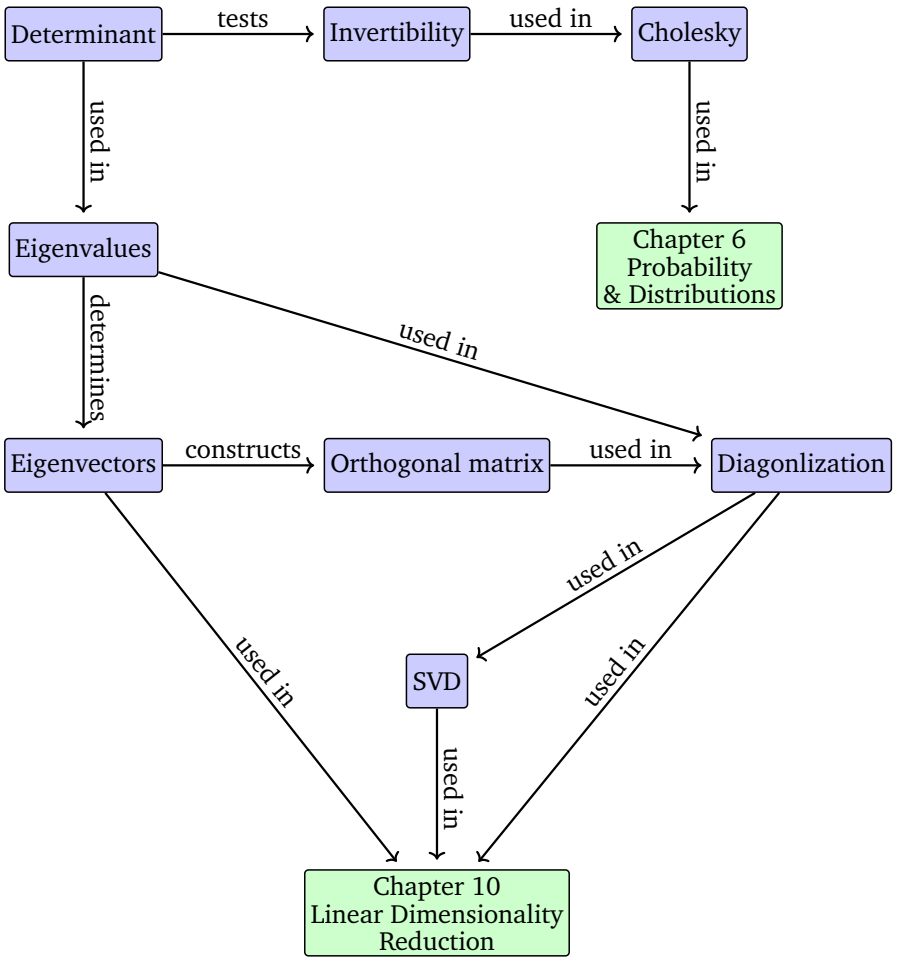


Figure 4.1 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

The methods that we cover in this chapter will become important in both subsequent mathematical chapters, such as Chapter 6 but also in applied chapters, such as dimensionality reduction in Chapters 10 or density estimation in Chapter 11. This chapter's overall structure is depicted in the mind map of Figure 4.1.

4.1 Determinant and Trace

Determinants are important concepts in linear algebra. A determinant is a mathematical object in the analysis and solution of systems of linear equations. Determinants are only defined for square matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, that is matrices with the same number of rows and columns. In this book we write this as $\det(\boldsymbol{A})$ (some textbooks may use $|\boldsymbol{A}|$, which we find confusing in terms of notation with the absolute value). However, we will use the straight lines when we write out the full matrix. Recall that a_{ij} be

the element in the i^{th} row and j^{th} column of a matrix $\boldsymbol{A}$. Then we write

$$\det(\boldsymbol{A}) = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} . \tag{4.1}$$

determinant

The *determinant* of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a function that maps $\boldsymbol{A}$ onto a real number. Before provide a definition of the determinant for general $n \times n$ matrices let us look at some motivating examples, and define determinants for some special matrices.

Example 4.1 (Testing for Matrix Invertibility)
Let us begin with exploring if a square matrix $\boldsymbol{A}$ is invertible (see Section 2.2.2). For the smallest cases, we already know when a matrix is invertible. If $\boldsymbol{A}$ is a 1×1 matrix, i.e., it is a scalar number, then $\boldsymbol{A} = a \implies \boldsymbol{A}^{-1} = \frac{1}{a}$. Thus $a\,\frac{1}{a} = 1$ holds, if and only if $a \neq 0$.

For the case of 2×2 matrices, by the definition of the inverse (Definition 2.3), we know that $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$ and thus we can write that the inverse of $\boldsymbol{A}^{-1}$ is (from Equation 2.23)

$$\boldsymbol{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} . \tag{4.2}$$

Thus, $\boldsymbol{A}$ is invertible if and only if

$$a_{11}a_{22} - a_{12}a_{21} \neq 0 \,. \tag{4.3}$$

This quantity is the determinant of $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$, that is

$$\det(\boldsymbol{A}) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21} \,. \tag{4.4}$$

The example above points already at the relationship between determinants and the existence of inverse matrices. The next theorem states the same result for $n \times n$ matrices.

Theorem 4.1. *For any square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ it holds that $\boldsymbol{A}$ is invertible if and only if $\det(\boldsymbol{A}) \neq 0$.*

We have explicit (closed form) expressions for determinants of small matrices in terms of the elements of the matrix. For $n = 1$,

$$\det(\boldsymbol{A}) = \det(a_{11}) = a_{11} \,. \tag{4.5}$$

For $n = 2$,

$$\det(\boldsymbol{A}) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21} \,, \tag{4.6}$$

which we have observed in the example above. For $n = 3$ (known as Sarrus' rule),

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23} - a_{31}a_{22}a_{13} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33} \,. \tag{4.7}$$

For a memory aid of the product terms in Sarrus' rule, try tracing the elements of the triple products in the matrix. We call a square matrix $\boldsymbol{A}$ a *upper triangular matrix* if $a_{ij} = 0$ for $i > j$, that is the matrix is zero below its diagonal. Analogously, we define a *lower triangular matrix* as a matrix with zeros above its diagonal. For an upper/lower triangular matrix $\boldsymbol{A}$, the determinant is the product of the diagonal elements:

upper triangular matrix
lower triangular matrix

$$\det(\boldsymbol{A}) = \prod_{i=1}^{n} a_{ii}. \tag{4.8}$$

The determinant is the signed volume of the parallelepiped formed by the columns of the matrix.

Example 4.2 (Determinants as Measures of Volume)
The notion of a determinant is natural when we consider it as a mapping from a set of n vectors spanning an object in $\mathbb{R}^n$. It turns out that the determinant is then the signed volume of an n-dimensional parallelepiped formed by columns of a matrix $\boldsymbol{A}$.

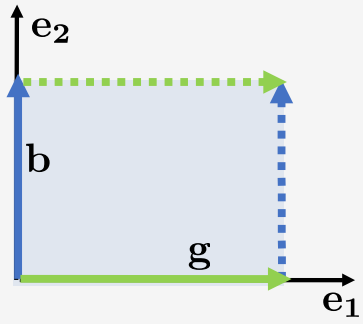


Figure 4.2 Determinants can measure areas spanned by vectors. The area A of the parallelogram (shaded region) spanned by the vectors $\boldsymbol{b}$ and $\boldsymbol{g}$ is given by the determinant $\det([\boldsymbol{b}, \boldsymbol{g}])$.

For $n = 2$ the columns of the matrix form a parallelogram. As the angle between vectors gets smaller the area of a parallelogram shrinks, too. Figure 4.2 illustrates this setting. Assume two linearly independent vectors $\boldsymbol{b}, \boldsymbol{g}$ that form the columns of a matrix $\boldsymbol{A} = [\boldsymbol{b}, \boldsymbol{g}]$. Then, the absolute value of the determinant of $\boldsymbol{A}$ is the area of the parallelogram with vertices $\boldsymbol{0}, \boldsymbol{b}, \boldsymbol{g}, \boldsymbol{b} + \boldsymbol{g}$. In particular, if the two vectors $\boldsymbol{b}, \boldsymbol{g}$ were linearly dependent so that $\boldsymbol{b} = \lambda \boldsymbol{g}$ for some $\lambda \in \mathbb{R}$ they no longer form a two-dimensional parallelogram. Therefore, the corresponding area is 0. On the contrary, if $\boldsymbol{b}, \boldsymbol{g}$ were al and lie along the canonical coordinate axes $\boldsymbol{e}_1, \boldsymbol{e}_2$ then they

would reduce to $\boldsymbol{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}$ and $\boldsymbol{g} = \begin{bmatrix} 0 \\ g \end{bmatrix}$ and the determinant

$$\begin{vmatrix} b & 0 \\ 0 & g \end{vmatrix} = bg - 0 = bg \tag{4.9}$$

becomes the familiar formula: area = height × length.

The sign of the determinant measures the orientation of the spanning vectors $\boldsymbol{b}, \boldsymbol{g}$ with respect to the standard coordinate system $\boldsymbol{e}_1, \boldsymbol{e}_2$. In our figure, flipping the spanning order to $\boldsymbol{g}, \boldsymbol{b}$ swaps the columns of $\boldsymbol{A}$ and reverses the orientation of the shaded surface A.

This intuition extends to higher dimensions. In $\mathbb{R}^3$, we consider three vectors $\boldsymbol{r}, \boldsymbol{b}, \boldsymbol{g} \in \mathbb{R}^3$ spanning the edges of a parallelepiped, i.e., a solid with faces that are parallel parallelograms (see Figure 4.3). The absolute value of the determinant of the 3×3 matrix $[\boldsymbol{r},\ \boldsymbol{b},\ \boldsymbol{g}]$ is the volume of the solid. Thus, the determinant acts as a function that measures the signed volume formed by column vectors composed in a matrix.

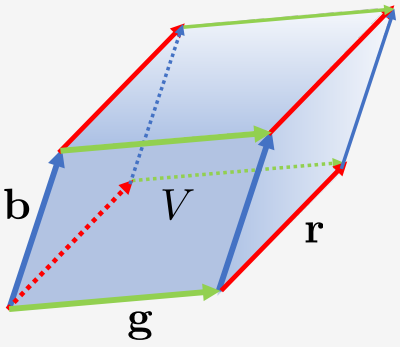


Figure 4.3 Determinants can measure volumes spanned by vectors. The volume of the parallelepiped (shaded volume) spanned by vectors $\boldsymbol{r}, \boldsymbol{b}, \boldsymbol{g}$ is given by the determinant $\det([\boldsymbol{r},\ \boldsymbol{b},\ \boldsymbol{g}])$.

Consider the three linearly independent vectors $\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{b} \in \mathbb{R}^3$ given as

$$\boldsymbol{r} = \begin{bmatrix} 2 \\ 0 \\ -8 \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} 6 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 4 \\ -1 \end{bmatrix}. \tag{4.10}$$

$$\boldsymbol{A} = [\boldsymbol{r},\ \boldsymbol{g},\ \boldsymbol{b}] = \begin{bmatrix} 2 & 6 & 1 \\ 0 & 1 & 4 \\ -8 & 0 & -1 \end{bmatrix}. \tag{4.11}$$

Therefore, the volume is given as

$$V = |\det(\boldsymbol{A})| = 186\,. \tag{4.12}$$

Computing the determinant of an $n \times n$ matrix requires a general algorithm to solve the cases for $n > 3$, which we are going to explore in the following. The theorem below reduces the problem of computing the deter-

minant of an $n\times n$ matrix to computing the determinant of $(n-1)\times(n-1)$ matrices. By recursively applying the Laplace expansion we can therefore compute determinants of $n \times n$ matrices by ultimately computing determinants of 2×2 matrices.

Theorem 4.2 (Laplace Expansion)**.** *Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Then, for all $j = 1, \dots, n$:*

1. *Expansion along column j*

$$\det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k+j} a_{kj} \det(\boldsymbol{A}_{k,j}) \,. \tag{4.13}$$

$\det(\boldsymbol{A}_{k,j})$ is called a *minor* and $(-1)^{k+j}\det(\boldsymbol{A}_{k,j})$ a *cofactor*.

2. *Expansion along row j*

$$\det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k+j} a_{jk} \det(\boldsymbol{A}_{j,k}) \,. \tag{4.14}$$

Here $\boldsymbol{A}_{k,j} \in \mathbb{R}^{(n-1)\times(n-1)}$ is the submatrix of $\boldsymbol{A}$ that we obtain when deleting row k and column j.

Example 4.3 (Laplace Expansion)
Let us compute the determinant of

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

using the Laplace expansion along the first row. By applying (4.14) we obtain

$$\begin{vmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 0 & 0 & 1 \end{vmatrix} = (-1)^{1+1} \cdot 1 \begin{vmatrix} 1 & 2 \\ 0 & 1 \end{vmatrix} + (-1)^{1+2} \cdot 2 \begin{vmatrix} 3 & 2 \\ 0 & 1 \end{vmatrix} + (-1)^{1+3} \cdot 3 \begin{vmatrix} 3 & 1 \\ 0 & 0 \end{vmatrix} . \tag{4.16}$$

Then we can use (4.6) to compute the determinants of all 2×2 matrices and obtain.

$$\det(\boldsymbol{A}) = 1(1-0) - 2(3-0) + 3(0-0) = -5 \,.$$

For completeness we can compare this result to computing the determinant using Sarrus' rule (4.7):

$$\det(\boldsymbol{A}) = 1\cdot1\cdot1+3\cdot0\cdot3+0\cdot2\cdot2-0\cdot1\cdot3-1\cdot0\cdot2-3\cdot2\cdot1 = 1-6 = -5 \,. \tag{4.17}$$

For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ the determinant exhibits the following properties:

- The determinant of a product is the product of the determinant, $\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$.

- Determinants are invariant to transposition $\det(\boldsymbol{A}) = \det(\boldsymbol{A}^\top)$.
- If $\boldsymbol{A}$ is regular (Section 2.2.2) then $\det(\boldsymbol{A}^{-1}) = \frac{1}{\det(\boldsymbol{A})}$
- Similar matrices (Defintion 2.21) possess the same determinant. Therefore, for a linear mapping $\Phi : V \to V$ all transformation matrices $\boldsymbol{A}_\Phi$ of Φ have the same determinant. Thus, the determinant is invariant to the choice of basis of a linear mapping.
- Adding a multiple of a column/row to another one does not change $\det(\boldsymbol{A})$.
- Multiplication of a column/row with $\lambda \in \mathbb{R}$ scales $\det(\boldsymbol{A})$ by λ. In particular, $\det(\lambda\boldsymbol{A}) = \lambda^n \det(\boldsymbol{A})$.
- Swapping two rows/columns changes the sign of $\det(\boldsymbol{A})$.

Because of the last three properties, we can use Gaussian elimination (see Section 2.1) to compute $\det(\boldsymbol{A})$ by bringing $\boldsymbol{A}$ into row-echelon form. We can stop Gaussian elimination when we have $\boldsymbol{A}$ in a triangular form where the elements below the diagonal are all 0. Recall from Equation (4.8) that the determinant is then the product of the diagonal elements.

Theorem 4.3. *A square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ has $\det(\boldsymbol{A}) \neq 0$ if and only if $\mathrm{rk}\boldsymbol{A} = n$. In other words a square matrix is invertible if and only if it is full rank.*

When mathematics was mainly performed by hand, the determinant calculation was considered an essential way to analyze matrix invertibility. However, contemporary approaches in machine learning use direct numerical methods that superseded the explicit calculation of the determinant. For example, in Chapter 2 we learned that inverse matrices can be computed by Gaussian elimination. Gaussian elimination can thus be used to compute the determinant of a matrix.

Determinants will play an important theoretical role for the following sections, especially when we learn about eigenvalues and eigenvectors (Section 4.2) through the characteristic polynomial of a matrix.

trace

Definition 4.4. The *trace* of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a linear function denoted by $\mathrm{tr}(\boldsymbol{A})$ and defined as

$$\mathrm{tr}(\boldsymbol{A}) := \sum_{i=1}^{n} a_{ii}\,, \tag{4.18}$$

in other words, the trace is the sum of the diagonal elements of $\boldsymbol{A}$.

Remark. For $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ the trace satisfies the following properties:

1. $\mathrm{tr}(\boldsymbol{A} + \boldsymbol{B}) = \mathrm{tr}(\boldsymbol{A}) + \mathrm{tr}(\boldsymbol{B})$
2. $\mathrm{tr}(\alpha\boldsymbol{A}) = \alpha\mathrm{tr}(\boldsymbol{A})\,, \quad \alpha \in \mathbb{R}$
3. $\mathrm{tr}(\boldsymbol{I}_n) = n$
4. $\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}) = \mathrm{tr}(\boldsymbol{B}\boldsymbol{A})$

It can be shown that only one function satisfies these four properties together – the trace (Gohberg et al., 2012). ♢

Remark. The properties of the trace of matrix products are more general:

- The trace is invariant under *cyclic permutations*, i.e., cyclic permutations

$$\text{tr}(\boldsymbol{AKL}) = \text{tr}(\boldsymbol{KLA}) \tag{4.19}$$

  for matrices $\boldsymbol{A} \in \mathbb{R}^{a\times k}, \boldsymbol{K} \in \mathbb{R}^{l\times l}, \boldsymbol{L} \in \mathbb{R}^{l\times a}$. This property generalizes to products of arbitrarily many matrices.
- As a special case of (4.19) it follows that the trace is invariant under permutations of two non-square matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$:

$$\text{tr}(\boldsymbol{AB}) = \text{tr}(\boldsymbol{BA})\,. \tag{4.20}$$

  In particular, this means that for two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$

$$\text{tr}(\boldsymbol{xy}^\top) = \text{tr}(\boldsymbol{y}^\top\boldsymbol{x}) = \boldsymbol{y}^\top\boldsymbol{x} \in \mathbb{R}\,. \tag{4.21}$$

♢

Remark. Given some linear map $\Phi : V \to V$, we define the trace of this map by considering the trace of matrix representation of ϕ. We need to choose a basis for V and describe Φ as a matrix $\boldsymbol{A}$ relative to this basis, and taking the trace of this square matrix. Assume that $\boldsymbol{B}$ is transformation matrix between bases of V. Then, we can write

$$\text{tr}(\boldsymbol{BAB}^{-1}) = \text{tr}(\boldsymbol{B}^{-1}\boldsymbol{BA}) = \text{tr}(\boldsymbol{IA}) = \text{tr}(\boldsymbol{A})\,. \tag{4.22}$$

Thus, while matrix representations of linear mappings are basis dependent its trace is independent of the basis. ♢

The trace is useful in certain classes of machine learning models where data is fitted using linear regression models. The trace captures model complexity in these models and can be used to compare between models (a more principled foundation for model comparison is discussed in detail in Section 8.5).

In this section, we covered determinants and traces as functions characterizing a square matrix. Taking together our understanding of determinants and traces we can now define an important equation describing a matrix $\boldsymbol{A}$ in terms of a polynomial, which we will use extensively in the following sections.

Definition 4.5 (Characteristic Polynomial)**.** For $\lambda \in \mathbb{R}$ and a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$

$$p_{\boldsymbol{A}}(\lambda) = \det(\boldsymbol{A} - \lambda\boldsymbol{I}) \tag{4.23}$$

$$= c_0 + c_1\lambda + c_2\lambda^2 + \cdots + c_{n-1}\lambda^{n-1} + (-1)^n\lambda^n\,, \tag{4.24}$$

$c_0, \ldots, c_{n-1} \in \mathbb{R}$, is the *characteristic polynomial* of $\boldsymbol{A}$. In particular, characteristic polynomial

$$c_0 = \det(\boldsymbol{A})\,, \tag{4.25}$$

$$c_{n-1} = (-1)^{n-1}\text{tr}(\boldsymbol{A})\,. \tag{4.26}$$

The characteristic polynomial will allow us to compute eigenvalues and eigenvectors, covered in the next section.

4.2 Eigenvalues and Eigenvectors

We will now get to know a new way to characterize a matrix and, its associated linear mapping. Let us recall from Section 2.7.1 that every linear mapping has a unique transformation matrix given an ordered basis. We can interpret linear mappings and their associated transformation matrices by performing an "Eigen" analysis. *Eigen* is a German word meaning "characteristic", "self" or "own". As we will see the eigenvalues of a linear mapping will tell us how a special set of vectors, the eigenvectors, are transformed by the linear mapping.

Definition 4.6. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a square matrix. Then $\lambda \in \mathbb{R}$ is an *eigenvalue* of $\boldsymbol{A}$ and a nonzero $\boldsymbol{x} \in \mathbb{R}^n$ is the corresponding *eigenvector* of $\boldsymbol{A}$ if

eigenvalue
eigenvector

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}\,. \tag{4.27}$$

We call this the *eigenvalue equation*.

eigenvalue equation

Remark. In linear algebra literature and software, it is a often a convention that eigenvalues are sorted in descending order, so that the largest eigenvalue and associated eigenvector are called the first eigenvalue and its associated eigenvector, and the second largest called the second eigenvalue and its associated eigenvector, and so on. However textbooks and publications may have different or no notion of orderings. We do not want to presume an ordering in our book. ♢

Definition 4.7 (Collinearity & Codirection)**.** Two vectors that point in the same direction are called *codirected*. Two vectors are *collinear* if they point in the same or the opposite direction.

codirected
collinear

Remark (Non-uniqueness of Eigenvectors). If $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$ associated with eigenvalue λ then for any $c \in \mathbb{R}\setminus\{0\}$ it holds that $c\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$ with the same eigenvalue since

$$\boldsymbol{A}(c\boldsymbol{x}) = c\boldsymbol{A}\boldsymbol{x} = c\lambda\boldsymbol{x} = \lambda(c\boldsymbol{x})\,. \tag{4.28}$$

Thus, all vectors that are collinear to $\boldsymbol{x}$ are also eigenvectors of $\boldsymbol{A}$.

♢

Theorem 4.8. *$\lambda \in \mathbb{R}$ is eigenvalue of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ if and only if λ is a root of the characteristic polynomial $p_{\boldsymbol{A}}(\lambda)$ of $\boldsymbol{A}$.*

Definition 4.9 (Eigenspace and Eigenspectrum)**.** For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ the set of all eigenvectors of $\boldsymbol{A}$ associated with an eigenvalue λ spans a subspace of $\mathbb{R}^n$, which is called *eigenspace* of $\boldsymbol{A}$ with respect to λ and is denoted

eigenspace

enspectrum 2058 by E_λ. The set of all eigenvalues of $\boldsymbol{A}$ is called the *eigenspectrum*, or just
2059 spectrum, of $\boldsymbol{A}$.

2060 There are a number of ways to think about these characteristics

2061 • The eigenvector is a special vector that, left multiplying with the matrix
2062 $\boldsymbol{A}$ merely stretches the vector by a factor – the eigenvalue.
• Recall the definition of the kernel from Section 2.7.3, it follows that $E_\lambda = \ker(\boldsymbol{A} - \lambda\boldsymbol{I})$ since

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \iff \boldsymbol{A}\boldsymbol{x} - \lambda\boldsymbol{x} = \boldsymbol{0} \tag{4.29}$$

$$\iff (\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = \boldsymbol{0} \iff \boldsymbol{x} \in \ker(\boldsymbol{A} - \lambda\boldsymbol{I}). \tag{4.30}$$

2063 • Similar matrices (see Definition 2.21) possess the same eigenvalues.
2064 Therefore, a linear mapping Φ has eigenvalues that are independent
2065 from the choice of basis of its transformation matrix. This makes eigen-
2066 values, together with the determinant and the trace, the key character-
2067 istic parameters of a linear mapping as they are all invariant under basis
2068 change.

Example 4.4 (Eigenvalues, Eigenvectors and Eigenspaces)
Here is an example of how to find the eigenvalues and eigenvectors of a 2×2 matrix.

$$\boldsymbol{A} = \begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix}. \tag{4.31}$$

Step 1: Characteristic Polynomial

From our definition of the eigenvector $\boldsymbol{x}$ and eigenvalue λ for $\boldsymbol{A}$ there will be a vector such that $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$, i.e., $(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = \boldsymbol{0}$. Since $\boldsymbol{x} \neq \boldsymbol{0}$ by definition of the eigenvectors, this condition requires that the kernel (nullspace) of $\boldsymbol{A} - \lambda\boldsymbol{I}$ contains more elements than just $\boldsymbol{0}$. This means that $\boldsymbol{A} - \lambda\boldsymbol{I}$ is not invertible and therefore $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0$. Hence we need to compute the roots of the characteristic polynomial (Equation (4.23)).

Step 2: Eigenvalues

The characteristic polynomial is given as

$$p_{\boldsymbol{A}}(\lambda) = \det(\boldsymbol{A} - \lambda\boldsymbol{I}) = \det\left(\begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = \begin{vmatrix} 4-\lambda & 2 \\ 1 & 3-\lambda \end{vmatrix} \tag{4.32}$$

$$= (4-\lambda)(3-\lambda) - 2 \cdot 1\,. \tag{4.33}$$

We factorize the characteristic polynomial

$$p(\lambda) = (4-\lambda)(3-\lambda) - 2 \cdot 1 = 10 - 7\lambda + \lambda^2 = (2-\lambda)(5-\lambda) \tag{4.34}$$

and obtain the roots $\lambda_1 = 2$ and $\lambda_2 = 5$.

Step 3: Eigenvectors and Eigenspaces

We find the eigenvectors that correspond to these eigenvalues by looking at vectors $\boldsymbol{x}$ such that

$$\begin{bmatrix} 4-\lambda & 2 \\ 1 & 3-\lambda \end{bmatrix} \boldsymbol{x} = \mathbf{0}\,. \tag{4.35}$$

For $\lambda = 5$ we obtain

$$\begin{bmatrix} 4-5 & 2 \\ 1 & 3-5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{0}\,. \tag{4.36}$$

We now solve this homogeneous equation system and obtain a solution space

$$E_5 = \operatorname{span}[\begin{bmatrix} 2 \\ 1 \end{bmatrix}]\,, \tag{4.37}$$

where for $c \neq 0$ all vectors $c[2,1]^\top$ are eigenvectors for $\lambda = 5$. Note, that this eigenspace is one-dimensional (spanned by a single vector) but that in other cases where we have multiple eigenvalues (see Definition 4.13) the eigenspace may have more than one dimension.

Analogously, we find the eigenvector for $\lambda = 2$ by solving the homogeneous equation system

$$\begin{bmatrix} 4-2 & 2 \\ 1 & 3-2 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \boldsymbol{x} = \mathbf{0}\,. \tag{4.38}$$

This means any vector $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ where $x_2 = -x_1$, such as $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ is an eigenvector with eigenvalue 2. The corresponding eigenspace is given as

$$E_2 = \operatorname{span}[\begin{bmatrix} 1 \\ -1 \end{bmatrix}]\,. \tag{4.39}$$

Remark (Eigenvalues and Eigenspaces). If λ is an eigenvalue of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ then the corresponding eigenspace E_λ is the solution space of the homogeneous linear equation system $(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = \mathbf{0}$. Geometrically, the eigenvector corresponding to a nonzero eigenvalue points in a direction that is stretched by the linear mapping, and the eigenvalue is the factor by which it is stretched. If the eigenvalue is negative, the direction is of the stretching is flipped. In particular, the eigenvector does not change its direction under $\boldsymbol{A}$. ♢

Remark. The following statements are equivalent:

- λ is eigenvalue of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$
- There exists an $\boldsymbol{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ or equivalently, $(\boldsymbol{A} - \lambda\boldsymbol{I}_n)\boldsymbol{x} = \mathbf{0}$ can be solved non-trivially, i.e., $\boldsymbol{x} \neq \mathbf{0}$.

- $\mathrm{rk}(\boldsymbol{A} - \lambda \boldsymbol{I}_n) < n$
- $\det(\boldsymbol{A} - \lambda \boldsymbol{I}_n) = 0$

♢

Useful properties regarding eigenvalues and eigenvectors of various matrix types include

- A matrix $\boldsymbol{A}$ and its transpose $\boldsymbol{A}^\top$ possess the same eigenvalues, but not necessarily the same eigenvectors.
- Symmetric matrices always have real-valued eigenvalues.
- Symmetric positive definite matrices always have positive, real eigenvalues.
- The eigenvectors of symmetric matrices are always orthogonal to each other.

Theorem 4.10. *Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ we can always obtain a $\boldsymbol{S}$ that is a symmetric positive semi-definite matrix by computing*

$$\boldsymbol{S} = \boldsymbol{A}^\top \boldsymbol{A}\,. \tag{4.40}$$

Understanding why this theorem holds is insightful for how we can use symmetrised matrices: Symmetry requires $\boldsymbol{S} = \boldsymbol{S}^\top$ and by inserting (4.40) we obtain $\boldsymbol{S} = \boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{A}^\top(\boldsymbol{A}^\top)^\top = (\boldsymbol{A}^\top\boldsymbol{A})^\top = \boldsymbol{S}^\top$. Moreover, positive semi-definiteness (Section 3.2.3) requires that $\boldsymbol{x}^\top\boldsymbol{S}\boldsymbol{x} \geqslant 0$ and inserting (4.40) we obtain $\boldsymbol{x}^\top\boldsymbol{S}\boldsymbol{x} = \boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{x}^\top\boldsymbol{A}^\top)(\boldsymbol{A}\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x})^\top(\boldsymbol{A}\boldsymbol{x}) \geqslant 0$, because the scalar product computes a sum of squares (which are themselves always positive or zero).

Theorem 4.11 (Hogben (2006))**.** *Consider a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with distinct eigenvalues $\lambda_1, \ldots, \lambda_n$ and corresponding eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Then the eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are linearly independent.*

The theorem states that eigenvectors belonging to different eigenvalues form a linearly independent set. For symmetric matrices we can state a stronger version of Theorem 4.11.

Theorem 4.12 (Meyer (2000))**.** *Any symmetric matrix $\boldsymbol{A} = \boldsymbol{A}^\top \in \mathbb{R}^{n\times n}$. has n independent eigenvectors that form an orthogonal basis for $\mathbb{R}^n$.*

Graphical Intuition in Two Dimensions

Let us gain some intuition for determinants, eigenvectors, eigenvalues and how linear maps affect space. Figure 4.4 depicts five transformation matrices and their impact on a square grid of points. The square grid of points are contained within a box of dimensions 2×2 with its centre at the origin.

- $\boldsymbol{A}_1 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 2 \end{bmatrix}$. The direction of the two eigenvectors correspond to the canonical basis vectors in $\mathbb{R}^2$, i.e. to two cardinal axes. The horizontal axis is compressed by factor $\frac{1}{2}$ (eigenvalue $\lambda_1 = \frac{1}{2}$) and the vertical axis

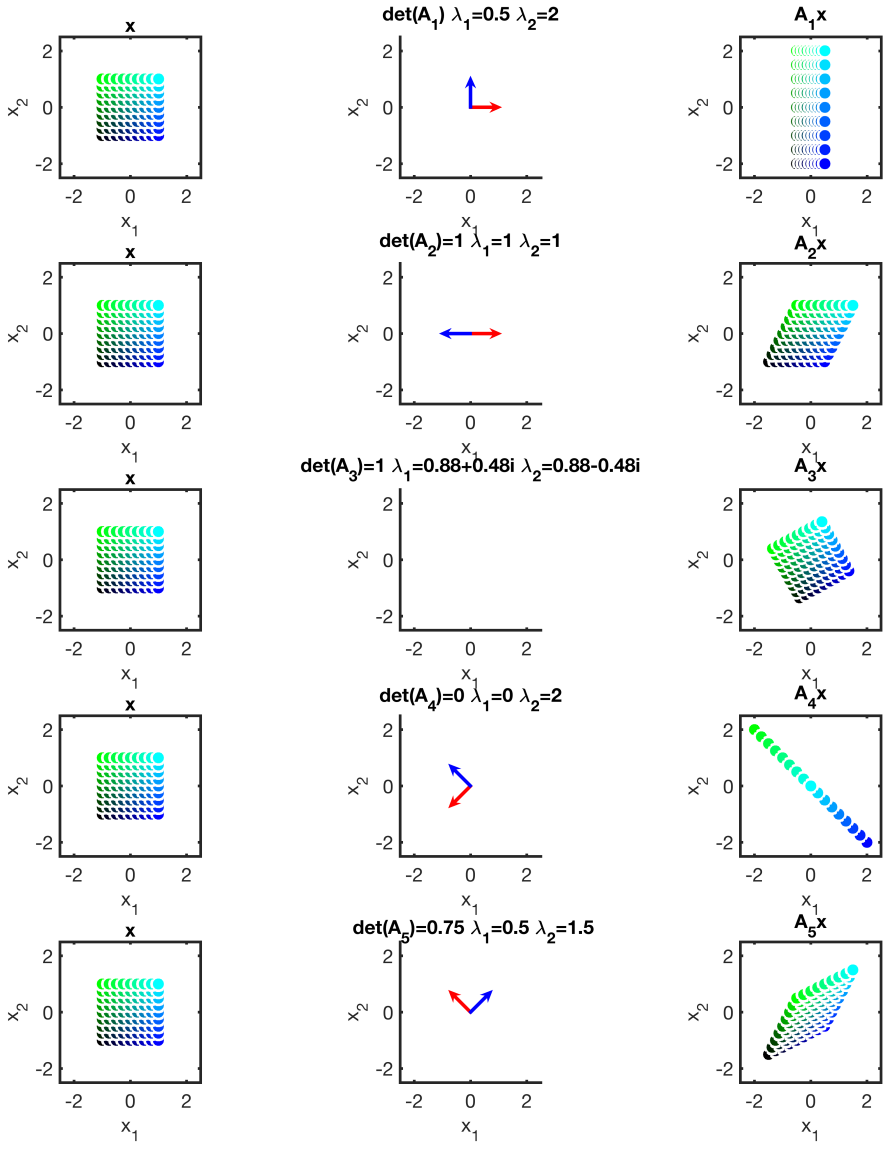


Figure 4.4 Determinants and eigenspaces. Overview of five linear mappings and their associated transformation matrices $\boldsymbol{A}_i \in \mathbb{R}^{2\times 2}$ project 81 color-coded points $\boldsymbol{x} \in \mathbb{R}^2$ (left column of plots) to target points $\boldsymbol{A}_i\boldsymbol{x}$ (right column of plots). The central column depicts the first eigenvector associated with eigenvalue λ_1, the second eigenvector associated with eigenvalue λ_2, as well as the value of the determinant. Each row depicts the effect of one of five transformation mappings in the standard basis $\boldsymbol{A}_i, i = \{1, \ldots, 5\}$.

is extended by a factor of 2 (eigenvalue $\lambda_2 = 2$). The mapping is area preserving ($\det(\boldsymbol{A}_1) = 1 = 2 \times \frac{1}{2}$). Note, that while the area covered by the box of points remained the same, the circumference around the box has increased by 20%.

- $\boldsymbol{A}_2 = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$ corresponds to a shearing mapping , i.e., it shears the points along the horizontal axis to the right if they are on the positive half of the vertical axis, and to the left vice versa. This mapping is area preserving ($\det(\boldsymbol{A}_2) = 1$). The eigenvalue $\lambda_1 = 1 = \lambda)2$ is repeated and the hence the eigenvectors are co-linear (drawn here for emphasis

in two opposite directions). This indicating that the mapping acts only along one direction (the horizontal axis). In geometry, the area preserving properties of this type of shearing parallel to an axis is also known as Cavalieri's principle of equal areas for parallelograms (Katz, 2004). Note, that the repeated identical eigenvalues make the two eigenvectors collinear, these are drawn in opposite directions to emphasize the shearing. Note, that the while the mapping is area preserving the circumference around the box of points has increased.

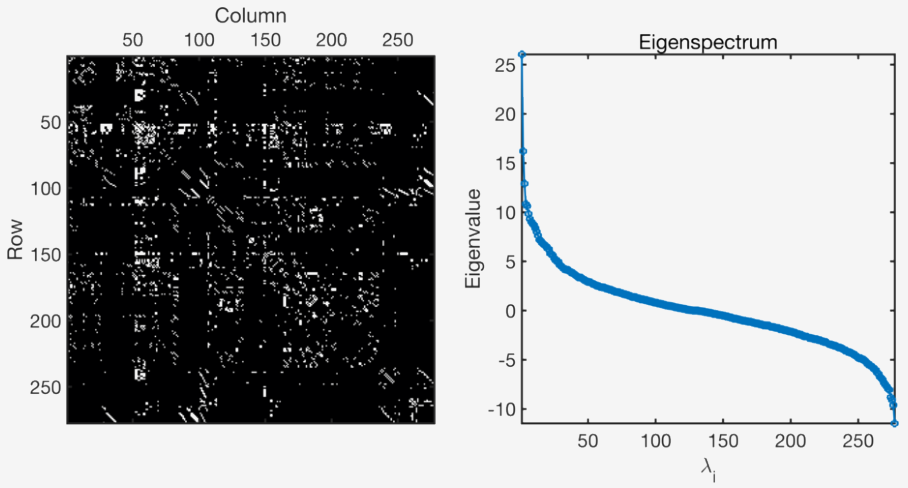
- $\boldsymbol{A}_3 = \begin{bmatrix} \cos(\frac{\pi}{6}) & -\sin(\frac{\pi}{6}) \\ \sin(\frac{\pi}{6}) & \cos(\frac{\pi}{6}) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \sqrt{3} & -1 \\ 1 & \sqrt{3} \end{bmatrix}$ The rotation matrix $\boldsymbol{A}_3$ rotates the points by $\frac{\pi}{6}$ (or 30° degrees) anti-clockwise, and has complex eigenvalues (reflecting that the mapping is a rotation) and no real valued eigenvalues (hence no eigenvectors are drawn). A pure rotation has to be area preserving, and hence the determinant is 1. Moreover, the circumference around the box of points has not changed. For more details on rotations we refer to Figure 3.14 in the corresponding section on rotations.

- $\boldsymbol{A}_4 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ reflects a mapping in the standard basis that collapses a two-dimensional domain onto a one-dimensional image space, hence the area is 0. We can see this because one eigenvalue is 0, collapsing the space in direction of the (red) eigenvector corresponding to $\lambda_1 = 0$, while the orthogonal (blue) eigenvector stretches space by a factor of $2 = \lambda_2$. Note, that while the area of the box of points vanishes the circumference does increase by around 41%.

- $\boldsymbol{A}_5 = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}$ is a shear-and-stretch mapping that shrinks space space by 75% ($|\det(\boldsymbol{A}_5)| = \frac{3}{4}$), stretching space along the (blue) eigenvector of λ_2 by 50% and compressing it along the orthogonal (red) eigenvector by a factor of 50%.

Example 4.5 (Eigenspectrum of a biological neural network)

Figure 4.5 Application of the eigenspectrum to characterize a biological neural network. See text for details.



Methods to analyze and learn from network data are an essential component of machine learning methods. Key to understanding networks is the connectivity between network nodes, especially if two nodes are connected to each other or not. In data science applications, it is often useful to study the matrix that captures this connectivity data. In Figure 4.5, we see the left plot showing the connectivity matrix (277×277), also referred to as adjacency matrix, of the complete neural network of the worm *C. Elegans*. Each row/column represents one of the 277 neurons of this worm's brain and the connectivity matrix $\boldsymbol{A}$ has a value of $a_{ij} = 1$ (white pixel) if neuron i talks to neuron j through a synapse, and $a_{ij} = 0$ (black pixel) otherwise. The neural network connectivity matrix is not symmetric, which implies that eigenvalues may not be real valued. Therefore we compute a version of the connectivity matrix as follows $\boldsymbol{A}_{sym} = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^\top)$. This new matrix $\boldsymbol{A}_{sym}$ has a value of 1 whenever two neurons are connected (irrespective of the direction of the connection) and zero otherwise. In the right panel, we show the eigenspectrum of $\boldsymbol{A}_{sym}$ in a scatter plot, on the horizontal axis we have the order of the eigenvalues from the largest (left most) to smallest eigenvalue and on the vertical axis the absolute of the eigenvalue. The S-like shape of this eigenspectrum is typical for many biological neural networks.

algebraic multiplicity

Definition 4.13. Let a square matrix $\boldsymbol{A}$ have an eigenvalue λ_i. The *algebraic multiplicity* of λ_i is the number of times the root appears in the characteristic polynomial.

geometric multiplicity

Definition 4.14. Let a square matrix $\boldsymbol{A}$ have an eigenvalue λ_i. The *geometric multiplicity* of λ_i is the total number of linearly independent eigenvectors associated with λ_i. In other words it is the dimensionality of the eigenspace spanned by the eigenvectors associated with λ_i.

Remark. A specific eigenvalue's geometric multiplicity must be at least one, as by definition every eigenvalue has at least one associated eigenvector. An eigenvalue's geometric multiplicity cannot exceed its algebraic multiplicity, but it may be lower. ♢

Example 4.6

The matrix $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$ has two repeated eigenvalues $\lambda_1 = \lambda_2 = 2$ and an algebraic multiplicity of 2. The eigenvalue has however only one distinct eigenvector $\boldsymbol{x}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and thus geometric multiplicity 1.

Before we conclude our considerations of eigenvalues and eigenvectors it is useful to tie these matrix characteristics together with the previously covered concept of the determinant and the trace.

Theorem 4.15. *The determinant of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the product of its eigenvalues, i.e.,*

$$\det(\boldsymbol{A}) = \prod_{i=1}^{n} \lambda_i \,, \tag{4.41}$$

where λ_i are (possibly repeated) eigenvalues of $\boldsymbol{A}$.

Theorem 4.16. *The trace of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the sum of its eigenvalues, i.e.,*

$$tr(\boldsymbol{A}) = \sum_{i=1}^{n} \lambda_i \,, \tag{4.42}$$

where λ_i are (possibly repeated) eigenvalues of $\boldsymbol{A}$.

While we leave these two theorems without a proof, we point to the application of the determinant and trace of the characteristic polynomial as a way to derive them.

Remark. A geometric intuition for these two theorems goes as follows (see also Figure 4.2 and corresponding text for other examples): Imagine a unit cube (a box with equal sides of length 1) in $\mathbb{R}^3$. We then map the 8 corner points of this box through our matrix $\boldsymbol{A}$ and obtain a new box, defined by the mapped 8 new corner points. We know that the eigenvalues capture the scaling of the basis with respect to the standard basis. Thus, they capture how the volume of the unit cube (which has volume 1) was transformed into our box. Thus, the determinant as product of eigenvalues is akin to the volume of the box, a large determinant suggests a large expansion of volume and vice versa. In contrast the trace is a sum of eigenvalues, i.e. a sum of length scales. Consider a gift ribbon we would want to tie around the box. The length of ribbon is proportional to the

length of the sides of the box. The trace of $\boldsymbol{A}$ captures therefore a notion of how the matrix acts on the circumference of a volume. ♢

Example 4.7 (Google's PageRank – Webpages as Eigenvectors)
Google uses the eigenvector corresponding to the maximal eigenvalue of a matrix $\boldsymbol{A}$ to determine the rank of a page for search. The idea that the PageRank algorithm, developed at Stanford University by Larry Page and Sergey Brin in 1996, came up up was that the importance of any web page can be judged by looking at the pages that link to it. For this, they write down all websites as a huge directed graph that shows which page links to which. PageRank computes the weight (importance) $x_i \geqslant 0$ of a website a_i by counting the number of pages pointing to a_i. PageRank also take the importance of the website into account that links to a website to a_i. Then, the navigation behavior of a user can be described by a transition matrix $\boldsymbol{A}$ of this graph that tells us with what (click) probability somebody will end up on a different website. The matrix $\boldsymbol{A}$ has the property that for any initial rank/importance vector $\boldsymbol{x}$ of a website the sequence $\boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}^2\boldsymbol{x}, \dots$ converges to a vector $\boldsymbol{x}^*$. This vector is called the *PageRank* and satisfies $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{x}^*$, i.e., it is an eigenvector (with corresponding eigenvalue 1) of $\boldsymbol{A}$. After normalizing by $\boldsymbol{x}^*$, such that $\|\boldsymbol{x}^*\| = 1$ we can interpret the entries as probabilities. More details and different perspectives on PageRank can be found in the original technical report (Page et al., 1999).

PageRank

4.3 Cholesky Decomposition

There are many ways to factorize special types of matrices that we encounter often in machine learning. In the positive real numbers we have the square-root operation that yields us a decomposition of the number into components, for example, $9 = 3 \cdot 3$. For matrices, we need to be careful that we compute a square-root like operation on positive quantities. For symmetric, positive definite matrices (see Section 3.2.3) we can choose from a number of square-root equivalent operations. The *Cholesky decomposition* or *Cholesky factorization* provides a square-root equivalent operations that is very useful.

Cholesky decomposition

Cholesky factorization

Theorem 4.17. *Cholesky Decomposition: A symmetric positive definite matrix $\boldsymbol{A}$ can be factorized into a product $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\top$, where $\boldsymbol{L}$ is a lower triangular matrix with positive diagonal elements:*

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & \cdots & l_{n1} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & l_{nn} \end{bmatrix}. \tag{4.43}$$

$\boldsymbol{L}$ is called the Cholesky factor of $\boldsymbol{A}$.

Cholesky factor

Example 4.8
It is not immediately apparent why the Cholesky decomposition should exist for any symmetric, positive definite matrix. While we omit the proof we can go through an 3×3 matrix example.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \equiv \boldsymbol{L}\boldsymbol{L}^\top = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} \tag{4.44}$$

Expanding the right hand side yields

$$\boldsymbol{A} = \begin{bmatrix} l_{11}^2 & l_{21}l_{11} & l_{31}l_{11} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{31}l_{21} + l_{32}l_{22} \\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix} .$$

Comparing the left hand side and the right hand side shows that there is a simple pattern in the diagonal elements (l_{ii}):

$$l_{11} = \sqrt{a_{11}}\,, \qquad l_{22} = \sqrt{a_{22} - l_{21}^2}\,, \qquad l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)}\,. \tag{4.45}$$

Similarly for the elements below the diagonal (l_{ij}, where $i > j$) there is also a repeating pattern:

$$l_{21} = \frac{1}{l_{11}} a_{21}\,, \qquad l_{31} = \frac{1}{l_{11}} a_{31}\,, \qquad l_{32} = \frac{1}{l_{22}}(a_{32} - l_{31}l_{21})\,. \tag{4.46}$$

Thus, we have now constructed the Cholesky decomposition for any semi-positive definite 3×3 matrix. The key realization is that we can backwards calculate what the components l_{ij} for the $\boldsymbol{L}$ should be, given the values a_{ij} for $\boldsymbol{A}$ and previously computed values of l_{ij}.

The Cholesky decomposition is an important tool for the numerical computations underlying machine learning. The Cholesky decomposition is used as a computationally more efficient and numerically more stable way to solve systems of equations that form symmetric positive definite matrices, than computing the inverse of such a matrix, and is thus used under the hood in numerical linear algebra packages.

For matrices that are symmetric positive definite such as the covariance of a multivariate Gaussian 6.6, one approach is to transform the matrix into a set of upper or lower triangular matrices. After applying the Cholesky decomposition we efficiently compute the inverse $\boldsymbol{L}^{-1}$ of a triangular matrix by back substitution. Then the original matrix inverse is computed simply by multiplying the two inverses as $\boldsymbol{A}^{-1} = (\boldsymbol{L}\boldsymbol{L}^\top)^{-1} = (\boldsymbol{L}^{-1})^\top(\boldsymbol{L}^{-1})$. As bonus, the determinant is also much easier to compute, because $\det(\boldsymbol{A}) = \det(\boldsymbol{L})^2$, and the determinant of the triangular

Cholesky factor $\boldsymbol{L}$ is the product of its diagonal elements so that $\det(\boldsymbol{A}) = \prod_i l_{ii}^2$.

4.4 Eigendecomposition and Diagonalization

Diagonal matrices are of the form

$$\boldsymbol{D} = \begin{bmatrix} c_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & c_n \end{bmatrix} \tag{4.47}$$

and possess a very simple structure. Therefore, they allow fast computation of determinants, powers and inverses. The determinant is the product of its diagonal entries, a matrix power $\boldsymbol{D}^k$ is given by each diagonal element raised to the power k, and the inverse $\boldsymbol{D}^{-1}$ is the reciprocal of its diagonal elements if all of them are non-zero.

In this section, we will look at how to transform matrices into diagonal form. This is an important application of the basis change we discussed in Section 2.7.2 and eigenvalues from Section 4.2.

Let us recall that two matrices $\boldsymbol{A}, \boldsymbol{D}$ are similar (Definition 2.21) if there exists an invertible matrix $\boldsymbol{P}$, such that $\boldsymbol{D} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$. More specifically, we will look at matrices $\boldsymbol{A}$ that are similar to a diagonal matrix $\boldsymbol{D}$ that contains the eigenvalues of $\boldsymbol{A}$ on its diagonal.

diagonalizable

Definition 4.18 (Diagonalizable)**.** A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *diagonalizable* if it is similar to a diagonal matrix, in other words there exists a matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ so that $\boldsymbol{D} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$.

In the following, we will see that diagonalizing a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a way of expressing the same linear mappping but in another basis (see Section 2.6.1). Specifically we will try to diagonalize a matrix $\boldsymbol{A}$ by finding a new basis that consists of the eigenvectors of $\boldsymbol{A}$. We present two theorems, first for square matrices (Theorem 4.19) then for symmetric matrices (Theorem 4.21). The following results parallels the discussion we had about eigenvalues and eigenvectors (Theorem 4.11 and Theorem 4.12).

We first explore how to compute $\boldsymbol{P}$ so as to diagonalize $\boldsymbol{A}$. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, let $\lambda_1, \ldots, \lambda_n$ be a set of scalars, and let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ be a set of vectors in $\mathbb{R}^n$. Then we set $\boldsymbol{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$ and let $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ be a diagonal matrix with diagonal entries $\lambda_1, \ldots, \lambda_n$. Then we can show that

$$\boldsymbol{A}\boldsymbol{P} = \boldsymbol{P}\boldsymbol{D} \tag{4.48}$$

if and only if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $\boldsymbol{A}$ and the $\boldsymbol{p}_i$ are the corresponding eigenvectors of $\boldsymbol{A}$.

We can see that this statement holds because

$$\boldsymbol{A}\boldsymbol{P} = \boldsymbol{A}[\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n] = [\boldsymbol{A}\boldsymbol{p}_1, \ldots, \boldsymbol{A}\boldsymbol{p}_n] \tag{4.49}$$

$$\boldsymbol{P}\boldsymbol{D} = [\boldsymbol{p}_1, \dots, \boldsymbol{p}_n] \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} = [\lambda_1 \boldsymbol{p}_1, \dots, \lambda_n \boldsymbol{p}_n] \quad . \tag{4.50}$$

Thus, (4.48) implies that

$$\boldsymbol{A}\boldsymbol{p}_1 = \lambda_1 \boldsymbol{p}_1 \tag{4.51}$$

$$\vdots$$

$$\boldsymbol{A}\boldsymbol{p}_n = \lambda_n \boldsymbol{p}_n \tag{4.52}$$

and vice versa.

Thus, the matrix $\boldsymbol{P}$ must be composed of columns of eigenvectors. But this is not sufficient to know if we can diagonalize $\boldsymbol{A}$, as our definition of diagonalization requires that $\boldsymbol{P}$ is invertible. From Theorem 4.3 we know that our square matrix $\boldsymbol{P}$ is only invertible (has determinant $\neq 0$) if it has full rank. This implies that the eigenvectors $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ must be linearly independent. Moreover, consider that Theorem 4.11 tells us when $\boldsymbol{A}$ is diagonalizable by having n independent eigenvectors, namely in only those cases where $\boldsymbol{A}$ has n distinct eigenvalues. Taking together these arguments we can now combine them to formulate a key theorem of this chapter.

Theorem 4.19. *Eigendecomposition/Diagonalization theorem. A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ can be factored as* Diagonalization

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1} \tag{4.53}$$

where $\boldsymbol{P}$ is an invertible matrix of eigenvectors and $\boldsymbol{D}$ is a diagonal matrix which diagonal entries are the eigenvalues of $\boldsymbol{A}$, if and only if $\boldsymbol{A}$ has n independent eigenvectors (i.e. $\mathrm{rk}(\boldsymbol{P}) = n$).

Definition 4.20. A *defective matrix* is a square matrix if it does not have a complete set of eigenvectors (i.e. n linearly independent eigenvectors or the sum of the dimensions of the eigenspaces is n) and is therefore not diagonalizable (see also Theorem 4.11). defective matrix

Remark.
- Any defective matrix must has fewer than n distinct eigenvalues because distinct eigenvalues have linearly independent eigenvectors. Specifically, a defective matrix has at least one eigenvalue λ with an algebraic multiplicity $m > 1$ and fewer than m linearly independent eigenvectors associated with λ.
- The *Jordan Normal Form* of a matrix offers a decomposition that works for defective matrices but is beyond the scope of this book (Lang, 1987). Jordan Normal Form

♢

For symmetric matrices we can obtain even stronger outcomes for the eigenvalue decomposition.

Theorem 4.21. *A symmetric matrix $\boldsymbol{S} = \boldsymbol{S}^\top \in \mathbb{R}^{n\times n}$ can always be diagonalized into*

$$\boldsymbol{S} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top \tag{4.54}$$

where $\boldsymbol{P}$ is matrix of n orthogonal eigenvectors and $\boldsymbol{D}$ is a diagonal matrix of its n eigenvalues.

Proof By Theorem 4.12 we know that $\boldsymbol{P} = [\boldsymbol{p}_1, \dots, \boldsymbol{p}_n]$ has n orthogonal eigenvectors of $\boldsymbol{S}$ with eigenvalues $\lambda_1, \dots, \lambda_n$. We can then write

$$(\boldsymbol{P}^\top \boldsymbol{P})_{ij} = \boldsymbol{p}_i^\top \boldsymbol{p}_j \tag{4.55}$$

where

$$\boldsymbol{p}_i^\top \boldsymbol{p}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{4.56}$$

and therefore $\boldsymbol{P}^\top \boldsymbol{P} = \boldsymbol{I}$ and $\boldsymbol{P}^{-1} = \boldsymbol{P}^\top$.

We observe the following product

$$\lambda_i \boldsymbol{P}\boldsymbol{p}_i = \lambda_i \left[\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\right] \boldsymbol{p}_i = \lambda_i \boldsymbol{e}_i, \tag{4.57}$$

which we will use in the following derivation.

$$\begin{aligned}
\boldsymbol{P}^\top \boldsymbol{S}\boldsymbol{P} &= \boldsymbol{P}^\top \boldsymbol{S} \left[\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\right] && (4.58)\\
&= \boldsymbol{P}^\top \left[\boldsymbol{S}\boldsymbol{p}_1, \dots, \boldsymbol{S}\boldsymbol{p}_n\right] && (4.59)\\
&= \boldsymbol{P}^\top \left[\lambda_1 \boldsymbol{p}_1, \dots, \lambda_n \boldsymbol{p}_n\right] && (4.60)\\
&= \left[\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\right]^\top \left[\lambda_1 \boldsymbol{p}_1, \dots, \lambda_n \boldsymbol{p}_n\right] && (4.61)\\
&= \left[\lambda_1 \boldsymbol{e}_1, \dots, \lambda_n \boldsymbol{e}_n\right] = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} = \boldsymbol{D} && (4.62)
\end{aligned}$$

□

Geometric intuition for the eigendecomposition

We can interpret the eigendecomposition of a matrix as follows (see also Figure 4.6): Let $\boldsymbol{A}$ be the transformation matrix of a linear mapping with respect to the standard basis. $\boldsymbol{P}^{-1}$ performs a basis change from the standard basis into the eigenbasis. This maps the eigenvectors $\boldsymbol{p}_i$ (red and green arrows in Figure 4.6) onto the standard axes $\boldsymbol{e}_i$. Then, the diagonal $\boldsymbol{D}$ scales thevectors along these axes by the eigenvalues $\lambda_i \boldsymbol{e}_i$ and, finally, $\boldsymbol{P}$ transforms these scaled vectors back into the standard/canonical coordinates (yielding $\lambda_i \boldsymbol{p}_i$).

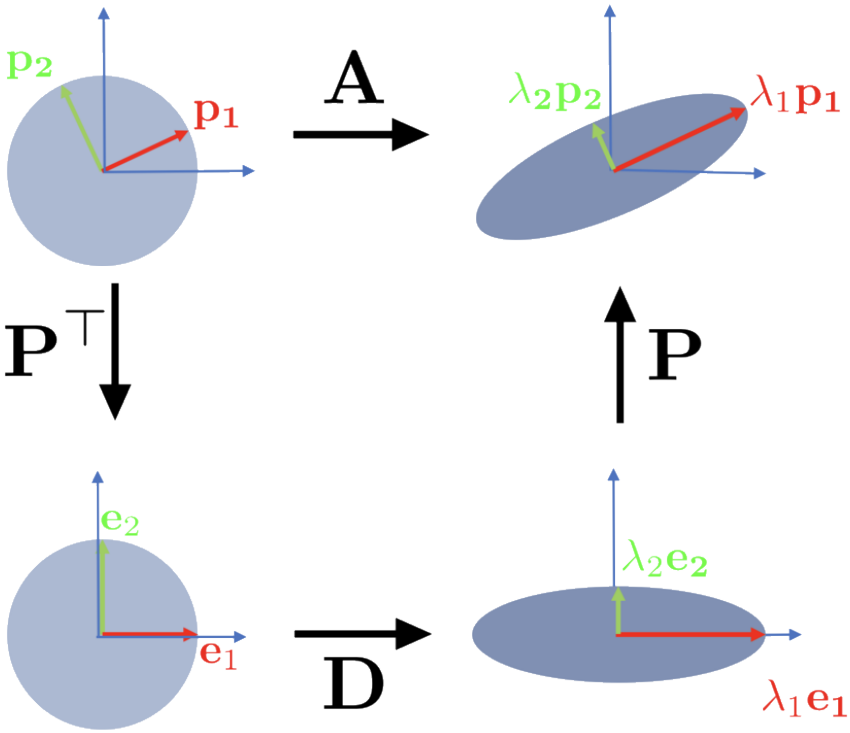


Figure 4.6 Intuition behind the eigendecomposition of a $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ in the standard basis as sequential transformations. Top-left to bottom-left: $\boldsymbol{P}^\top$ performs a basis change (here drawn in $\mathbb{R}^2$ and depicted as a rotation-like operation) mapping the eigenvectors into the standard basis. Bottom-left-to-bottom right $\boldsymbol{D}$ performs a scaling along the remapped orthogonal eigenvectors, depicted here by a circle being stretched to an ellipse. Bottom-left to top-left: $\boldsymbol{P}$ undoes the basis change (depicted as a reverse rotation) and restores the original coordinate frame.

Example 4.9

Let us compute the eigendecomposition of a (symmetric) matrix $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

Step 1: Compute the eigenvalues and eigenvectors

The matrix has eigenvalues

$$\begin{aligned} \det(\boldsymbol{A} - \lambda \boldsymbol{I}) &= \det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right) && (4.63) \\ &= (2-\lambda)^2 - 1 = \lambda^2 - 2\lambda + 3 \\ &= (\lambda - 3)(\lambda - 1) = 0. && (4.64) \end{aligned}$$

So the eigenvalues of $\boldsymbol{A}$ are $\lambda_1 = 1$ and $\lambda_2 = 3$ and the associated normalized eigenvectors are obtained via

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \boldsymbol{p}_1 = 1\boldsymbol{p}_1 \tag{4.65}$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \boldsymbol{p}_2 = 3\boldsymbol{p}_2\,. \tag{4.66}$$

This yields

$$\boldsymbol{p}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}, \boldsymbol{p}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}. \tag{4.67}$$

Step 2: Check for existence

The matrix is symmetric, we therefore know that the eigenvectors are linearly independent and the eigenvalues are distinct (but we can also quickly eye-ball this to validate our calculations), and so a diagonalization is possible.

Step 3: Compute the diagonalizing matrix $\boldsymbol{P}$

To compute the diagonalizing matrix we collect these normalized eigenvectors together

$$\boldsymbol{P} = [\boldsymbol{p}_1,\ \boldsymbol{p}_2] = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\-1 & 1\end{bmatrix} \tag{4.68}$$

so that we obtain

$$\begin{aligned}\boldsymbol{A}\boldsymbol{P} &= \frac{1}{\sqrt{2}}\begin{bmatrix}2 & 1\\1 & 2\end{bmatrix}\begin{bmatrix}1 & 1\\-1 & 1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 3\\-1 & 3\end{bmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\-1 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\0 & 3\end{bmatrix} = \boldsymbol{P}\boldsymbol{D}.\end{aligned} \tag{4.69}$$

We can now obtain the matrices of the eigendecomposition by right multiplying with $\boldsymbol{P}^{-1}$. Alternatively as the matrix $\boldsymbol{A}$ is symmetric we can use the orthogonality property of its eigenvectors with $\boldsymbol{P}^\top = \boldsymbol{P}^{-1}$ and solve for $\boldsymbol{A}$ directly to obtain the eigendecomposition:

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top \tag{4.70}$$

$$\begin{bmatrix}2 & 1\\1 & 2\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1\\-1 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\0 & 3\end{bmatrix}\begin{bmatrix}1 & -1\\1 & 1\end{bmatrix}. \tag{4.71}$$

The eigenvalue decomposition of a matrix has a number of convenient properties

- Diagonal matrices $\boldsymbol{D}$ have the nice property that they can be efficiently raised to a power. Therefore we can find a matrix power for a general matrix $\boldsymbol{A}$ via the eigenvalue decomposition

$$\boldsymbol{A}^k = (\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1})^k = \boldsymbol{P}\boldsymbol{D}^k\boldsymbol{P}^{-1}. \tag{4.72}$$

  Computing $\boldsymbol{D}^k$ is efficient because we apply this operation individually to any diagonal element.
- A different property of diagonal matrices is that they can be used to decouple variables. This will be important in probability theory to in-

terpret random variables, e.g., for the Gaussian distributions we will encounter in Section 6.6 and in applications such as dimensionality reduction Chapter 10.

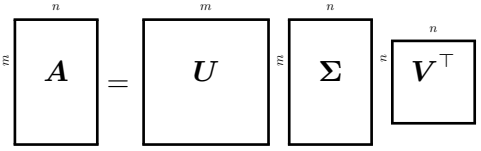
The eigenvalue decomposition requires square matrices, and for non-symmetric square matrices it is not guaranteed that we can transform them into diagonal form. It would be useful to be able to perform a decomposition on general matrices. In the next section, we introduce a more general matrix decomposition technique, the Singular Value Decomposition.

4.5 Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix is a central matrix decomposition method in linear algebra. It has been referred to as the "fundamental theorem of linear algebra" (Strang, 1993) because it can be applied to all matrices, not only to square matrices, and it always exists. Moreover, as we will explore in the following, the SVD of a linear mapping $\Phi : V \to W$ quantifies the resulting change between the underlying geometry of these two vector spaces. We recommend Kalman (1996); Roy and Banerjee (2014) for a deeper overview of the mathematics of the SVD.

Theorem 4.22 (SVD theorem)**.** *Let $\boldsymbol{A}^{m\times n}$ be a rectangular matrix of rank r, with $r \in [0, \min(m, n)]$. The Singular Value Decomposition or SVD of $\boldsymbol{A}$ is a decomposition of $\boldsymbol{A}$ of the form*

Singular Value Decomposition
SVD



(4.73)

where $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix composed of column vectors $\boldsymbol{u}_i$, $i = 1, \ldots, m$, and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix of column vectors $\boldsymbol{v}_j$, $j = 1, \ldots, n$, and $\boldsymbol{\Sigma}$ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geqslant 0$ and $\Sigma_{ij} = 0$, $i \neq j$. The SVD is always possible for any matrix $\boldsymbol{A}$.

The σ_i are called the singular values, $\boldsymbol{u}_i$ are called the left-singular vectors and $\boldsymbol{v}_j$ are called the right-singular vectors. By convention the singular vectors are ordered, i.e., $\sigma_1 \geqslant \sigma_2 \geqslant \sigma_r \geqslant 0$.

singular values
left-singular vectors
right-singular vectors

We will see a proof of this theorem later in this section. The SVD allows us to decompose general matrices, and the existence of the unique *singular value matrix* $\boldsymbol{\Sigma}$ requires attention. Observe that the $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is rectangular, that is it is non-square. In particular note that $\boldsymbol{\Sigma}$ is the same size as $\boldsymbol{A}$. This means that $\boldsymbol{\Sigma}$ has a diagonal submatrix that contains the singular values and needs additional zero vectors that increase the dimension.

singular value matrix

Specifically, if $m > n$ then the matrix $\mathbf{\Sigma}$ has diagonal structure up to row n and then consists of $\mathbf{0}^\top$ row vectors from $n+1$ to m below

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix} . \tag{4.74}$$

Conversely, if $m < n$ the matrix $\mathbf{\Sigma}$ has a diagonal structure up to column m and columns that consist of $\mathbf{0}$ from $m+1$ to n.

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & & 0 \\ 0 & 0 & \sigma_n & 0 & \dots & 0 \end{bmatrix} \tag{4.75}$$

4.5.1 Geometric Intuitions for the SVD

The SVD has a number of interesting geometric intuitions to offer to describe a transformation matrix. Broadly there are two intuitive views we can have. First we consider the SVD as sequential operations performed on the bases (discussed in the following), and second we consider the SVD as operations performed on sets of (data) points as described in Example 4.10.

The SVD can be interpreted as a decomposition of a linear mapping (recall Section 2.7.1) $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ into three operations (see Figure 4.7 for the following). The SVD intuition follows superficially a similar structure to our eigendecomposition intuition (confront Figure 4.7 for the SVD with Figure 4.6 for the eigendecomposition: Broadly speaking the SVD performs a basis change ($\boldsymbol{V}^\top$) followed by a scaling and augmentation (or reduction) in dimensionality ($\mathbf{\Sigma}$) and then performs a second basis change ($\boldsymbol{U}$). The SVD entails a number of important details and caveats which is why we will review our intuition in more detail and precision, than ew have then for the eigendecomposition.

Assume we are given a transformation matrix of Φ with respect to the standard bases B and C of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Moreover, assume a second basis $\tilde{B}$ of $\mathbb{R}^n$ and $\tilde{C}$ of $\mathbb{R}^m$. Then

1. $\boldsymbol{V}$ performs a basis change in the domain $\mathbb{R}^n$ from $\tilde{B}$ (represented by the red and green vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in Figure 4.7 top left) to the canonical basis B. It is useful here to recall our discussion of basis changes Section 2.7.2 and orthogonal matrices and orthonormal bases in Section **??**), as $\boldsymbol{V}^\top = \boldsymbol{V}^{-1}$ performs a basis change from B to $\tilde{B}$ (the red and green vectors are now aligned with the canonical basis in Figure 4.7 bottom left).

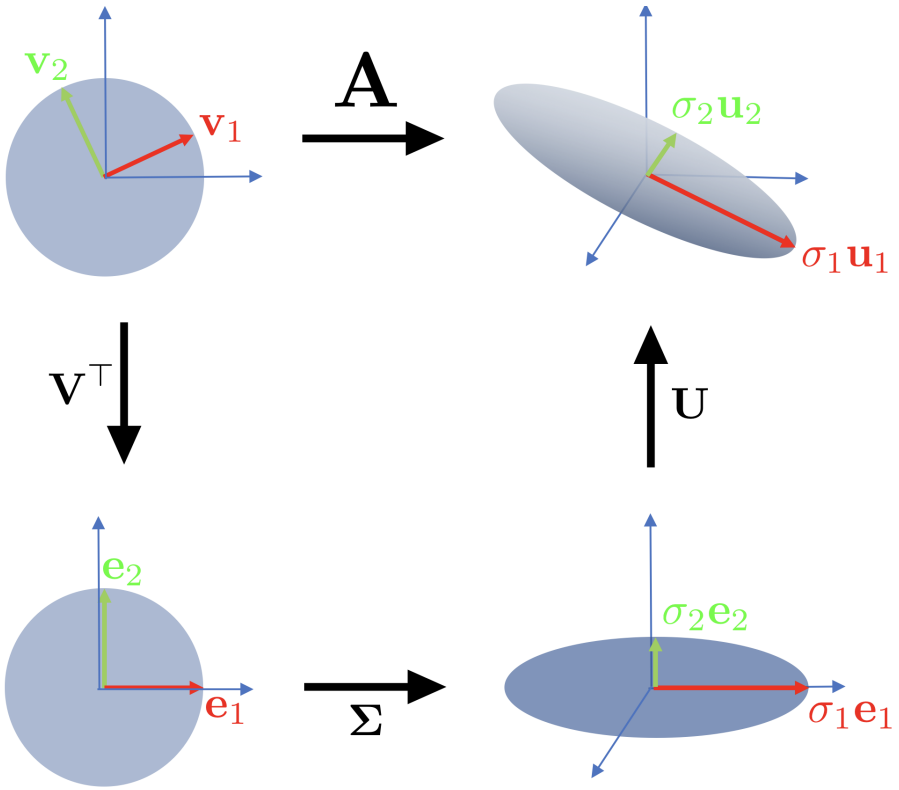


Figure 4.7 Intuition behind SVD of a $\boldsymbol{A} \in \mathbb{R}^{3\times 2}$ in the standard basis as sequential transformations. Top-left to bottom-left: $\boldsymbol{V}^\top$ performs a basis change in $\mathbb{R}^2$. Bottom-left-to-bottom right $\boldsymbol{\Sigma}$ performs a scaling and increases the dimensionality from $\mathbb{R}^2$ to $\mathbb{R}^3$. The ellipse in the bottom-right lives in $\mathbb{R}^3$ and the third dimension is orthogonal to the surface of the elliptical disk. Bottom-left to top-left: $\boldsymbol{U}$ performs a second basis change within $\mathbb{R}^3$.

2. Having changed the coordinate system to $\tilde{B}$, $\boldsymbol{\Sigma}$ scales the new coordinates by the singular values σ_i (and adding or deleting dimensions), i.e., $\boldsymbol{\Sigma}$ is the transformation matrix of Φ with respect to $\tilde{B}$ and $\tilde{C}$ (represented by the red and green vectors being stretched and lying in the $\boldsymbol{e}_1$-$\boldsymbol{e}_2$ plane which is now embedded in a third dimension in Figure 4.7 bottom right) .
3. $\boldsymbol{U}$ performs a basis change in the codomain $\mathbb{R}^m$ from $\tilde{C}$ into the canonical basis of $\mathbb{R}^m$ (represented by a rotation of red and green vectors out of the plane of the $\boldsymbol{e}_1$-$\boldsymbol{e}_2$ plane in Figure 4.7 bottom right).

The SVD expresses a change of basis in both the domain and codomain: The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are the bases $\tilde{B}$ of $\mathbb{R}^n$ and $\tilde{C}$ of $\mathbb{R}^m$, respectively. Note, how this is in contrast with the eigendecomposition that operates within the same vector space (where the same basis change is applied and then undone). What makes the SVD special is that these two (different) bases are simultaneously linked by the singular values matrix $\boldsymbol{\Sigma}$. We refer to Section 2.7.2 and Figure 2.9 for a more detailed discussion on basis change.

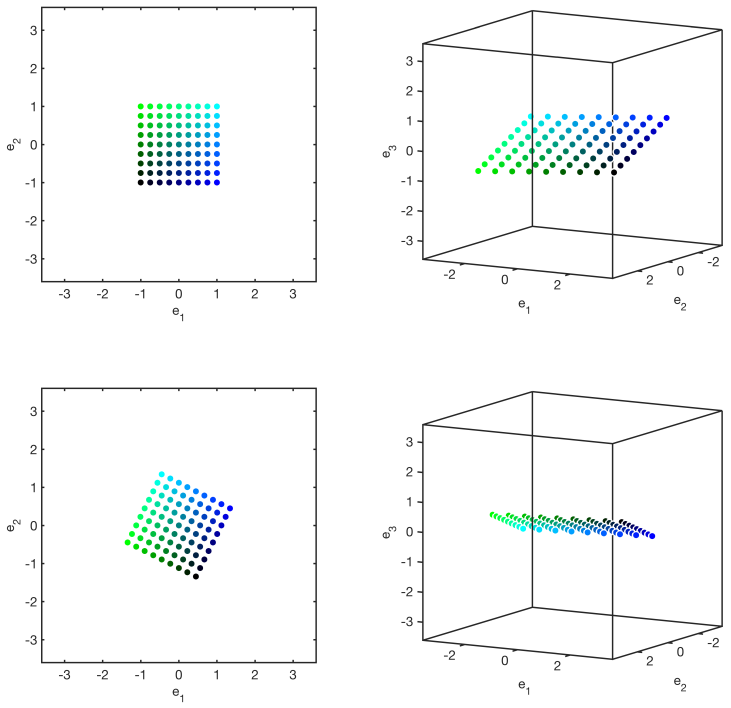


Figure 4.8 SVD and mapping of data points. The panels follow the same anti-clockwise structure of Figure 4.7. See main text for details.

Example 4.10

Data points and the SVD. Consider a mapping of a square grid of points $\mathcal{X} \in \mathbb{R}^2$ which fit in a box of size 2×2 centered at the origin. Using the standard basis we map these points using

$$\boldsymbol{A} = \begin{bmatrix} 1 & -2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.76}$$

$$= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{4.77}$$

$$= \begin{bmatrix} 0.913 & 0 & -0.408 \\ -0.365 & 0.4472 & -0.816 \\ 0.182 & 0.894 & 0.4082 \end{bmatrix} \begin{bmatrix} 2.449 & 0 \\ 0 & 1.0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.4472 & -0.894 \\ 0.8941 & 0.4472 \end{bmatrix} \tag{4.78}$$

.

We start with a set of points $\mathcal{X}$ (colored dots, see top left panel of Figure 4.8) arranged in a grid.

The points $\mathcal{X}$ after rotating them using $\boldsymbol{V}^\top \in \mathbb{R}^{2\times 2}$ are shown in the bottom-left panel of Figure 4.8. After a mapping $\boldsymbol{\Sigma}$ to the codomain $\mathbb{R}^3$

(see bottom right panel in Figure 4.8) we can see how all the points lie on the $\boldsymbol{e}_1$-$\boldsymbol{e}_2$ plane. The third dimension was added, and the arrangement of points has been stretched by the singular values.

The direct mapping of the points $\mathcal{X}$ by $\boldsymbol{A}$ to the codomain $\mathbb{R}^3$ equals the transformation of $\mathcal{X}$ by $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{U}$ performs a rotation within the codomain $\mathbb{R}^3$ so that the mapped points are no longer restricted to the $\boldsymbol{e}_1$-$\boldsymbol{e}_2$ plane; they still are on a plane (see top-right panel of Figure 4.8.

4.5.2 Existence and Construction of the SVD

We will next discuss why the SVD exists and show how to compute it in detail. The SVD of a general matrix is related to the eigendecomposition of a square matrix and has some similarities.

Remark. Compare the eigenvalue decomposition of a symmetric matrix

$$\boldsymbol{S} = \boldsymbol{S}^\top = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top \tag{4.79}$$

(which always exists) to the structure of the SVD of

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top . \tag{4.80}$$

We identify

$$\boldsymbol{U} = \boldsymbol{P} = \boldsymbol{V} , \tag{4.81}$$
$$\boldsymbol{D} = \boldsymbol{\Sigma} , \tag{4.82}$$

so that the SVD of symmetric matrices is their eigenvalue decomposition. ♢

In the following we will explore why Theorem 4.22 should hold and how it is constructed. Computing the SVD of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ its existence is equivalent to finding two sets of orthonormal bases $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m)$ and $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ of the domain $\mathbb{R}^m$ and the codomain $\mathbb{R}^n$, respectively. From these ordered bases we will construct the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively.

Our plan is to start with constructing the orthonormal set of right-singular vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^n$. We then construct the orthonormal set of left-singular vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^n$. Thereafter, we will link the two and require that the orthogonality of the $\boldsymbol{v}_i$ is preserved under the transformation of $\boldsymbol{A}$. This is important because we know the images $\boldsymbol{A}\boldsymbol{v}_i$ form a set of orthogonal vectors. We will then need to normalize these images by scalar factors, which will turn out to be the singular values, so that the images are also normalized in length.

Let us begin with constructing the right-singular vectors. We have previously learned that the eigenvalue decomposition is a method to construct

an orthonormal basis, and it always exists for symmetric matrices by Theorem 4.21. Moreover, from Theorem 4.10 we can always construct a symmetric matrix $\boldsymbol{A}^\top\boldsymbol{A} \in \mathbb{R}^{n\times n}$ from any rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. Thus, we can always diagonalize $\boldsymbol{A}^\top\boldsymbol{A}$ and obtain

$$\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top = \boldsymbol{P}\begin{bmatrix}\lambda_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \lambda_n\end{bmatrix}\boldsymbol{P}^\top. \tag{4.83}$$

Take note that the $\lambda_i \geqslant 0$ are the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$. Let us assume the SVD of $\boldsymbol{A}$ exists and inject (4.73) into (4.83).

$$\boldsymbol{A}^\top\boldsymbol{A} = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top) = \boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top. \tag{4.84}$$

where $\boldsymbol{U}, \boldsymbol{V}$ are orthogonal matrices. Therefore, with $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$ we obtain

$$\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\begin{bmatrix}\sigma_1^2 & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & \sigma_n^2\end{bmatrix}\boldsymbol{V}^\top. \tag{4.85}$$

Comparing now (4.83) and (4.85) we identify

$$\boldsymbol{V} = \boldsymbol{P}, \tag{4.86}$$
$$\sigma_i^2 = \lambda_i. \tag{4.87}$$

Therefore, the eigenvectors $\boldsymbol{P}$ of $\boldsymbol{A}^\top\boldsymbol{A}$ are the right-singular vectors $\boldsymbol{V}$ of $\boldsymbol{A}$ (see (4.86)). They form an orthonormal basis because of Theorem 4.21, for the domain of the SVD. Moreover, the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ are the squared singular values of $\boldsymbol{\Sigma}$ (see (4.87)).

Let us now repeat this derivation but this time we will focus on obtaining the left singular vectors $\boldsymbol{U}$ instead of $\boldsymbol{V}$. Therefore we start again by computing the SVD of a symmetric matrix, this time $\boldsymbol{A}\boldsymbol{A}^\top \in \mathbb{R}^{m\times m}$ (instead of the above $\boldsymbol{A}^\top\boldsymbol{A} \in \mathbb{R}^{n\times n}$). We inject again (4.73) and obtain:

$$\boldsymbol{A}\boldsymbol{A}^\top = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top = \boldsymbol{U}\boldsymbol{\Sigma}^\top\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top \tag{4.88}$$
$$= \boldsymbol{U}\begin{bmatrix}\sigma_1^2 & 0 & 0\\ 0 & \ddots & 0\\ 0 & 0 & \sigma_m^2\end{bmatrix}\boldsymbol{U}^\top. \tag{4.89}$$

We can now obtain from the same arguments about symmetric matrices and their diagonalization, now applied to $\boldsymbol{A}\boldsymbol{A}^\top$, the orthonormal eigenvectors of $\boldsymbol{A}^\top\boldsymbol{A}$. These are the left-singular vectors $\boldsymbol{U}$ and form an orthonormal basis set in the codomain of the SVD.

This leaves the question of the structure of the matrix $\boldsymbol{\Sigma}$. We need to show that regardless of $n > m$ or $n < m$, that $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{A}^\top\boldsymbol{A}$ have the

same non-zero eigenvalues: Let us assume that λ is a non-zero eigenvalue of $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{x}$ is an eigenvector belonging to λ_i. Then

$$(\boldsymbol{A}\boldsymbol{A}^\top)\boldsymbol{x} = \lambda\boldsymbol{x} \tag{4.90}$$

left multiplying by $\boldsymbol{A}$ yields and pulling on the right-hand side the scalar factor λ forward

$$\boldsymbol{A}(\boldsymbol{A}\boldsymbol{A}^\top)\boldsymbol{x} = \boldsymbol{A}(\lambda\boldsymbol{x}) = \lambda(\boldsymbol{A}\boldsymbol{x}) \tag{4.91}$$

and we can use (2.30) to reorder the left-hand side factors

$$(\boldsymbol{A}^\top\boldsymbol{A})(\boldsymbol{A}^\top\boldsymbol{x}) = \lambda(\boldsymbol{A}\boldsymbol{x})\,. \tag{4.92}$$

This is the eigenvalue equation for $\boldsymbol{A}\boldsymbol{A}^\top$. Therefore, λ is the same eigenvalue for $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{A}^\top\boldsymbol{A}$, and $\boldsymbol{A}\boldsymbol{x}$ is its eigenvector. Thus, both matrices have the same non-zero eigenvalues. Thus, the $\boldsymbol{\Sigma}$ matrices in the SVD for both cases have to be the same.

The last step in the proof is to link up all the parts so far. We have now an orthonormal set of right-singular vectors in $\boldsymbol{V}$. But, to finish construction of the SVD we link them to the orthonormal vectors $\boldsymbol{U}$. To reach this goal we use the fact the images of the $\boldsymbol{v}_i$ under $\boldsymbol{A}$ have to be orthonormal, too. Using the results from Section 3.4, we require that the inner product between $\boldsymbol{A}\boldsymbol{v}_i$ and $\boldsymbol{A}\boldsymbol{v}_j$ must be 0 for $i \neq j$. For any two orthogonal eigenvectors $\boldsymbol{v}_i, \boldsymbol{v}_j$, $i \neq j$ it holds that

$$(\boldsymbol{A}\boldsymbol{v}_i)^\top(\boldsymbol{A}\boldsymbol{v}_j) = \boldsymbol{v}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})\boldsymbol{v}_j = \boldsymbol{v}_i^\top(\lambda_j\boldsymbol{v}_j) = \lambda_j\boldsymbol{v}_i^\top\boldsymbol{v}_j = 0\,. \tag{4.93}$$

For the case $m > r$ this holds for all pairs $\boldsymbol{A}\boldsymbol{v}_1, \ldots, \boldsymbol{A}\boldsymbol{v}_r$ the images are a basis of $\mathbb{R}^m$, while if any further vectors $\boldsymbol{A}\boldsymbol{v}_i, i > r$ exist, they must be in the nullspace of $\boldsymbol{A}$ (see remark after proof for the converse case).

To complete the SVD construction we need left-singular vectors that are ortho*normal*: we normalize the images of the right-singular vectors $\boldsymbol{A}\boldsymbol{v}_i$ and call them $\boldsymbol{u}_i$,

$$\boldsymbol{u}_i = \frac{\boldsymbol{A}\boldsymbol{v}_i}{\|\boldsymbol{A}\boldsymbol{v}_i\|} = \frac{1}{\sqrt{\lambda_i}}\boldsymbol{A}\boldsymbol{v}_i = \frac{1}{\sigma_i}\boldsymbol{A}\boldsymbol{v}_i \tag{4.94}$$

where the last equality was obtained from (4.87) and from equation (4.89) showing us that the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^\top$ are such that $\sigma_i^2 = \lambda_i$.

Therefore, the eigenvectors of $\boldsymbol{A}^\top\boldsymbol{A}$, we which we know are the right-singular vectors $\boldsymbol{v}_i$ and their normalized images under $\boldsymbol{A}$, the left singular vectors $\boldsymbol{u}_i$, form two self-consistent sets of orthonomal bases that are coupled by the singular value matrix $\boldsymbol{\Sigma}$.

Remark. Let us rearrange (4.94) to obtain the *singular value equation* singular value equation

$$\boldsymbol{A}\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i\,, \quad i = 1, \ldots, r\,. \tag{4.95}$$

This equation closely resembles the eigenvalue equation (4.27), but the vectors on the left and the right-hand sides are not the same.

For $n > m$ (4.95) holds only for $i \leqslant m$ and (4.95) say nothing about the $\boldsymbol{u}_i$ for $i > m$, but we know by construction that they are orthonormal. Conversely for $m > n$, then (4.95) holds only for $i \leqslant n$. For $i > n$ we have $\boldsymbol{A}\boldsymbol{v}_i = 0$ and we still know that the $\boldsymbol{v}_i$ form an orthonormal set. This means that the SVD also supplies an orthonormal basis of the kernel (or null space) or $\boldsymbol{A}$, the set of vectors $\boldsymbol{x}$ with $\boldsymbol{A}\boldsymbol{x} = 0$ (see Section 2.7.3).

Moreover, collecting the $\boldsymbol{v}_i$ as the columns of $\boldsymbol{V}$ and $\boldsymbol{u}_i$ as the columns of $\boldsymbol{U}$ yields

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}\,. \tag{4.96}$$

where $\boldsymbol{\Sigma}$ has the same dimensions as $\boldsymbol{A}$ and a diagonal structure for rows $1, \ldots, r$. Hence, right-multiplying with $\boldsymbol{V}^\top = \boldsymbol{V}^{-1}$ yields $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, which is again our singular value decomposition of $\boldsymbol{A}$. ♢

Example 4.11

Let us find the singular value decomposition of

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}. \tag{4.97}$$

Step 1: Compute the symmetrized matrix $\boldsymbol{A}^\top\boldsymbol{A}$

$$\boldsymbol{A}^\top\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & -2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & -2 & 1 \\ -2 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}. \tag{4.98}$$

Step 2: Compute the eigenvalue decomposition of $\boldsymbol{A}^\top\boldsymbol{A}$

We compute the singular values and right-singular vectors through the eigenvalue decomposition of $\boldsymbol{A}^\top\boldsymbol{A}$

$$\boldsymbol{A}^\top\boldsymbol{A} = \begin{bmatrix} 5 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.99}$$

$$= \begin{bmatrix} \frac{5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{2}} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} 6 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \frac{5}{\sqrt{30}} & \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{30}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{-1}{\sqrt{2}} & \frac{-2}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top. \tag{4.100}$$

Note, that due to our orthonormality requirement implies that we chose the 3rd column of $\boldsymbol{P}$ so as to be orthogonal to the other two columns. As the singular values σ_i are the square root of the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ we obtain them straight from $\boldsymbol{D}$. Note that because $\mathrm{rk}(\boldsymbol{A}) = 2$ there are only two non-zero singular values, $\sigma_1 = \sqrt{6}$ and $\sigma_2 = 1$. The singular value matrix must be the same size as $\boldsymbol{A}$, hence,

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{4.101}$$

We also have obtained already the right-singular vectors because

$$\boldsymbol{V} = \boldsymbol{P} = \begin{bmatrix} \frac{5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{2}} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{2}} \end{bmatrix} . \tag{4.102}$$

Step 3: Compute the normalized image of the right-singular vectors

We now find the left singular-vectors by computing the image of the right-singular vectors under $\boldsymbol{A}$ and normalizing them by dividing them by their corresponding singular value.

$$\boldsymbol{u}_1 = \frac{1}{\sigma_1}\boldsymbol{A}\boldsymbol{v}_1 = \frac{1}{\sqrt{6}}\begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}\begin{bmatrix} \frac{5}{\sqrt{30}} \\ \frac{-2}{\sqrt{30}} \\ \frac{1}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{bmatrix} , \tag{4.103}$$

$$\boldsymbol{u}_2 = \frac{1}{\sigma_2}\boldsymbol{A}\boldsymbol{v}_2 = \frac{1}{1}\begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} \tag{4.104}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2] = \frac{1}{\sqrt{5}}\begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix} . \tag{4.105}$$

Note that in practice the approach illustrated here has poor numerical behaviour, and the SVD of $\boldsymbol{A}$ is computed without resorting to the eigenvalue decomposition of $\boldsymbol{A}^\top\boldsymbol{A}$.

4.5.3 Eigenvalue Decomposition vs Singular Value Decomposition

Let us consider the eigendecomposition $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$ and SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and review the core elements of the past sections.

- The SVD always exists for any matrix $\mathbb{R}^{n\times m}$. The eigendecomposition is only defined for square matrices $\mathbb{R}^{n\times n}$ and only exists if we can find a basis of eigenvectors (or n independent eigenvectors).
- The vectors in the eigendecomposition matrix $\boldsymbol{P}$ are not necessarily orthogonal, so the change of basis is not a simple rotation and scaling. On the other hand, the vectors in the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ in the SVD are orthonormal, so they do represent rotations (or possibly reflections).
- Both the eigendecomposition and the SVD are compositions of three linear mappings:
  1. Change of basis in the domain
  2. Independent scaling of each new basis vector and mapping from domain to co-domain
  3. Change of basis in the co-domain

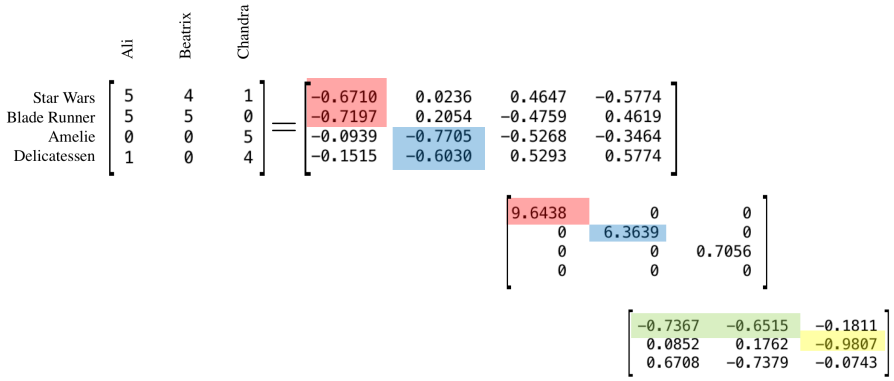


Figure 4.9 Movie ratings of three people for four movies and its SVD decomposition.

A key difference between the eigendecomposition and the SVD is that in the SVD, domain and co-domain can be vector spaces of different dimensions.

- In the SVD, the left and right singular vector matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are generally not inverse of each other. In the eigendecomposition the eigenvector matrices $\boldsymbol{P}$ and $\boldsymbol{P}^{-1}$ are inverses of each other.
- In the SVD, the entries in the diagonal matrix $\boldsymbol{\Sigma}$ are all real and nonnegative, which is not generally true for the diagonal matrix in the eigendecomposition.
- The SVD and the eigendecomposition are closely related through their projections
  - The left-singular vectors of $\boldsymbol{A}$ are eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$
  - The right-singular vectors of $\boldsymbol{A}$ are eigenvectors of $\boldsymbol{A}^\top\boldsymbol{A}$.
  - The non-zero singular values of $\boldsymbol{A}$ are the square roots of the non-zero eigenvalues of $\boldsymbol{A}\boldsymbol{A}^\top$, and equal the non-zero eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$.
- For symmetric matrices the eigenvalue decomposition and the SVD are one and the same.

Example 4.12 (Finding Structure in Movie Ratings and Consumers)
Let us understand a way to interpret the practical meaning of the SVD by analysing data on people and their preferred movies. Consider 3 viewers (Ali, Beatrix, Chandra) rating 4 different movies (Star Wars, Blade Runner, Amelie, Delicatessen). Their ratings are values between 0 (worst) and 5 (best) and encoded in a data matrix $\boldsymbol{A} \in \mathbb{R}^{4\times 3}$ (see Figure 4.9). Each row represents a movie and each column a user. Thus, the column vectors of movie ratings, one for each viewer, are $\boldsymbol{x}_{\text{Ali}}$, $\boldsymbol{x}_{\text{Beatrix}}$, $\boldsymbol{x}_{\text{Chandra}}$.

Factoring $\boldsymbol{A}$ using SVD provides a way to capture the relationships of how people rate movies, and especially if there is a structure linking which

people like which movies. Applying the SVD to our data matrix makes a number of assumptions

1. All viewers rate movies consistently using the same linear mapping.
2. There are no errors or noise in the ratings data.
3. We interpret the left-singular vectors $\boldsymbol{u}_i$ as stereotypical movies and the right-singular vectors $\boldsymbol{v}_j$ as stereotypical viewers.

We then make the assumption that any viewer's specific movie preferences can be expressed as a linear combination of the $\boldsymbol{v}_j$. Similarly, any, movie's like-ability can be expressed as a linear combination of the $\boldsymbol{u}_i$.

Let us look at the specific outcome of performing SVD: The first left-singular vector $\boldsymbol{u}_1$ has large absolute values for the two science fiction movies and a large first singular value (red shading in Figure 4.9). Thus, this groups a type of users with a set of movies – we interpret this here as the notion of a science fiction theme. Similarly, the first right-singular $\boldsymbol{v_1}$ shows large absolute values for Ali and Beatrix which give high ratings to science fiction movies (green shading in Figure 4.9). This suggests that $\boldsymbol{v_1}$ may reflect an idealized notion of a science fiction lover.

Similarly, $\boldsymbol{u}_2$, seems to capture a French art house film theme, and $\boldsymbol{v}_2$ may be reflecting that Chandra is to close to an idealized lover of such movies. An idealized science fiction lover is a purist and only loves science fiction movies, so a science fiction lover $\boldsymbol{v}_1$ gives a rating of zero to everything but science fiction themed – this logic is implied by us requiring a diagonal substructure for the singular value matrix. A specific movie is therefore represented by how it decomposes (linearly) into its stereotypical movies. Likewise a person would be represented by how they decompose (via linear combination) into movie themes.

Remark. It is worth discussing briefly SVD terminology and conventions as there are different versions used in the literature—the mathematics remains invariant to these differences— but can confuse the unaware reader:

- For convenience in notation and abstraction we use here an SVD notation where the SVD is described as having two square left- and right-singular vector matrices, but a non-square singular value matrix. Our definition (4.73) for the SVD is sometimes called the *full SVD*. full SVD
- Some authors define the SVD a bit differently, for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $m \geqslant n$

$$\underset{m\times n}{\boldsymbol{A}} = \underset{m\times n, n\times n, n\times n}{\boldsymbol{U}\ \boldsymbol{\Sigma}\ \boldsymbol{V}^T} \tag{4.106}$$

  Some authors call this the *reduced SVD* (e.g. Datta (2010)) other refer to this as *the* SVD (e.g. Press et al. (2007)). This alternative format changes merely how the matrices are constructed but leaves the mathematical structure of the SVD unchanged. The convenience of this reduced SVD

alternative notation is that $\mathbf{\Sigma}$ is diagonal, as in the eigenvalue decomposition. However, it looses the interpretation of $\mathbf{\Sigma}$ as a transformation matrix.

- In Section 4.6, we will learn about matrix approximation techniques using the SVD, which is also called the *truncated SVD*.

truncated SVD

- One can also define the SVD of a rank-r matrix $\boldsymbol{A}$ so that $\boldsymbol{U}$ is an $m \times r$ matrix, $\mathbf{\Sigma}$ as a diagonal matrix $r \times r$, and $\boldsymbol{V}$ as $r \times n$ matrix. This construction is very similar to our definition, and ensures that the diagonal matrix $\mathbf{\Sigma}$ has only non-zero entries along the diagonal. The main convenience of this alternative notation is that $\mathbf{\Sigma}$ is diagonal, as in the eigenvalue decomposition.
- One could also introduce the restriction that the SVD for $\boldsymbol{A}$ only applies to $m \times n$ matrices with $m > n$. However, this restriction is practically unnecessary. When $m < n$ the SVD decomposition will yield $\mathbf{\Sigma}$ with more zero columns than rows and, consequently, the singular values $\sigma_{m+1}, \dots, \sigma_n$ are implicitly 0.

$\diamondsuit$

The SVD is used in a variety of applications in machine learning from least squares problems in curve fitting to solving systems of linear equations. These applications harness various important properties of the SVD, its relation to the rank of a matrix and its ability to approximate matrices of a given rank with lower rank matrices. Substituting the SVD form of a matrix in computations rather use the original matrix has often the advantage of making the calculation more robust to numerical rounding errors. As we will explore in the next section the SVD's ability to approximate matrices with "simpler" matrices in a principled manner opens up machine learning applications ranging from dimensionality reduction, topic modeling to data compression and clustering.

4.6 Matrix Approximation

We will now investigate how the SVD allows us to represent a matrix $\boldsymbol{A}$ as a sum of simpler matrices $\boldsymbol{A}_i$.

Let us construct a rank-1 $m \times n$ matrix $\boldsymbol{A}_i$ as

$$\boldsymbol{A}_i = \boldsymbol{u}_i \boldsymbol{v}_i^\top \tag{4.107}$$

which is formed by the outer product of ith orthogonal column vector of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively (see Figure 4.10 for a a visual example). For a matrix $\boldsymbol{A}$ of rank r the matrix can be decomposed into a sum of rank-1 matrices as follows $\boldsymbol{A}_i$:

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top = \sum_{i=1}^{r} \sigma_i \boldsymbol{A}_i \tag{4.108}$$

where the outer product matrices $\boldsymbol{A}_i$ are weighed by the size of the ith

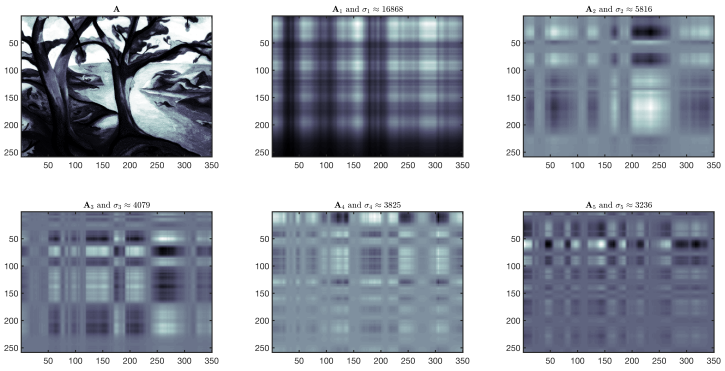


Figure 4.10 (Top left) A grayscale image is a 280×350 matrix of values between 0 (black) and 1 (white). (Middle left to Bottom right) rank-1 matrices $\boldsymbol{A}_1 \dots \boldsymbol{A}_5$ and their corresponding singular values $\sigma_1, \dots, \sigma_5$. Note, that the grid like structure of each rank-1 matrix is imposed by the outer-product of the left and right singular vectors.

singular value σ_i. Thus, the sum of the outer products of matching left and right singular vectors (weighted by their singular value) is equal to $\boldsymbol{A}$. Note, that any terms $i > r$ are zero, as the singular values will be 0. We can see why (4.107) holds: the diagonal structure of the singular value matrix $\boldsymbol{\Sigma}$ multiplies only matching left- and right-singular vectors $(\boldsymbol{u}_i, \boldsymbol{v}_i^\top)$ and adds them up, while setting non-matching left- and right-singular vectors $(\boldsymbol{u}_i, \boldsymbol{v}_j^\top, i \neq j)$ to zero.

In the previous paragraph we introduced a low-rank matrix $\boldsymbol{A}_i$ (of rank 1). We summed up the r individual rank-1 matrices to obtain a rank r matrix $\boldsymbol{A}$. What happens if the sum does not over all matrices $\boldsymbol{A}_i$ from $i = 1 \dots r$ but instead run the sum only up to an intermediate value $k < r$. We are obtaining now an approximation of $\boldsymbol{A}$ that we call the *rank-k approximation* $\widehat{\boldsymbol{A}}(k)$

rank-k approximation

$$\widehat{\boldsymbol{A}}(k) = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top \tag{4.109}$$

of $\boldsymbol{A}$ with $\mathrm{rk}(\widehat{\boldsymbol{A}}) = k$.

It would be useful if we could measure how large the difference between $\boldsymbol{A}$ and its approximation $\widehat{\boldsymbol{A}}(k)$ is in terms of a single number – we thus need the notion of a norm. We have already used norms on vectors that measure the length of a vector. By analogy we can also define a norm on matrices (one of the many ways to define matrix norms).

Definition 4.23 (Spectral norm of a matrix)**.** The spectral norm of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is defined as the following for $\boldsymbol{x} \in \mathbb{R}^n$

$$\|\boldsymbol{A}\|_2 := \max_{\boldsymbol{x}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \quad \boldsymbol{x} \neq \boldsymbol{0}. \tag{4.110}$$

The operator norm implies how long any vector $\boldsymbol{x}$ can at most become once it is multiplied by $\boldsymbol{A}$. This maximum lengthening is given by the SVD of $\boldsymbol{A}$.

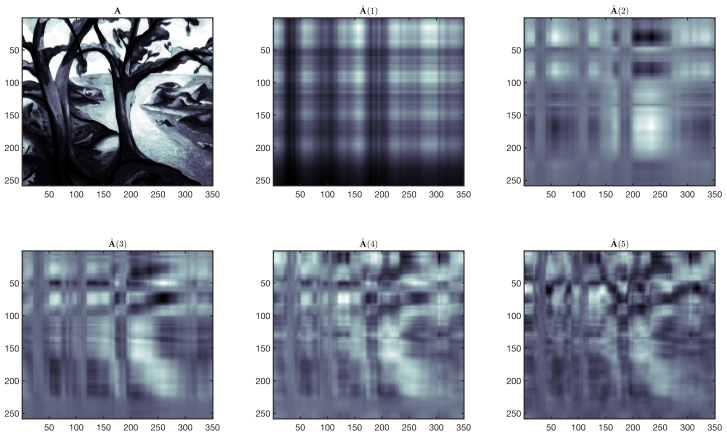


Figure 4.11 (Top left) The same grayscale image as in Figure 4.10. (Middle left to Bottom right) Image reconstruction using the low-rank approximation of the SVD: (Top middle) is $\widehat{\boldsymbol{A}(1)} = \sigma_1 \boldsymbol{A}_1$. (Top right) is the rank-2 approximation $\widehat{\boldsymbol{A}(2)} = \sigma_1 \boldsymbol{A}_1 + \sigma_2 \boldsymbol{A}_2$. (Bottom left to Bottom right) are $\widehat{\boldsymbol{A}(3)}$ to $\widehat{\boldsymbol{A}(5)}$. Note how the shape of the trees becomes increasingly visible and clearly recognizable in the a rank-6 approximation. While the original image requires $280 \times 350 = 98000$ numbers, the rank-6 approximation requires us only to store only the 6 singular values and the 6 left and right singular vectors (255 and 380 dimensional each) for a total of $6 \times (250+380+1) = 3786$ numbers – just about 4% of the original.

Theorem 4.24. *The spectral norm of $\boldsymbol{A}$ is its largest singular value σ_1.*

We provide here a derivation of the largest singular value of matrix $\boldsymbol{A}$, illustrating the relation between the spectral norm and SVD.

$$\|\boldsymbol{A}\|_2 = \max_{\boldsymbol{x}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \max_{\boldsymbol{x}} \sqrt{\frac{\|\boldsymbol{A}\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_2^2}} \tag{4.111}$$

$$= \max_{\boldsymbol{x}} \sqrt{\frac{(\boldsymbol{x}\boldsymbol{A})^\top(\boldsymbol{A}\boldsymbol{x})}{\boldsymbol{x}^\top\boldsymbol{x}}} = \max_{\boldsymbol{x}} \sqrt{\frac{\boldsymbol{x}^\top(\boldsymbol{A}^\top\boldsymbol{A})\boldsymbol{x}}{\boldsymbol{x}^\top\boldsymbol{x}}} \tag{4.112}$$

the matrix $\boldsymbol{A}^\top\boldsymbol{A}$ is symmetric by construction and therefore we can compute the eigenvalue decomposition $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top$

$$\|\boldsymbol{A}\|_2 = \max_{\boldsymbol{x}} \sqrt{\frac{\boldsymbol{x}^\top(\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top)\boldsymbol{x}}{\boldsymbol{x}^\top\boldsymbol{x}}}, \tag{4.113}$$

$$\tag{4.114}$$

where $\boldsymbol{D}$ is a diagonal matrix containing the eigenvalues. Recall that $\boldsymbol{P}^\top$ and $\boldsymbol{P}$ perform merely a basis change and then undo it. Therefore, the most a vector $\boldsymbol{x}$ can be lengthened is if it is collinear with the eigenvector associated with the largest eigenvalue.

$$\|\boldsymbol{A}\|_2 = \sqrt{\lambda_1} \tag{4.115}$$

the largest eigenvalue of $\boldsymbol{A}^\top\boldsymbol{A}$ is by (4.87) the largest singular value of $\boldsymbol{A}$

$$\|\boldsymbol{A}\|_2 = \sigma_1 \tag{4.116}$$

Theorem 4.25 (Eckart-Young (or Eckart-Young-Minsky) theorem)**.** *Let*

$\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a matrix of rank r and and $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ be a matrix of rank k. For any $k \leqslant r$ such that $\widehat{\boldsymbol{A}}(k) = \sum_i^k \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$, it holds that

$$\left\|\boldsymbol{A} - \widehat{\boldsymbol{A}}(k)\right\|_2 = \sigma_{k+1} \tag{4.117}$$

$$= \min_{\mathrm{rk}(\boldsymbol{B})\leqslant k} \|\boldsymbol{A} - \boldsymbol{B}\|_2 \ . \tag{4.118}$$

Remark. We can interpret the rank-k approximation obtained with the SVD is a projection of the full rank matrix $\boldsymbol{A}$ onto the lower-dimensional space of rank at-most-k matrices. Of all possible projections the SVD rank-k approximation minimizes the difference with respect to the spectral norm between $\boldsymbol{A}$ and any rank-k matrix. ♢

We can retrace some of the steps to understand why (4.117) should hold. We observe that the difference between $\boldsymbol{A} - \widehat{\boldsymbol{A}}(k)$ is a matrix containing the sum of the remaining rank-1 matrices

$$\boldsymbol{A} - \widehat{\boldsymbol{A}}(k) = \sum_{i=k+1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top \tag{4.119}$$

Thus, by applying the definition of the spectral norm, (4.110), the most a vector can be lengthened by the difference matrix is given its largest singular value i.e. σ_{k+1}, which is the difference matrix's spectral norm.

Let us proceed to better understand (4.118) validity. We assume that there is another matrix $\boldsymbol{B}$ with $\mathrm{rk}(\boldsymbol{B}) \leqslant k$ such that

$$\|\boldsymbol{A} - \boldsymbol{B}\|_2 < \left\|\boldsymbol{A} - \widehat{\boldsymbol{A}}(k)\right\|_2 \tag{4.120}$$

Then there exists an $(n-k)$-dimensional nullspace $Z \subseteq \mathbb{R}^n$ such that $\boldsymbol{x} \in Z \implies \boldsymbol{B}\boldsymbol{x} = \boldsymbol{0}$. In other words, have an n-dimensional space $\mathbb{R}^n$ in which lies a lower dimensional nullspace of $\boldsymbol{B}$. Then it follows that

$$\|\boldsymbol{A}\boldsymbol{x}\|_2 = \|(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{x}\|_2 \ , \tag{4.121}$$

and by using a version of the Cauchy-Schwartz inequality (3.5) that encompasses norms of matrices we obtain

$$\|\boldsymbol{A}\boldsymbol{x}\|_2 \leqslant \|\boldsymbol{A} - \boldsymbol{B}\|_2 \|\boldsymbol{x}\|_2 < \sigma_{k+1} \|\boldsymbol{x}\|_2 \tag{4.122}$$

Therefore, V is a $(n-k)$ dimensional subspace where $\|\boldsymbol{A}\boldsymbol{x}\|_2 < \sigma_{k+1} \|\boldsymbol{x}\|_2$.

On the other hand there is a $(n+1)$-dimensional subspace where $\|\boldsymbol{A}\boldsymbol{x}\|_2 \geqslant \sigma_{k+1} \|\boldsymbol{x}\|_2$ which is spanned by the right singular vector $\boldsymbol{v}_{k+1}$ of $\boldsymbol{A}$. Adding up dimensions of these two spaces yields a number greater n, as there must be a non-zero vector in both spaces. This is a contradiction because of the Rank-Nullity Theorem (recall Theorem 2.23 in Section 2.7.3).

The Eckart-Young theorem implies that we can use SVD to reduce a rank-r matrix $\boldsymbol{A}$ to a rank-k matrix $\widehat{\boldsymbol{A}}$ in a principled, optimal (in the spectral norm sense) manner. The effect of the low-rank approximation is that we can obtain a more compact representation of the values of the

matrix with limited loss of information, this is a form of data compression. Therefore, the low-rank approximation of a matrix appears in many machine learning applications, such as image processing, noise filtering, and regularization of ill-posed problems. Furthermore, it plays a key role in dimensionality reduction and principal component analysis as we shall see in Chapter 10.

Example 4.13 (Finding Structure in Movie Ratings and Consumers (continued))
Following from our previous movie rating example we can now apply the concept of low-rank approximation to describe the data matrix. Recall that our first singular value captures the notion of science fiction theme in movies and science fiction lovers. Thus, by using only the first singular value term in a rank-1 decomposition of the movie rating matrix we obtain the following predicted ratings

$$\boldsymbol{M}_1 = \sigma_1(\boldsymbol{u}_1\boldsymbol{v}_1^\top) \tag{4.123}$$

$$= 9.6438 \begin{bmatrix} -0.6710 \\ -0.7197 \\ -0.0939 \\ -0.1515 \end{bmatrix} \begin{bmatrix} -0.7367 & -0.6515 & -0.1811 \end{bmatrix} \tag{4.124}$$

$$= \begin{bmatrix} 4.7673 & 4.2154 & 1.1718 \\ 5.1138 & 4.5218 & 1.2570 \\ 0.6671 & 0.5899 & 0.1640 \\ 1.0765 & 0.9519 & 0.2646 \end{bmatrix} \tag{4.125}$$

This first rank-1 approximation $\boldsymbol{M}_1$ is insightful: it tells us that Ali and Beatrix like science fiction movies such as Star Wars and Bladerunner (entries have values > 4), but on the other hand fails to capture the ratings of the other movies by Chandra. This is not surprising as Chandra's type of movies are not captured by the first singular value. The second singular value however gives us a better rank-1 approximation for those movie theme-movie lovers types.

$$\boldsymbol{M}_2 = \sigma_2(\boldsymbol{u}_2\boldsymbol{v}_2^\top) \tag{4.126}$$

$$= 6.3639 \begin{bmatrix} 0.0236 \\ 0.2054 \\ -0.7705 \\ -0.6030 \end{bmatrix} \begin{bmatrix} 0.0852 & 0.1762 & -0.9807 \end{bmatrix} \tag{4.127}$$

$$= \begin{bmatrix} 0.0128 & 0.0265 & -0.1475 \\ 0.1114 & 0.2304 & -1.2820 \\ -0.4178 & -0.8642 & 4.8084 \\ -0.3270 & -0.6763 & 3.7631 \end{bmatrix} \tag{4.128}$$

In this second rank-1 approximation $\boldsymbol{M}_2$ we capture Chandra's ratings

and movie types well, but for the science fiction movies and people the predictions are, not surprisingly, poor.

This leads us to consider the rank-2 approximation $\widehat{\boldsymbol{A}}(2)$ where we combine the first two rank-1 approximations

$$\widehat{\boldsymbol{A}}(2) = \boldsymbol{M}_1 + \boldsymbol{M}_2 \tag{4.129}$$

$$= \begin{bmatrix} 4.7801 & 4.2419 & 1.0244 \\ 5.2252 & 4.7522 & -0.0250 \\ 0.2493 & -0.2743 & 4.9724 \\ 0.7495 & 0.2756 & 4.0278 \end{bmatrix} \tag{4.130}$$

$\widehat{\boldsymbol{A}}(2)$ is close to the original movie ratings table

$$\boldsymbol{A} = \begin{bmatrix} 5 & 4 & 1 \\ 5 & 5 & 0 \\ 0 & 0 & 5 \\ 1 & 0 & 4 \end{bmatrix} \tag{4.131}$$

and this suggests that we can ignore the third singular value (after all it is much smaller than the first two). We can interpret this as to imply that in the data table there really is no evidence of a third movie-theme-movie lovers category. This also means that the entire space of movie themes-movie lovers is spanned in our example by a two-dimensional space spanned by science fiction and French art house movies and lovers.

4.7 Matrix Phylogeny

In Chapter 2 and 3 we covered the basics of linear algebra and analytic geometry, in this chapter we now looked at fundamental characteristics and methods on matrices and linear mappings. We are depicting in Figure 4.12 the phylogenetic tree of relationships between different types of matrices (black arrows indicating "is a subset of") and the covered operations we can perform on them (in red). For example, we already learned in Chapter 2 about **square** matrices, which are a subset of **all (complex) matrices** (top level node in the tree). We will then learn here that we can compute a specific characteristic (**determinant**) in Section 4.1 that will inform us whether a square matrix has an associate **inverse matrix**, thus if it belongs to the class of non-singular, invertible matrices.

The word phylogenetic describes how we capture the relationships among individuals or groups and derived from the greek words for "tribe" and "source".

Going backward through the chapter, we start with the most general case of real matrices $\mathbb{R}^{n\times m}$ for which we can define a pseude-inverse to "invert" them, as well as perform **singular value decomposition (SVD)** (Theorem 4.22). This superset of matrices is divided into the square $\mathbb{R}^{n\times n}$ matrices for which we can define the characteristic feature of the **determinant** and the **trace** (Section 4.1).

Here the set of matrices splits in two: If the square $\mathbb{R}^{n\times n}$ matrix has n distinct eigenvalues (or equivalently n linearly independent eigenvectors) then the matrix is non-defective and a unique **diagonalisation/eigendecomposition** exists for these matrices (Theorem 4.11). In other cases we know that a multiplicity of eigenvalues may result (see Definitions 4.13 and 4.14).

Alternatively, if this square $\mathbb{R}^{n\times n}$ matrix has a non-zero determinant, than the matrix is non-singular, i.e. an inverse matrix exists (Theorem 4.1). Non-singular matrices are closed under addition and multiplication, have an identity element ($\boldsymbol{I}$) and an inverse element, thus they form a group.

Note, that non-singular and non-defective matrices are not identical sets, as for example a rotation matrix will be invertible (determinant is non-zero) but not diagonalizable in the real numbers (non-distinct real eigenvalues).

Let us follow the branch of non-defective square $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ matrices. $\boldsymbol{A}$ is normal if the condition $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\top$ holds. Moreover, if the more restrictive condition holds $\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{I}$, then the matrix is called orthogonal (see Definition 3.8) and is a subset of the non-singular (invertible) matrices and satisfy the very useful condition $\boldsymbol{A}^\top = \boldsymbol{A}^{-1}$. Orthogonal matrices are closed under addition and multiplication, have an identity element ($\boldsymbol{I}$) and an inverse element, thus they also form a group.

The normal matrices have a frequently encountered subset, the symmetric matrices $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ which satisfy $\boldsymbol{S} = \boldsymbol{S}^\top$. Symmetric matrices have only real eigenvalues. A subset of the symmetric matrices are the positive definite matrices $\boldsymbol{P}$ that satisfy the condition of $\boldsymbol{x}^\top\boldsymbol{P}\boldsymbol{x} > 0$, then a unique a unique **Cholesky decomposition** exists (Theorem 4.17). Positive definite matrices have only positive eigenvalues and are always invertible (i.e. have a non-zero determinant).

Another subset of the symmetric matrices are the **diagonal matrices $\boldsymbol{D}$** in which the entries outside the main diagonal are all zero. Diagonal matrices are closed under multiplication and addition, but do not necessarily form a group (this is only the case if all diagonal entries are non-zero so that the matrix is invertible). A prominent special case of the diagonal matrices is the identity matrix $\boldsymbol{I}$.

4.8 Further Reading

Most of the content in this chapter establishes underlying mathematics and connects them to methods for studying mappings, many of these underly machine learning at the level of underpinning software solutions and building blocks for almost all machine learning theory. Matrix characterization using determinants, eigenspectra and eigenspaces are fundamental features and conditions for categorizing and analyzing matrices, this extends to all forms of representations of data and mappings involving data,

as well as judging the numerical stability of computational operations on such matrices(Press et al., 2007).

Determinants are fundamental tools in order to invert matrices and compute eigenvalues "by hand", yet for almost all but the smallest instances computation by Gaussian elimination outperforms determinants (Press et al., 2007). Determinants remain however a powerful theoretical concept, e.g. to gain intuition about the orientation of a basis based on the sign of the determinant. Eigenvectors can be used to perform change of basis operations so as to transform complicated looking data into more meaningful orthogonal, features vectors. Similarly, matrix decomposition methods such as Cholesky decomposition reappear often when we have to compute or simulate random events (Rubinstein and Kroese, 2016).

Eigendecomposition is fundamental in enabling us to extract meaningful and interpretable information that characterizes linear mappings. Therefore, eigendecomposition underlies a general class of machine learning algorithms called *spectral methods* that perform eigendecomposition of a positive-definite kernel. These spectral decomposition methods encompass classical approaches to statistical data analysis, such as

- Principal Components Analysis (PCA (Pearson, 1901a), see also Chapter 10), in which a low-dimensional subspace that explains most of the variability in the data is sought.
- Fisher Discriminant Analysis, which aims to determine a separating hyperplane for data classification (Mika et al., 1999).
- Multidimensional Scaling (MDS) (Carroll and Chang, 1970).

The computational efficiency of these methods typically results from finding the best rank-k approximation to a symmetric, positive semidefinite matrix. More contemporary examples of spectral methods have different origins , but each of them requires the computation of the eigenvectors and eigenvalues of a positive-definite kernel, such as

- Isomap (Tenenbaum et al., 2000),
- Laplacian eigenmaps (Belkin and Niyogi, 2003),
- Hessian eigenmaps (Donoho and Grimes, 2003),
- Spectral clustering (Shi and Malik, 2000).

The core computations of these are generally underpinned by low-rank matrix approximation techniques (Belabbas and Wolfe, 2009), as we encountered here via the SVD.

The SVD allows us to discover some of the same kind of information as the eigendecomposition. However, the SVD is more generally applicable to non-square matrices, such as tables of data. These matrix factorisation methods become relevant whenever we want to identify heterogeneity in data when we want to perform data compression by approximation, e.g. instead of storing ($n \times m$ values just storing $(n + m) \times k$ values,

or when we want to perform data preprocessing, e.g. to decorrelate predictor variables of a design matrix (e.g. Ormoneit et al. (2001)). SVD is the basic two-dimensional version of a more general decomposition of data in, so called, tensors (Kolda and Bader, 2009). Tensors reflect higher-dimensional arrays and SVD-like and low-rank approximation s on tensors are for example the CP (Carroll and Chang, 1970) or Tucker Decomposition (Tucker, 1966).

The SVD low-rank approximation is frequently used in machine learning for both computational efficiency reasons. This is because it reduces the amount of memory and operations with non-zero multiplications we need to perform on potentially very large matrices of data (Trefethen and Bau III, 1997). Moreover, low-rank approximation is used to operate on matrices that may contain missing values as well as for purposes of lossy compression and dimensionality reduction (Moonen and De Moor, 1995; Markovsky, 2011).

Exercises

4.1 Compute the determinant using the Laplace expansion (using the the first row) and the Sarrus Rule for

$$\boldsymbol{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 0 & 2 & 4 \end{bmatrix} \tag{4.132}$$

4.2 Compute the following determinant efficiently.

$$\begin{bmatrix} 2 & 0 & 1 & 2 & 0 \\ 2 & -1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 1 & 2 \\ -2 & 0 & 2 & -1 & 2 \\ 2 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.133}$$

4.3 Let us compute the eigenspaces of $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} -2 & 2 \\ 2 & 1 \end{bmatrix}$

4.4 Compute the eigenspaces of

$$\boldsymbol{A} = \begin{bmatrix} 0 & -1 & 1 & 1 \\ -1 & 1 & -2 & 3 \\ 2 & -1 & 0 & 0 \\ 1 & -1 & 1 & 0 \end{bmatrix} \tag{4.134}$$

4.5 Diagonalizability of a matrix is unrelated to its invertibility. Determine for the following for matrices it if is diagonalizable and/or invertible $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

4.6 Find the SVD of the following matrix

$$\boldsymbol{A} = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \tag{4.135}$$

4.7 Let us find the singular value decomposition of

$$\boldsymbol{A} = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} . \tag{4.136}$$

4.8 Find the best rank-1 approximation of

$$\boldsymbol{A} = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \tag{4.137}$$

5

Vector Calculus

Many algorithms in machine learning are inherently based on optimizing an objective function with respect to a set of desired model parameters that control how well a model explains the data: Finding good parameters can be phrased as an optimization problem. Examples include linear regression (see Chapter 9), where we look at curve-fitting problems, and we optimize linear weight parameters to maximize the likelihood; neural-network auto-encoders for dimensionality reduction and data compression, where the parameters are the weights and biases of each layer, and where we minimize a reconstruction error by repeated application of the chain-rule; Gaussian mixture models (see Chapter 11) for modeling data distributions, where we optimize the location and shape parameters of each mixture component to maximize the likelihood of the model. Figure 5.1 illustrates some of these problems, which we typically solve by using optimization algorithms that exploit gradient information (first-order methods). Figure 5.2 gives an overview of how concepts in this chapter are related and how they are connected to other chapters of the book.

In this chapter, we will discuss how to compute gradients of functions, which is often essential to facilitate learning in machine learning models. Therefore, vector calculus is one of the fundamental mathematical tools we need in machine learning.

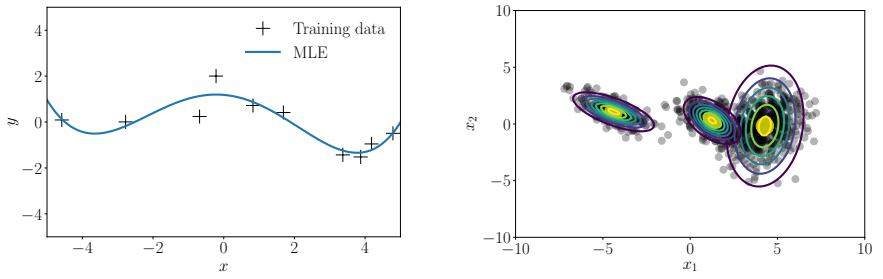


Figure 5.1 Vector calculus plays a central role in (a) regression (curve fitting) and (b) density estimation, i.e., modeling data distributions.

(a) Regression problem: Find parameters, such that the curve explains the observations (circles) well.

(b) Density estimation with a Gaussian mixture model: Find means and covariances, such that the data (dots) can be explained well.

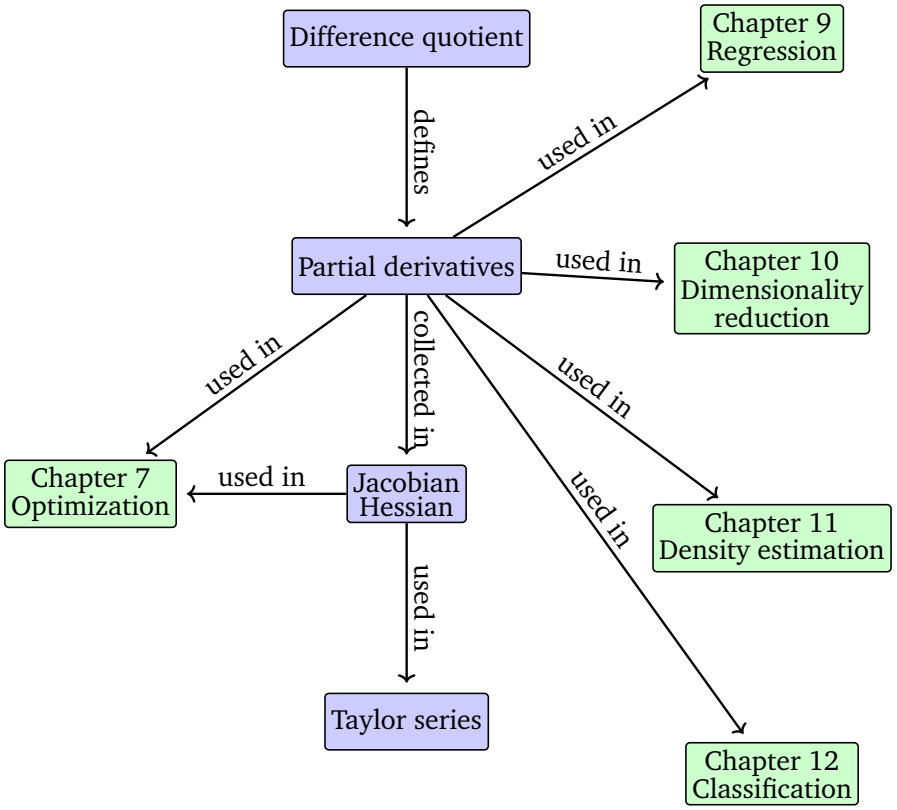


Figure 5.2 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

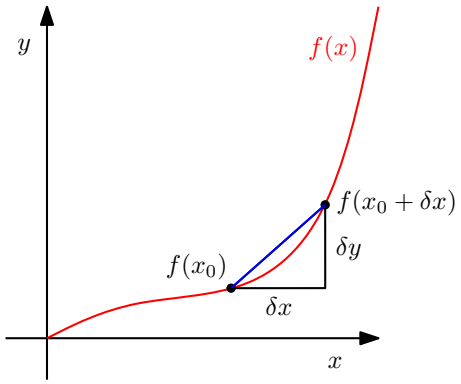


Figure 5.3 The average incline of a function f between x_0 and $x_0 + \delta x$ is the incline of the secant (blue) through $f(x_0)$ and $f(x_0 + \delta x)$ and given by $\delta y/\delta x$.

5.1 Differentiation of Univariate Functions

In the following, we briefly revisit differentiation of a univariate function, which we may already know from school. We start with the difference quotient of a univariate function $y = f(x)$, $x, y \in \mathbb{R}$, which we will subsequently use to define derivatives.

Definition 5.1 (Difference Quotient)**.** The *difference quotient* difference quotient

$$\frac{\delta y}{\delta x} := \frac{f(x+\delta x) - f(x)}{\delta x} \tag{5.1}$$

computes the slope of the secant line through two points on the graph of f. In Figure 5.3 these are the points with x-coordinates x_0 and $x_0 + \delta x$.

The difference quotient can also be considered the average slope of f between x and $x + \delta x$ if we assume a f to be a linear function. In the limit for $\delta x \to 0$, we obtain the tangent of f at x, if f is differentiable. The tangent is then the derivative of f at x.

derivative

Definition 5.2 (Derivative)**.** More formally, for $h > 0$ the *derivative* of f at x is defined as the limit

$$\frac{\mathrm{d}f}{\mathrm{d}x} := \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}\,, \tag{5.2}$$

and the secant in Figure 5.3 becomes a tangent.

Example 5.1 (Derivative of a Polynomial)
We want to compute the derivative of $f(x) = x^n, n \in \mathbb{N}$. We may already know that the answer will be nx^{n-1}, but we want to derive this result using the definition of the derivative as the limit of the difference quotient.

Using the definition of the derivative in (5.2) we obtain

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \tag{5.3}$$

$$= \lim_{h\to 0} \frac{(x+h)^n - x^n}{h} \tag{5.4}$$

$$= \lim_{h\to 0} \frac{\sum_{i=0}^{n} \binom{n}{i} x^{n-i}h^i - x^n}{h}\,. \tag{5.5}$$

We see that $x^n = \binom{n}{0} x^{n-0}h^0$. By starting the sum at 1 the x^n-term cancels, and we obtain

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{h\to 0} \frac{\sum_{i=1}^{n} \binom{n}{i} x^{n-i}h^i}{h} \tag{5.6}$$

$$= \lim_{h\to 0} \sum_{i=1}^{n} \binom{n}{i} x^{n-i}h^{i-1} \tag{5.7}$$

$$= \lim_{h\to 0} \binom{n}{1} x^{n-1} + \underbrace{\sum_{i=2}^{n} \binom{n}{i} x^{n-i}h^{i-1}}_{\to 0 \text{ as } h\to 0} \tag{5.8}$$

$$= \frac{n!}{1!(n-1)!} x^{n-1} = nx^{n-1}\,. \tag{5.9}$$

5.1.1 Taylor Series

The Taylor series is a representation of a function f as an infinite sum of terms. These terms are determined using derivatives of f evaluated at x_0.

Definition 5.3 (Taylor Polynomial)**.** The *Taylor polynomial* of degree n of $f : \mathbb{R} \to \mathbb{R}$ at x_0 is defined as

Taylor polynomial

$$T_n(x) := \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \,, \tag{5.10}$$

where $f^{(k)}(x_0)$ is the kth derivative of f at x_0 (which we assume exists) and $\frac{f^{(k)}(x_0)}{k!}$ are the coefficients of the polynomial.

Definition 5.4 (Taylor Series)**.** For a smooth function $f \in \mathcal{C}^\infty$, $f : \mathbb{R} \to \mathbb{R}$, the *Taylor series* of f at x_0 is defined as

Taylor series

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \,. \tag{5.11}$$

For $x_0 = 0$, we obtain the *Maclaurin series* as a special instance of the Taylor series. If $f(x) = T_\infty(x)$ then f is called *analytic*.

$f \in \mathcal{C}^\infty$ means that f is continuously differentiable infinitely many times.
Maclaurin series
analytic

Remark. In general, a Taylor polynomial of degree n is an approximation of a function, which does not need to be a polynomial. The Taylor polynomial is similar to f in a neighborhood around x_0. However, a Taylor polynomial of degree n is an exact representation of a polynomial f of degree $k \leqslant n$ since all derivatives $f^{(i)}$, $i > k$ vanish. ♢

Example 5.2 (Taylor Polynomial)
We consider the polynomial

$$f(x) = x^4 \tag{5.12}$$

and seek the Taylor polynomial T_6, evaluated at $x_0 = 1$. We start by computing the coefficients $f^{(k)}(1)$ for $k = 0, \ldots, 6$:

$$f(1) = 1 \tag{5.13}$$
$$f'(1) = 4 \tag{5.14}$$
$$f''(1) = 12 \tag{5.15}$$
$$f^{(3)}(1) = 24 \tag{5.16}$$
$$f^{(4)}(1) = 24 \tag{5.17}$$
$$f^{(5)}(1) = 0 \tag{5.18}$$
$$f^{(6)}(1) = 0 \tag{5.19}$$

Therefore, the desired Taylor polynomial is

$$T_6(x) = \sum_{k=0}^{6} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \tag{5.20}$$

$$= 1 + 4(x-1) + 6(x-1)^2 + 4(x-1)^3 + (x-1)^4 + 0\,. \tag{5.21}$$

Multiplying out and re-arranging yields

$$T_6(x) = (1 - 4 + 6 - 4 + 1) + x(4 - 12 + 12 - 4) + x^2(6 - 12 + 6) + x^3(4 - 4) + x^4 \tag{5.22}$$

$$= x^4 = f(x)\,, \tag{5.23}$$

i.e., we obtain an exact representation of the original function.

Example 5.3 (Taylor Series)
Consider the function

$$f(x) = \sin(x) + \cos(x) \in \mathcal{C}^\infty\,. \tag{5.24}$$

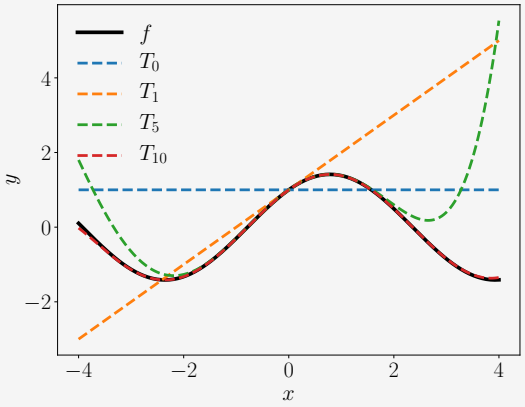


Figure 5.4 Taylor polynomials. The original function $f(x) = \sin(x) + \cos(x)$ (black, solid) is approximated by Taylor polynomials (dashed) around $x_0 = 0$. Higher-order Taylor polynomials approximate the function f better and more globally. T_{10} is already similar to f in $[-4, 4]$.

We seek a Taylor series expansion of f at $x_0 = 0$, which is the Maclaurin series expansion of f. We obtain the following derivatives:

$$f(0) = \sin(0) + \cos(0) = 1 \tag{5.25}$$

$$f'(0) = \cos(0) - \sin(0) = 1 \tag{5.26}$$

$$f''(0) = -\sin(0) - \cos(0) = -1 \tag{5.27}$$

$$f^{(3)}(0) = -\cos(0) + \sin(0) = -1 \tag{5.28}$$

$$f^{(4)}(0) = \sin(0) + \cos(0) = f(0) = 1 \tag{5.29}$$

$$\vdots$$

We can see a pattern here: The coefficients in our Taylor series are only ± 1 (since $\sin(0) = 0$), each of which occurs twice before switching to the other one. Furthermore, $f^{(k+4)}(0) = f^{(k)}(0)$.

Therefore, the full Taylor series expansion of f at $x_0 = 0$ is given by

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \tag{5.30}$$

$$= 1 + x - \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 - \cdots \tag{5.31}$$

$$= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 \mp \cdots + x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 \mp \cdots \tag{5.32}$$

$$= \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k)!}x^{2k} + \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k+1)!}x^{2k+1} \tag{5.33}$$

$$= \cos(x) + \sin(x)\,, \tag{5.34}$$

where we used the *power series representations*

power series representations

$$\cos(x) = \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k)!}x^{2k}\,, \tag{5.35}$$

$$\sin(x) = \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k+1)!}x^{2k+1}\,. \tag{5.36}$$

Figure 5.4 shows the corresponding first Taylor polynomials T_n for $n = 0, 1, 5, 10$.

5.1.2 Differentiation Rules

In the following, we briefly state basic differentiation rules, where we denote the derivative of f by f'.

Product Rule: $$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x) \tag{5.37}$$

Quotient Rule: $$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2} \tag{5.38}$$

Sum Rule: $$(f(x) + g(x))' = f'(x) + g'(x) \tag{5.39}$$

Chain Rule: $$\big(g(f(x))\big)' = (g \circ f)'(x) = g'(f(x))f'(x) \tag{5.40}$$

Here, $g \circ f$ is a function composition $x \mapsto f(x) \mapsto g(f(x))$.

Example 5.4 (Chain rule)
Let us compute the derivative of the function $h(x) = (2x + 1)^4$ using the

chain rule. With

$$h(x) = (2x+1)^4 = g(f(x))\,, \tag{5.41}$$
$$f(x) = 2x+1\,, \tag{5.42}$$
$$g(f) = f^4 \tag{5.43}$$

we obtain the derivatives of f and g as

$$f'(x) = 2\,, \tag{5.44}$$
$$g'(f) = 4f^3\,, \tag{5.45}$$

such that the derivative of h is given as

$$h'(x) = g'(f)f'(x) = (4f^3)\cdot 2 \overset{(5.42)}{=} 4(2x+1)^3 \cdot 2 = 8(2x+1)^3\,, \tag{5.46}$$

where we used the chain rule, see (5.40), and substituted the definition of f in (5.42) in $g'(f)$.

5.2 Partial Differentiation and Gradients

Differentiation as discussed in Section 5.1 applies to functions f of a scalar variable $x \in \mathbb{R}$. In the following, we consider the general case where the function f depends on one or more variables $\boldsymbol{x} \in \mathbb{R}^n$, e.g., $f(\boldsymbol{x}) = f(x_1, x_2)$. The generalization of the derivative to functions of several variables is the *gradient*.

We find the gradient of the function f with respect to $\boldsymbol{x}$ by *varying one variable at a time* and keeping the others constant. The gradient is then the collection of these *partial derivatives*.

Definition 5.5 (Partial Derivative)**.** For a function $f : \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{x} \mapsto f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^n$ of n variables $x_1, \ldots, x_n$ we define the *partial derivatives* as

partial derivatives

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= \lim_{h\to 0}\frac{f(x_1+h, x_2, \ldots, x_n) - f(\boldsymbol{x})}{h}\\ &\vdots\\ \frac{\partial f}{\partial x_n} &= \lim_{h\to 0}\frac{f(x_1, \ldots, x_{n-1}, x_n+h) - f(\boldsymbol{x})}{h}\end{aligned} \tag{5.47}$$

and collect them in the row vector

$$\nabla_{\boldsymbol{x}} f = \operatorname{grad} f = \frac{df}{d\boldsymbol{x}} = \begin{bmatrix}\frac{\partial f(\boldsymbol{x})}{\partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n}\end{bmatrix} \in \mathbb{R}^{1\times n}\,, \tag{5.48}$$

where n is the number of variables and 1 is the dimension of the image/range of f. Here, we defined the column vector $\boldsymbol{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$. The row vector in (5.48) is called the *gradient* of f or the *Jacobian* and is the generalization of the derivative from Section 5.1.

gradient
Jacobian

Remark. This definition of the Jacobian is a special case of the general definition of the Jacobian for vector-valued functions as the collection of partial derivatives. We will get back to this in Section 5.3. ♢

Example 5.5 (Partial Derivatives using the Chain Rule)
For $f(x, y) = (x + 2y^3)^2$, we obtain the partial derivatives

$$\frac{\partial f(x,y)}{\partial x} = 2(x+2y^3)\frac{\partial}{\partial x}(x+2y^3) = 2(x+2y^3)\,, \tag{5.49}$$

$$\frac{\partial f(x,y)}{\partial y} = 2(x+2y^3)\frac{\partial}{\partial y}(x+2y^3) = 12(x+2y^3)y^2\,. \tag{5.50}$$

where we used the chain rule (5.40) to compute the partial derivatives.

We can use results from scalar differentiation: Each partial derivative is a derivative with respect to a scalar.

Remark (Gradient as a Row Vector). It is not uncommon in the literature to define the gradient vector as a column vector, following the convention that vectors are generally column vectors. The reason why we define the gradient vector as a row vector is twofold: First, we can consistently generalize the gradient to a setting where $f : \mathbb{R}^n \to \mathbb{R}^m$ no longer maps onto the real line (then the gradient becomes a matrix). Second, we can immediately apply the multi-variate chain-rule without paying attention to the dimension of the gradient. We will discuss both points later. ♢

Example 5.6 (Gradient)
For $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$, the partial derivatives (i.e., the derivatives of f with respect to x_1 and x_2) are

$$\frac{\partial f(x_1,x_2)}{\partial x_1} = 2x_1x_2 + x_2^3 \tag{5.51}$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} = x_1^2 + 3x_1x_2^2 \tag{5.52}$$

and the gradient is then

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix}\frac{\partial f(x_1,x_2)}{\partial x_1} & \frac{\partial f(x_1,x_2)}{\partial x_2}\end{bmatrix} = \begin{bmatrix}2x_1x_2 + x_2^3 & x_1^2 + 3x_1x_2^2\end{bmatrix} \in \mathbb{R}^{1\times 2}\,. \tag{5.53}$$

5.2.1 Basic Rules of Partial Differentiation

In the multivariate case, where $\boldsymbol{x} \in \mathbb{R}^n$, the basic differentiation rules that we know from school (e.g., sum rule, product rule, chain rule; see also Section 5.1.2) still apply. However, when we compute derivatives with respect to vectors $\boldsymbol{x} \in \mathbb{R}^n$ we need to pay attention: Our gradients now

Product rule: $(fg)' = f'g + fg'$,
Sum rule: $(f+g)' = f' + g'$,
Chain rule: $(g(f))' = g'(f)f'$

involve vectors and matrices, and matrix multiplication is no longer commutative (see Section 2.2.1), i.e., the order matters.

Here are the general product rule, sum rule and chain rule:

$$\text{Product Rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}\big(f(\boldsymbol{x})g(\boldsymbol{x})\big) = \frac{\partial f}{\partial \boldsymbol{x}}g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}} \tag{5.54}$$

$$\text{Sum Rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}\big(f(\boldsymbol{x}) + g(\boldsymbol{x})\big) = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{x}} \tag{5.55}$$

$$\text{Chain Rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\big(g(f(\boldsymbol{x}))\big) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}} \tag{5.56}$$

This is only an intuition, but not mathematically correct since the partial derivative is not a fraction.

Let us have a closer look at the chain rule. The chain rule (5.56) resembles to some degree the rules for matrix multiplication where we said that neighboring dimensions have to match for matrix multiplication to be defined, see Section 2.2.1. If we go from left to right, the chain rule exhibits similar properties: ∂f shows up in the "denominator" of the first factor and in the "numerator" of the second factor. If we multiply the factors together, multiplication is defined, i.e., the dimensions of ∂f match, and ∂f "cancels", such that $\partial g/\partial \boldsymbol{x}$ remains.

5.2.2 Chain Rule

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x_1, x_2. Furthermore, $x_1(t)$ and $x_2(t)$ are themselves functions of t. To compute the gradient of f with respect to t, we need to apply the chain rule (5.56) for multivariate functions as

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{5.57}$$

where d denotes the gradient and ∂ partial derivatives.

Example 5.7
Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin t$ and $x_2 = \cos t$, then

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{5.58}$$

$$= 2\sin t\frac{\partial \sin t}{\partial t} + 2\frac{\partial \cos t}{\partial t} \tag{5.59}$$

$$= 2\sin t\cos t - 2\sin t = 2\sin t(\cos t - 1) \tag{5.60}$$

is the corresponding derivative of f with respect to t.

If $f(x_1, x_2)$ is a function of x_1 and x_2, where $x_1(s, t)$ and $x_2(s, t)$ are themselves functions of two variables s and t, the chain rule yields the

partial derivatives

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s}\,, \tag{5.61}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}\,, \tag{5.62}$$

and the gradient is obtained by the matrix multiplication

$$\frac{\mathrm{d}f}{\mathrm{d}(s,t)} = \frac{\partial f}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial (s,t)} = \underbrace{\begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}}_{=\frac{\partial f}{\partial \boldsymbol{x}}} \underbrace{\begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t}\\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\end{bmatrix}}_{=\frac{\partial \boldsymbol{x}}{\partial (s,t)}}\,. \tag{5.63}$$

This compact way of writing the chain rule as a matrix multiplication only makes sense if the gradient is defined as a row vector. Otherwise, we will need to start transposing gradients for the matrix dimensions to match. This may still be straightforward as long as the gradient is a vector or a matrix; however, when the gradient becomes a tensor (we will discuss this in the following), the transpose is no longer a triviality.

The chain rule can be written as a matrix multiplication.

Remark (Verifying the Correctness of a Gradient Implementation). The definition of the partial derivatives as the limit of the corresponding difference quotient, see (5.47), can be exploited when numerically checking the correctness of gradients in computer programs: When we compute gradients and implement them, we can use finite differences to numerically test our computation and implementation: We choose the value h to be small (e.g., $h = 10^{-4}$) and compare the finite-difference approximation from (5.47) with our (analytic) implementation of the gradient. If the error is small, our gradient implementation is probably correct. "Small" could mean that $\sqrt{\frac{\sum_i (dh_i - df_i)^2}{\sum_i (dh_i + df_i)^2}} < 10^{-6}$, where dh_i is the finite-difference approximation and df_i is the analytic gradient of f with respect to the ith variable x_i. ♢

Gradient checking

5.3 Gradients of Vector-Valued Functions

Thus far, we discussed partial derivatives and gradients of functions $f : \mathbb{R}^n \to \mathbb{R}$ mapping to the real numbers. In the following, we will generalize the concept of the gradient to vector-valued functions (vector fields) $f : \mathbb{R}^n \to \mathbb{R}^m$, where $n, m \geqslant 1$.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $\boldsymbol{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$, the corresponding vector of function values is given as

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^m\,. \tag{5.64}$$

Writing the vector-valued function in this way allows us to view a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ as a vector of functions $[f_1, \dots, f_m]^\top$, $f_i : \mathbb{R}^n \to \mathbb{R}$ that map onto $\mathbb{R}$. The differentiation rules for every f_i are exactly the ones we discussed in Section 5.2.

partial derivative of a vector-valued function

Therefore, the *partial derivative of a vector-valued function* $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $x_i \in \mathbb{R}$, $i = 1, \dots n$, is given as the vector

$$\frac{\partial f}{\partial x_i} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{bmatrix} = \begin{bmatrix} \lim_{h\to 0} \frac{f_1(x_1,\dots,x_{i-1},x_i+h,x_{i+1},\dots x_n)-f_1(\boldsymbol{x})}{h} \\ \vdots \\ \lim_{h\to 0} \frac{f_m(x_1,\dots,x_{i-1},x_i+h,x_{i+1},\dots x_n)-f_m(\boldsymbol{x})}{h} \end{bmatrix} \in \mathbb{R}^m . \tag{5.65}$$

From (5.48), we know that we obtain the gradient of f with respect to a vector as the row vector of the partial derivatives. In (5.65), every partial derivative $\partial f/\partial x_i$ is a column vector. Therefore, we obtain the gradient of $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $\boldsymbol{x} \in \mathbb{R}^n$ by collecting these partial derivatives:

$$\frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}} = \begin{bmatrix} \boxed{\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1}} \cdots \boxed{\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n}} \end{bmatrix} = \begin{bmatrix} \boxed{\frac{\partial f_1(\boldsymbol{x})}{\partial x_1}} \cdots \boxed{\frac{\partial f_1(\boldsymbol{x})}{\partial x_n}} \\ \vdots \quad\quad \vdots \\ \boxed{\frac{\partial f_m(\boldsymbol{x})}{\partial x_1}} \cdots \boxed{\frac{\partial f_m(\boldsymbol{x})}{\partial x_n}} \end{bmatrix} \in \mathbb{R}^{m\times n} . \tag{5.66}$$

Definition 5.6 (Jacobian)**.** The collection of all first-order partial derivatives of a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is called the *Jacobian*. The Jacobian $\boldsymbol{J}$ is an $m \times n$ matrix, which we define and arrange as follows:

Jacobian

The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a matrix of size $m \times n$.

$$\boldsymbol{J} = \nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \tag{5.67}$$

$$= \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix} , \tag{5.68}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} , \quad J(i,j) = \frac{\partial f_i}{\partial x_j} . \tag{5.69}$$

In particular, a function $f : \mathbb{R}^n \to \mathbb{R}^1$, which maps a vector $\boldsymbol{x} \in \mathbb{R}^n$ onto a scalar (e.g., $f(\boldsymbol{x}) = \sum_{i=1}^n x_i$), possesses a Jacobian that is a row vector (matrix of dimension $1 \times n$), see (5.48).

Remark. (Variable Transformation and Jacobian Determinant)

In Section 4.1, we saw that the determinant can be used to compute

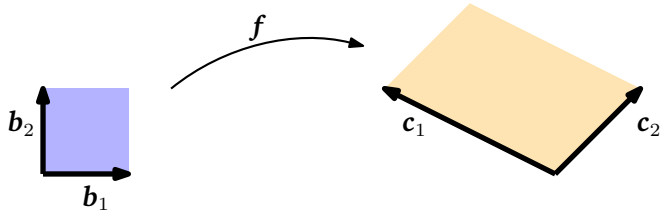


Figure 5.5 The determinant of the Jacobian of $\boldsymbol{f}$ can be used to compute the magnifier between the blue and orange area.

the area of a parallelogram. If we are given two vectors $\boldsymbol{b}_1 = [1, 0]^\top$, $\boldsymbol{b}_2 = [0, 1]^\top$ as the sides of the unit square (blue, see Figure 5.5), the area of this square is

$$\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1\,. \tag{5.70}$$

If we now take a parallelogram with the sides $\boldsymbol{c}_1 = [-2, 1]^\top$, $\boldsymbol{c}_2 = [1, 1]^\top$ (orange in Figure 5.5) its area is given as the absolute value of the determinant

$$\left|\det\left(\begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}\right)\right| = |-3| = 3\,, \tag{5.71}$$

i.e., the area of this is exactly 3 times the area of the unit square. We can find this scaling factor by finding a mapping that transforms the unit square into the other square. In linear algebra terms, we effectively perform a variable transformation from $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ to $(\boldsymbol{c}_1, \boldsymbol{c}_2)$. In our case, the mapping is linear and the absolute value of the determinant of this mapping gives us exactly the scaling factor we are looking for.

We will describe two approaches to identify this mapping. First, we exploit the fact that the mapping is linear so that we can use the tools from Chapter 2 to identify this mapping. Second, we will find the mapping using partial derivatives using the tools we have been discussing in this chapter.

Approach 1 To get started with the linear algebra approach, we identify both $\{\boldsymbol{b}_1, \boldsymbol{b}_2\}$ and $\{\boldsymbol{c}_1, \boldsymbol{c}_2\}$ as bases of $\mathbb{R}^2$ (see Section 2.6.1 for a recap). What we effectively perform is a change of basis from $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ to $(\boldsymbol{c}_1, \boldsymbol{c}_2)$, and we are looking for the transformation matrix that implements the basis change. Using results from Section 2.7.2, we identify the desired basis change matrix as

$$\boldsymbol{J} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}\,, \tag{5.72}$$

such that $\boldsymbol{J}\boldsymbol{b}_1 = \boldsymbol{c}_1$ and $\boldsymbol{J}\boldsymbol{b}_2 = \boldsymbol{c}_2$. The absolute value of the determinant of $\boldsymbol{J}$, which yields the scaling factor we are looking for, is given as $|\det(\boldsymbol{J})| = 3$, i.e., the area of the square spanned by $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ is three times greater than the area spanned by $(\boldsymbol{b}_1, \boldsymbol{b}_2)$.

Approach 2 The linear algebra approach works nicely for linear

transformations; for nonlinear transformations (which become relevant in Chapter 6), we can follow a more general approach using partial derivatives.

For this approach, we consider a function $\boldsymbol{f} : \mathbb{R}^2 \to \mathbb{R}^2$ that performs a variable transformation. In our example, $\boldsymbol{f}$ maps the coordinate representation of any vector $\boldsymbol{x} \in \mathbb{R}^2$ with respect to $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ onto the coordinate representation $\boldsymbol{y} \in \mathbb{R}^2$ with respect to $(\boldsymbol{c}_1, \boldsymbol{c}_2)$. We want to identify the mapping so that we can compute how an area (or volume) changes when it is being transformed by $\boldsymbol{f}$. For this we need to find out how $\boldsymbol{f}(\boldsymbol{x})$ changes if we modify $\boldsymbol{x}$ a bit. This question is exactly answered by the Jacobian matrix $\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}} \in \mathbb{R}^{2\times 2}$. Since we can write

$$y_1 = -2x_1 + x_2 \tag{5.73}$$
$$y_2 = x_1 + x_2 \tag{5.74}$$

we obtain the functional relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$, which allows us to get the partial derivatives

$$\frac{\partial y_1}{\partial x_1} = -2\,, \quad \frac{\partial y_1}{\partial x_2} = 1\,, \quad \frac{\partial y_2}{\partial x_1} = 1\,, \quad \frac{\partial y_2}{\partial x_2} = 1 \tag{5.75}$$

and compose the Jacobian as

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}\,. \tag{5.76}$$

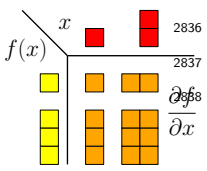
Geometrically, the Jacobian determinant gives the magnification/scaling factor when we transform an area or volume.
Jacobian determinant

The Jacobian represents the coordinate transformation we are looking for and is exact if the coordinate transformation is linear (as in our case), and (5.76) recovers exactly the basis change matrix in (5.72). If the coordinate transformation is nonlinear, the Jacobian approximates this nonlinear transformation locally with a linear one. The absolute value of the *Jacobian determinant* $|\det(\boldsymbol{J})|$ is the factor areas or volumes are scaled by when coordinates are transformed. In our case, we obtain $|\det(\boldsymbol{J})| = 3$.

The Jacobian determinant and variable transformations will become relevant in Section 6.5 when we transform random variables and probability distributions. These transformations are extremely relevant in machine learning in the context of training deep neural networks using the *reparametrization trick*, also called *infinite perturbation analysis*.

♢

Figure 5.6 Overview of the dimensionality of (partial) derivatives.



Throughout this chapter, we have encountered derivatives of functions. Figure 5.6 summarizes the dimensions of those gradients. If $f : \mathbb{R} \to \mathbb{R}$ the gradient is simply a scalar (top-left entry). For $f : \mathbb{R}^D \to \mathbb{R}$ the gradient is a $1 \times D$ row vector (to-right entry). For $f : \mathbb{R} \to \mathbb{R}^E$, the gradient is an $E \times 1$ column vector, and for $f : \mathbb{R}^D \to \mathbb{R}^E$ the gradient is an $E \times D$ matrix.

Example 5.8 (Gradient of a Vector-Valued Function)
We are given

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}\,, \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M\times N}, \quad \boldsymbol{x} \in \mathbb{R}^N\,.$$

To compute the gradient $d\boldsymbol{f}/d\boldsymbol{x}$ we first determine the dimension of $d\boldsymbol{f}/d\boldsymbol{x}$: Since $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, it follows that $d\boldsymbol{f}/d\boldsymbol{x} \in \mathbb{R}^{M\times N}$. Second, to compute the gradient we determine the partial derivatives of f with respect to every x_j:

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{N} A_{ij}x_j \implies \frac{\partial f_i}{\partial x_j} = A_{ij} \tag{5.77}$$

Finally, we collect the partial derivatives in the Jacobian and obtain the gradient as

$$\frac{d\boldsymbol{f}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} = \boldsymbol{A} \in \mathbb{R}^{M\times N}\,. \tag{5.78}$$

Example 5.9 (Chain Rule)
Consider the function $h : \mathbb{R} \to \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$f : \mathbb{R}^2 \to \mathbb{R} \tag{5.79}$$
$$g : \mathbb{R} \to \mathbb{R}^2 \tag{5.80}$$
$$f(\boldsymbol{x}) = \exp(x_1x_2^2)\,, \tag{5.81}$$
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = g(t) = \begin{bmatrix} t\cos t \\ t\sin t \end{bmatrix} \tag{5.82}$$

and compute the gradient of h with respect to t. Since $f : \mathbb{R}^2 \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}^2$ we note that

$$\frac{\partial f}{\partial \boldsymbol{x}} \in \mathbb{R}^{1\times 2}\,, \quad \frac{\partial g}{\partial t} \in \mathbb{R}^{2\times 1}\,. \tag{5.83}$$

The desired gradient is computed by applying the chain-rule:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{\partial f}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial t} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \tag{5.84}$$

$$= \begin{bmatrix} \exp(x_1x_2^2)x_2^2 & 2\exp(x_1x_2^2)x_1x_2 \end{bmatrix} \begin{bmatrix} \cos t - t\sin t \\ \sin t + t\cos t \end{bmatrix} \tag{5.85}$$

$$= \exp(x_1x_2^2)\big(x_2^2(\cos t - t\sin t) + 2x_1x_2(\sin t + t\cos t)\big)\,, \tag{5.86}$$

where $x_1 = t\cos t$ and $x_2 = t\sin t$, see (5.82).

Example 5.10 (Gradient of a Least-Squared Loss in a Linear Model)
Let us consider the linear model

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\theta}\,, \tag{5.87}$$

where $\boldsymbol{\theta} \in \mathbb{R}^D$ is a parameter vector, $\boldsymbol{\Phi} \in \mathbb{R}^{N\times D}$ are input features and $\boldsymbol{y} \in \mathbb{R}^N$ are the corresponding observations. We define the functions

$$L(\boldsymbol{e}) := \|\boldsymbol{e}\|^2\,, \tag{5.88}$$

$$\boldsymbol{e}(\boldsymbol{\theta}) := \boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\,. \tag{5.89}$$

We will discuss this model in much more detail in Chapter 9 in the context of linear regression, where we need derivatives of the least-squares loss L with respect to the parameters $\boldsymbol{\theta}$.

We seek $\frac{\partial L}{\partial \boldsymbol{\theta}}$, and we will use the chain rule for this purpose. L is called a *least-squares loss* function.

Before we start our calculation, we determine the dimensionality of the gradient as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1\times D}\,. \tag{5.90}$$

The chain rule allows us to compute the gradient as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{e}}\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}}\,, \tag{5.91}$$

```
dLdtheta =
np.einsum(
'n,nd',
dLde,dedtheta)
```

where the dth element is given by

$$\frac{\partial L}{\partial \boldsymbol{\theta}}[1,d] = \sum_{n=1}^{N}\frac{\partial L}{\partial \boldsymbol{e}}[n]\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}}[n,d]\,. \tag{5.92}$$

We know that $\|\boldsymbol{e}\|^2 = \boldsymbol{e}^\top\boldsymbol{e}$ (see Section 3.2) and determine

$$\frac{\partial L}{\partial \boldsymbol{e}} = 2\boldsymbol{e}^\top \in \mathbb{R}^{1\times N}\,. \tag{5.93}$$

Furthermore, we obtain

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}} = -\boldsymbol{\Phi} \in \mathbb{R}^{N\times D}\,, \tag{5.94}$$

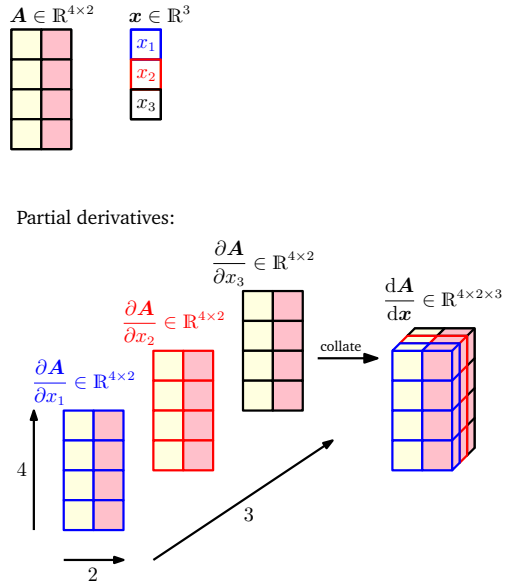
such that our desired derivative is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -2\boldsymbol{e}^\top\boldsymbol{\Phi} \overset{(5.89)}{=} -\underbrace{2(\boldsymbol{y}^\top - \boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top)}_{1\times N}\underbrace{\boldsymbol{\Phi}}_{N\times D} \in \mathbb{R}^{1\times D}\,. \tag{5.95}$$

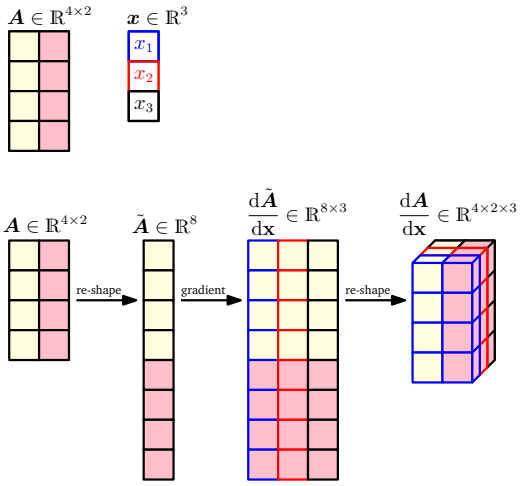
Remark. We would have obtained the same result without using the chain rule by immediately looking at the function

$$L_2(\boldsymbol{\theta}) := \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2 = (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})\,. \tag{5.96}$$

This approach is still practical for simple functions like L_2 but becomes impractical for deep function compositions. ♢



(a) Approach 1: We compute the partial derivative $\frac{\partial \boldsymbol{A}}{\partial x_1}, \frac{\partial \boldsymbol{A}}{\partial x_2}, \frac{\partial \boldsymbol{A}}{\partial x_3}$, each of which is a 4×2 matrix, and collate them in a $4 \times 2 \times 3$ tensor.



(b) Approach 2: We re-shape (flatten) $\boldsymbol{A} \in \mathbb{R}^{4\times 2}$ into a vector $\tilde{\boldsymbol{A}} \in \mathbb{R}^8$. Then, we compute the gradient $\frac{\mathrm{d}\tilde{\boldsymbol{A}}}{\mathrm{d}\boldsymbol{x}} \in \mathbb{R}^{8\times 3}$. We obtain the gradient tensor by re-shaping this gradient as illustrated above.

Figure 5.7 Visualization of gradient computation of a matrix with respect to a vector. We are interested in computing the gradient of $\boldsymbol{A} \in \mathbb{R}^{4\times 2}$ with respect to a vector $\boldsymbol{x} \in \mathbb{R}^3$. We know that gradient $\frac{\mathrm{d}\boldsymbol{A}}{\mathrm{d}\boldsymbol{x}} \in \mathbb{R}^{4\times 2\times 3}$. We follow two equivalent approaches to arrive there: (a) Collating partial derivatives into a Jacobian tensor; (b) Flattening of the matrix into a vector, computing the Jacobian matrix, re-shaping into a Jacobian tensor.

5.4 Gradients of Matrices

We will encounter situations where we need to take gradients of matrices with respect to vectors (or other matrices), which results in a multi-dimensional tensor. For example, if we compute the gradient of an $m \times n$

matrix with respect to a $p \times q$ matrix, the resulting Jacobian would be $(p \times q) \times (m \times n)$, i.e., a four-dimensional tensor (or array).

Since matrices represent linear mappings, we can exploit the fact that there is a vector-space isomorphism (linear, invertible mapping) between the space $\mathbb{R}^{m \times n}$ of $m \times n$ matrices and the space $\mathbb{R}^{mn}$ of mn vectors. Therefore, we can re-shape our matrices into vectors of lengths mn and pq, respectively. The gradient using these mn vectors results in a Jacobian of size $pq \times mn$. Figure 5.7 visualizes both approaches. In practical applications, it is often desirable to re-shape the matrix into a vector and continue working with this Jacobian matrix: The chain rule (5.56) boils down to simple matrix multiplication, whereas in the case of a Jacobian tensor, we will need to pay more attention to what dimensions we need to sum out.

Matrices can be transformed into vectors by stacking the columns of the matrix ("flattening").

Example 5.11 (Gradient of Vectors with Respect to Matrices)
Let us consider the following example, where

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{x}\,, \quad \boldsymbol{f} \in \mathbb{R}^M, \boldsymbol{A} \in \mathbb{R}^{M \times N}, \boldsymbol{x} \in \mathbb{R}^N \tag{5.97}$$

and where we seek the gradient $d\boldsymbol{f}/d\boldsymbol{A}$. Let us start again by determining the dimension of the gradient as

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} \in \mathbb{R}^{M \times (M \times N)}\,. \tag{5.98}$$

By definition, the gradient is the collection of the partial derivatives:

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \boldsymbol{A}} \end{bmatrix}, \quad \frac{\partial f_i}{\partial \boldsymbol{A}} \in \mathbb{R}^{1 \times (M \times N)}\,. \tag{5.99}$$

To compute the partial derivatives, it will be helpful to explicitly write out the matrix vector multiplication:

$$f_i = \sum_{j=1}^{N} A_{ij} x_j, \quad i = 1, \ldots, M\,, \tag{5.100}$$

and the partial derivatives are then given as

$$\frac{\partial f_i}{\partial A_{iq}} = x_q\,. \tag{5.101}$$

This allows us to compute the partial derivatives of f_i with respect to a row of $\boldsymbol{A}$, which is given as

$$\frac{\partial f_i}{\partial A_{i,:}} = \boldsymbol{x}^\top \in \mathbb{R}^{1 \times 1 \times N}\,, \tag{5.102}$$

$$\frac{\partial f_i}{\partial A_{k \neq i,:}} = \boldsymbol{0}^\top \in \mathbb{R}^{1 \times 1 \times N} \tag{5.103}$$

where we have to pay attention to the correct dimensionality. Since f_i maps onto $\mathbb{R}$ and each row of $\boldsymbol{A}$ is of size $1 \times N$, we obtain a $1 \times 1 \times N$-sized tensor as the partial derivative of f_i with respect to a row of $\boldsymbol{A}$.

We stack the partial derivatives to obtain the desired gradient as

$$\frac{\partial f_i}{\partial \boldsymbol{A}} = \begin{bmatrix} \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \\ \boldsymbol{x}^\top \\ \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \end{bmatrix} \in \mathbb{R}^{1 \times (M \times N)} . \tag{5.104}$$

Example 5.12 (Gradient of Matrices with Respect to Matrices)
Consider a matrix $\boldsymbol{L} \in \mathbb{R}^{m \times n}$ and $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{n \times n}$ with

$$\boldsymbol{f}(\boldsymbol{L}) = \boldsymbol{L}^\top \boldsymbol{L} =: \boldsymbol{K} \in \mathbb{R}^{n \times n} . \tag{5.105}$$

where we seek the gradient $d\boldsymbol{K}/d\boldsymbol{L}$. To solve this hard problem, let us first write down what we already know: We know that the gradient has the dimensions

$$\frac{d\boldsymbol{K}}{d\boldsymbol{L}} \in \mathbb{R}^{(n \times n) \times (m \times n)} , \tag{5.106}$$

which is a tensor. If we compute the partial derivative of f with respect to a single entry L_{ij}, $i, j \in \{1, \ldots, n\}$, of $\boldsymbol{L}$, we obtain an $n \times n$-matrix

$$\frac{\partial \boldsymbol{K}}{\partial L_{ij}} \in \mathbb{R}^{n \times n} . \tag{5.107}$$

Furthermore, we know that

$$\frac{dK_{pq}}{d\boldsymbol{L}} \in \mathbb{R}^{1 \times m \times n} \tag{5.108}$$

for $p, q = 1, \ldots, n$, where $K_{pq} = f_{pq}(\boldsymbol{L})$ is the (p, q)-th entry of $\boldsymbol{K} = f(\boldsymbol{L})$.

Denoting the i-th column of $\boldsymbol{L}$ by $\boldsymbol{l}_i$, we see that every entry of $\boldsymbol{K}$ is given by an inner product of two columns of $\boldsymbol{L}$, i.e.,

$$K_{pq} = \boldsymbol{l}_p^\top \boldsymbol{l}_q = \sum_{k=1}^{m} L_{kp} L_{kq} . \tag{5.109}$$

When we now compute the partial derivative $\frac{\partial K_{pq}}{\partial L_{ij}}$, we obtain

$$\frac{\partial K_{pq}}{\partial L_{ij}} = \sum_{k=1}^{m} \frac{\partial}{\partial L_{ij}} L_{kp} L_{kq} = \partial_{pqij} , \tag{5.110}$$

$$\partial_{pqij} = \begin{cases} L_{iq} & \text{if } j = p,\ p \neq q \\ L_{ip} & \text{if } j = q,\ p \neq q \\ 2L_{iq} & \text{if } j = p,\ p = q \\ 0 & \text{otherwise} \end{cases} \tag{5.111}$$

From (5.106), we know that the desired gradient has the dimension $(n \times n) \times (m \times n)$, and every single entry of this tensor is given by ∂_{pqij} in (5.111), where $p, q, j = 1, \ldots, n$ and $i = q, \ldots, m$.

5.5 Useful Identities for Computing Gradients

In the following, we list some useful gradients that are frequently required in a machine learning context (Petersen and Pedersen, 2012):

$$\frac{\partial}{\partial \boldsymbol{X}} f(\boldsymbol{X})^\top = \left(\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)^\top \tag{5.112}$$

$$\frac{\partial}{\partial \boldsymbol{X}} \mathrm{tr}(f(\boldsymbol{X})) = \mathrm{tr}\left(\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{5.113}$$

$$\frac{\partial}{\partial \boldsymbol{X}} \det(f(\boldsymbol{X})) = \det(f(\boldsymbol{X}))\mathrm{tr}\left(f^{-1}(\boldsymbol{X})\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{5.114}$$

$$\frac{\partial}{\partial \boldsymbol{X}} f^{-1}(\boldsymbol{X}) = -f^{-1}(\boldsymbol{X})\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} f^{-1}(\boldsymbol{X}) \tag{5.115}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}^{-1} \boldsymbol{b}}{\partial \boldsymbol{X}} = -(\boldsymbol{X}^{-1})^\top \boldsymbol{a}\boldsymbol{b}^\top (\boldsymbol{X}^{-1})^\top \tag{5.116}$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{5.117}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{5.118}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X} \boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^\top \tag{5.119}$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^\top (\boldsymbol{B} + \boldsymbol{B}^\top) \tag{5.120}$$

$$\frac{\partial}{\partial \boldsymbol{s}} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = -2(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}\boldsymbol{A} \quad \text{for symmetric } \boldsymbol{W} \tag{5.121}$$

Here, we use tr as the trace operator (see Definition 4.3) and $\det$ is the determinant (see Section 4.1).

5.6 Backpropagation and Automatic Differentiation

In many machine learning applications, we find good model parameters by performing gradient descent (Chapter 7), which relies on the fact that

we can compute the gradient of a learning objective with respect to the parameters of the model. For a given objective function, we can obtain the gradient with respect to the model parameters using calculus and applying the chain rule, see Section 5.2.2. We already had a taste in Section 5.3 when we looked at the gradient of a squared loss with respect to the parameters of a linear regression model.

Consider the function

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right) . \tag{5.122}$$

By application of the chain rule, and noting that differentiation is linear we compute the gradient

$$\begin{aligned}\frac{\mathrm{d}f}{\mathrm{d}x} &= \frac{2x + 2x\exp(x^2)}{2\sqrt{x^2 + \exp(x^2)}} - \sin\left(x^2 + \exp(x^2)\right)\left(2x + 2x\exp(x^2)\right) \\ &= 2x\left(\frac{1}{2\sqrt{x^2 + \exp(x^2)}} - \sin\left(x^2 + \exp(x^2)\right)\right)\left(1 + \exp(x^2)\right) .\end{aligned} \tag{5.123}$$

Writing out the gradient in this explicit way is often impractical since it often results in a very lengthy expression for a derivative. In practice, it means that, if we are not careful, the implementation of the gradient could be significantly more expensive than computing the function, which is an unnecessary overhead. For training deep neural network models, the *backpropagation* algorithm (Kelley, 1960; Bryson, 1961; Dreyfus, 1962; Rumelhart et al., 1986) is an efficient way to compute the gradient of an error function with respect to the parameters of the model.

backpropagation

5.6.1 Gradients in a Deep Network

In machine learning, the chain rule plays an important role when optimizing parameters of a hierarchical model (e.g., for maximum likelihood estimation). An area where the chain rule is used to an extreme is Deep Learning where the function value $\boldsymbol{y}$ is computed as a deep function composition

$$\boldsymbol{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\boldsymbol{x}) = f_K(f_{K-1}(\cdots(f_1(\boldsymbol{x}))\cdots)) , \tag{5.124}$$

where $\boldsymbol{x}$ are the inputs (e.g., images), $\boldsymbol{y}$ are the observations (e.g., class labels) and every function f_i, $i = 1, \ldots, K$ possesses its own parameters. In neural networks with multiple layers, we have functions $f_i(\boldsymbol{x}_{i-1}) = \sigma(\boldsymbol{A}_i\boldsymbol{x}_{i-1} + \boldsymbol{b}_i)$ in the ith layer. Here $\boldsymbol{x}_{i-1}$ is the output of layer $i-1$ and σ an activation function, such as the logistic sigmoid $\frac{1}{1+e^{-x}}$, $\tanh$ or a rectified linear unit (ReLU). In order to train these models, we require the gradient of a loss function L with respect to all model parameters $\boldsymbol{A}_j, \boldsymbol{b}_j$ for $j = 1, \ldots, K$. This also requires us to compute the gradient of L with respect to the inputs of each layer. For example, if we have inputs $\boldsymbol{x}$ and

We discuss the case where the activation functions are identical to unclutter notation.

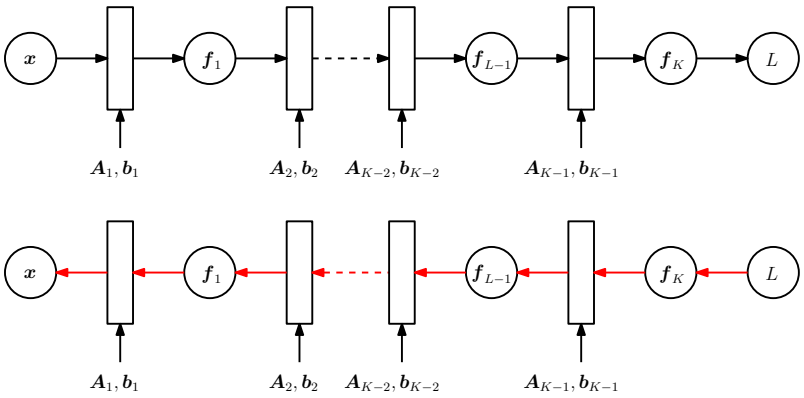


Figure 5.8 Forward pass in a multi-layer neural network to compute the loss L as a function of the inputs $\boldsymbol{x}$ and the parameters $\boldsymbol{A}_i$, $\boldsymbol{b}_i$.

Figure 5.9 Backward pass in a multi-layer neural network to compute the gradients of the loss function.

observations $\boldsymbol{y}$ and a network structure defined by

$$\boldsymbol{f}_0 := \boldsymbol{x} \tag{5.125}$$

$$\boldsymbol{f}_i := \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1}), \quad i = 1, \dots, K, \tag{5.126}$$

see also Figure 5.8 for a visualization, we may be interested in finding $\boldsymbol{A}_j, \boldsymbol{b}_j$ for $j = 0, \dots, K-1$, such that the squared loss

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_K(\boldsymbol{\theta}, \boldsymbol{x})\|^2 \tag{5.127}$$

is minimized, where $\boldsymbol{\theta} = \{\boldsymbol{A}_0, \boldsymbol{b}_0, \dots, \boldsymbol{A}_{K-1}, \boldsymbol{b}_{K-1}\}$.

To obtain the gradients with respect to the parameter set $\boldsymbol{\theta}$, we require the partial derivatives of L with respect to the parameters $\boldsymbol{\theta}_j = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$ of each layer $j = 0, \dots, K-1$. The chain rule allows us to determine the partial derivatives as

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}} \tag{5.128}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}} \tag{5.129}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}} \tag{5.130}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \boxed{\frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}} \tag{5.131}$$

The **orange** terms are partial derivatives of the output of a layer with respect to its inputs, whereas the **blue** terms are partial derivatives of the output of a layer with respect to its parameters. Assuming, we have already computed the partial derivatives $\partial L/\partial \boldsymbol{\theta}_{i+1}$, then most of the computation can be reused to compute $\partial L/\partial \boldsymbol{\theta}_i$. The additional terms that we need to compute are indicated by the boxes. Figure 5.9 visualizes that the gradients are passed backward through the network. A more

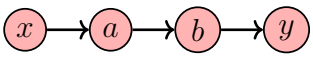


Figure 5.10 Simple graph illustrating the flow of data from x to y via some intermediate variables a, b.

in-depth discussion about gradients of neural networks can be found at `https://tinyurl.com/yalcxgtv`.

There are efficient ways of implementing this repeated application of the chain rule using *backpropagation* (Kelley, 1960; Bryson, 1961; Dreyfus, 1962; Rumelhart et al., 1986). A good discussion about backpropagation and the chain rule is available at `https://tinyurl.com/ycfm2yrw`.

backpropagation

5.6.2 Automatic Differentiation

It turns out that backpropagation is a special case of a general technique in numerical analysis called *automatic differentiation*. We can think of automatic differentation as a set of techniques to numerically (in contrast to symbolically) evaluate the exact (up to machine precision) gradient of a function by working with intermediate variables and applying the chain rule. Automatic differentiation applies a series of elementary arithmetic operations, e.g., addition and multiplication and elementary functions, e.g., $\sin, \cos, \exp, \log$. By applying the chain rule to these operations, the gradient of quite complicated functions can be computed automatically. Automatic differentiation applies to general computer programs and has forward and reverse modes.

Automatic differentiation is different from symbolic differentiation and numerical approximations of the gradient, e.g., by using finite differences.
automatic differentiation

Figure 5.10 shows a simple graph representing the data flow from inputs x to outputs y via some intermediate variables a, b. If we were to compute the derivative dy/dx, we would apply the chain rule and obtain

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}b}\frac{\mathrm{d}b}{\mathrm{d}a}\frac{\mathrm{d}a}{\mathrm{d}x}\,. \tag{5.132}$$

Intuitively, the forward and reverse mode differ in the order of multiplication. Due to the associativity of matrix multiplication we can choose between

In the general case, we work with Jacobians, which can be vectors, matrices or tensors.

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \left(\frac{\mathrm{d}y}{\mathrm{d}b}\frac{\mathrm{d}b}{\mathrm{d}a}\right)\frac{\mathrm{d}a}{\mathrm{d}x}\,, \tag{5.133}$$

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}b}\left(\frac{\mathrm{d}b}{\mathrm{d}a}\frac{\mathrm{d}a}{\mathrm{d}x}\right)\,. \tag{5.134}$$

Equation (5.133) would be the *reverse mode* because gradients are propagated backward through the graph, i.e., reverse to the data flow. Equation (5.134) would be the *forward mode*, where the gradients flow with the data from left to right through the graph.

reverse mode

forward mode

In the following, we will focus on reverse mode automatic differentiation, which is backpropagation. In the context of neural networks, where the input dimensionality is often much higher than the dimensionality of

the labels, the reverse mode is computationally significantly cheaper than the forward mode. Let us start with an instructive example.

Example 5.13
Consider the function

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right) \tag{5.135}$$

from (5.122). If we were to implement a function f on a computer, we would be able to save some computation by using *intermediate variables*:

intermediate variables

$$a = x^2\,, \tag{5.136}$$
$$b = \exp(a)\,, \tag{5.137}$$
$$c = a + b\,, \tag{5.138}$$
$$d = \sqrt{c}\,, \tag{5.139}$$
$$e = \cos(c)\,, \tag{5.140}$$
$$f = d + e\,. \tag{5.141}$$

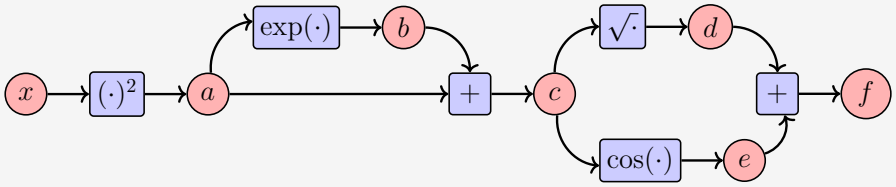


Figure 5.11 Computation graph with inputs x, function values f and intermediate variables a, b, c, d, e.

This is the same kind of thinking process that occurs when applying the chain rule. Observe that the above set of equations require fewer operations than a direct naive implementation of the function $f(x)$ as defined in (5.122). The corresponding computation graph in Figure 5.11 shows the flow of data and computations required to obtain the function value f.

The set of equations that include intermediate variables can be thought of as a computation graph, a representation that is widely used in implementations of neural network software libraries. We can directly compute the derivatives of the intermediate variables with respect to their corresponding inputs by recalling the definition of the derivative of elementary functions. We obtain:

$$\frac{\partial a}{\partial x} = 2x\,, \tag{5.142}$$
$$\frac{\partial b}{\partial a} = \exp(a)\,, \tag{5.143}$$
$$\frac{\partial c}{\partial a} = 1 = \frac{\partial c}{\partial b}\,, \tag{5.144}$$

$$\frac{\partial d}{\partial c} = \frac{1}{2\sqrt{c}}\,, \tag{5.145}$$

$$\frac{\partial e}{\partial c} = -\sin(c)\,, \tag{5.146}$$

$$\frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}\,. \tag{5.147}$$

By looking at the computation graph in Figure 5.11, we can compute $\partial f/\partial x$ by working backward from the output, and we obtain the following relations:

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c}\,, \tag{5.148}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial b}\,, \tag{5.149}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}\,, \tag{5.150}$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}\,. \tag{5.151}$$

Note that we have implicitly applied the chain rule to obtain $\partial f/\partial x$. By substituting the results of the derivatives of the elementary functions, we get

$$\frac{\partial f}{\partial c} = 1 \cdot \frac{1}{2\sqrt{c}} + 1 \cdot (-\sin(c))\,, \tag{5.152}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \cdot 1\,, \tag{5.153}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\exp(a) + \frac{\partial f}{\partial c} \cdot 1\,, \tag{5.154}$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \cdot 2x\,. \tag{5.155}$$

By thinking of each of the derivatives above as a variable, we observe that the computation required for calculating the derivative is of similar complexity as the computation of the function itself. This is quite counterintuitive since the mathematical expression for the derivative $\frac{\partial f}{\partial x}$ (5.123) is significantly more complicated than the mathematical expression of the function $f(x)$ in (5.122).

Automatic differentiation is a formalization of the example above. Let $x_1, \ldots, x_d$ be the input variables to the function, $x_{d+1}, \ldots, x_{D-1}$ be the intermediate variables and x_D the output variable. Then the computation graph can be expressed as an equation

$$\text{For } i = d+1, \ldots, D: \quad x_i = g_i(x_{\text{Pa}(x_i)}) \tag{5.156}$$

where $g_i(\cdot)$ are elementary functions and $x_{\mathrm{Pa}(x_i)}$ are the parent nodes of the variable x_i in the graph. Given a function defined in this way, we can use the chain rule to compute the derivative of the function in a step-by-step fashion. Recall that by definition $f = x_D$ and hence

$$\frac{\partial f}{\partial x_D} = 1. \tag{5.157}$$

For other variables x_i, we apply the chain rule

$$\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in \mathrm{Pa}(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in \mathrm{Pa}(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial g_j}{\partial x_i}\,, \tag{5.158}$$

where $\mathrm{Pa}(x_j)$ is the set of parent nodes of x_j in the computation graph. Equation (5.156) is the forward propagation of a function, whereas (5.158) is the backpropagation of the gradient through the computation graph. For neural network training we backpropagate the error of the prediction with respect to the label.

Auto-differentiation in reverse mode requires a parse tree.

The automatic differentiation approach above works whenever we have a function that can be expressed as a computation graph, where the elementary functions are differentiable. In fact, the function may not even be a mathematical function but a computer program. However, not all computer programs can be automatically differentiated, e.g., if we cannot find differential elementary functions. Programming structures, such as `for` loops and `if` statements require more care as well.

5.7 Higher-order Derivatives

So far, we discussed gradients, i.e., first-order derivatives. Sometimes, we are interested in derivatives of higher order, e.g., when we want to use Newton's Method for optimization, which requires second-order derivatives (Nocedal and Wright, 2006). In Section 5.1.1, we discussed the Taylor series to approximate functions using polynomials. In the multivariate case, we can do exactly the same. In the following, we will do exactly this. But let us start with some notation.

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x, y. We use the following notation for higher-order partial derivatives (and for gradients):

- $\frac{\partial^2 f}{\partial x^2}$ is the second partial derivative of f with respect to x
- $\frac{\partial^n f}{\partial x^n}$ is the nth partial derivative of f with respect to x
- $\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)$ is the partial derivative obtained by first partial differentiating with respect to x and then with respect to y
- $\frac{\partial^2 f}{\partial x \partial y}$ is the partial derivative obtained by first partial differentiating by y and then x

Hessian

The *Hessian* is the collection of all second-order partial derivatives.

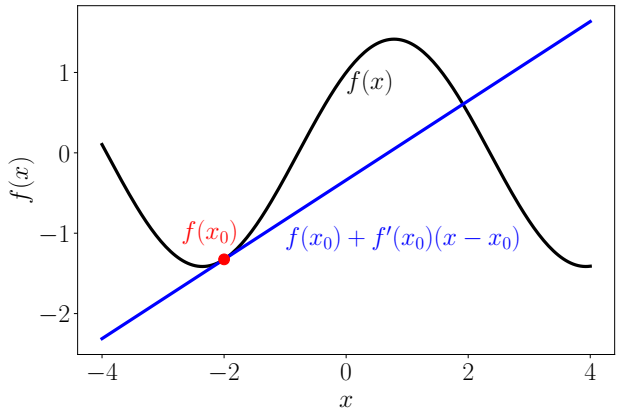


Figure 5.12 Linear approximation of a function. The original function f is linearized at $x_0 = -2$ using a first-order Taylor series expansion.

If $f(x, y)$ is a twice (continuously) differentiable function then

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}\,, \tag{5.159}$$

i.e., the order of differentiation does not matter, and the corresponding *Hessian matrix* Hessian matrix

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{5.160}$$

is symmetric. Generally, for $\boldsymbol{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian is an $n \times n$ matrix. The Hessian measures the local geometry of curvature.

Remark (Hessian of a Vector Field). If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a vector field, the Hessian is an $(m \times n \times n)$-tensor. ♢

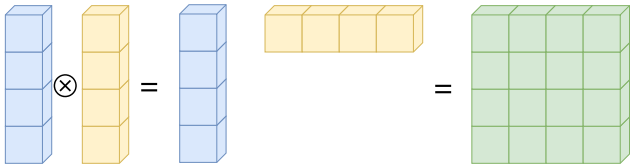
5.8 Linearization and Multivariate Taylor Series

The gradient ∇f of a function f is often used for a locally linear approximation of f around $\boldsymbol{x}_0$:

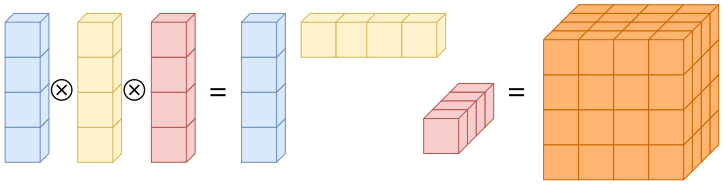
$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)\,. \tag{5.161}$$

Here $(\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)$ is the gradient of f with respect to $\boldsymbol{x}$, evaluated at $\boldsymbol{x}_0$. Figure 5.12 illustrates the linear approximation of a function f at an input x_0. The orginal function is approximated by a straight line. This approximation is locally accurate, but the further we move away from x_0 the worse the approximation gets. Equation (5.161) is a special case of a multivariate Taylor series expansion of f at $\boldsymbol{x}_0$, where we consider only the first two terms. We discuss the more general case in the following, which will allow for better approximations.

Definition 5.7 (Multivariate Taylor Series)**.** For the *multivariate Taylor* multivariate Taylor series



(a) Given a vector $\boldsymbol{\delta} \in \mathbb{R}^4$, we obtain the outer product $\boldsymbol{\delta}^2 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top \in \mathbb{R}^{4\times 4}$ as a matrix.



(b) An outer product $\boldsymbol{\delta}^3 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta} \in \mathbb{R}^{4\times 4\times 4}$ results in a third-order tensor ("three-dimensional matrix"), i.e., an array with three indexes.

Figure 5.13 Visualizing outer products. Outer products of vectors increase the dimensionality of the array by 1 per term.

series, we consider a function

$$f : \mathbb{R}^D \to \mathbb{R} \tag{5.162}$$

$$\boldsymbol{x} \mapsto f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^D, \tag{5.163}$$

that is smooth at $\boldsymbol{x}_0$.

When we define the difference vector $\boldsymbol{\delta} := \boldsymbol{x} - \boldsymbol{x}_0$, the Taylor series of f at $(\boldsymbol{x}_0)$ is defined as

$$f(\boldsymbol{x}) = \sum_{k=0}^{\infty} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} \boldsymbol{\delta}^k, \tag{5.164}$$

where $D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)$ is the k-th (total) derivative of f with respect to $\boldsymbol{x}$, evaluated at $\boldsymbol{x}_0$.

Taylor polynomial

Definition 5.8 (Taylor Polynomial)**.** The *Taylor polynomial* of degree n of f at $\boldsymbol{x}_0$ contains the first $n+1$ components of the series in (5.164) and is defined as

$$T_n = \sum_{k=0}^{n} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} \boldsymbol{\delta}^k. \tag{5.165}$$

Remark (Notation). In (5.164) and (5.165), we used the slightly sloppy notation of $\boldsymbol{\delta}^k$, which is not defined for vectors $\boldsymbol{x} \in \mathbb{R}^D$, $D > 1$, and $k > 1$. Note that both $D_x^k f$ and $\boldsymbol{\delta}^k$ are k-th order tensors, i.e., k-dimensional arrays. The k-th order tensor $\boldsymbol{\delta}^k \in \mathbb{R}^{\overbrace{D\times D\times \ldots \times D}^{k \text{ times}}}$ is obtained as a k-fold outer product, denoted by $\otimes$, of the vector $\boldsymbol{\delta} \in \mathbb{R}^D$. For example,

A vector can be implemented as a 1-dimensional array, a matrix as a 2-dimensional array.

$$\boldsymbol{\delta}^2 = \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top, \quad \boldsymbol{\delta}^2[i,j] = \delta[i]\delta[j] \tag{5.166}$$

$$\boldsymbol{\delta}^3 = \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta}, \quad \boldsymbol{\delta}^3[i,j,k] = \delta[i]\delta[j]\delta[k]. \tag{5.167}$$

Figure 5.13 visualizes two such outer products. In general, we obtain the following terms in the Taylor series:

$$D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k = \sum_a \cdots \sum_k D_x^k f(\boldsymbol{x}_0)[a,\ldots,k]\delta[a]\cdots\delta[k]\,, \tag{5.168}$$

where $D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k$ contains k-th order polynomials.

Now that we defined the Taylor series for vector fields, let us explicitly write down the first terms $D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k$ of the Taylor series expansion for $k = 0,\ldots,3$ and $\boldsymbol{\delta} := \boldsymbol{x} - \boldsymbol{x}_0$:

$$k = 0 : D_x^0 f(\boldsymbol{x}_0)\boldsymbol{\delta}^0 = f(\boldsymbol{x}_0) \in \mathbb{R} \tag{5.169}$$

```
np.einsum(
'i,i',Df1,d)
```

$$k = 1 : D_x^1 f(\boldsymbol{x}_0)\boldsymbol{\delta}^1 = \underbrace{\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)}_{1\times D}\underbrace{\boldsymbol{\delta}}_{D\times 1} = \sum_i \nabla_x f(\boldsymbol{x}_0)[i]\delta[i] \in \mathbb{R} \tag{5.170}$$

```
np.einsum(
'ij,i,j',
Df2,d,d)
```

$$k = 2 : D_x^2 f(\boldsymbol{x}_0)\boldsymbol{\delta}^2 = \mathrm{tr}(\underbrace{\boldsymbol{H}}_{D\times D}\underbrace{\boldsymbol{\delta}}_{D\times 1}\underbrace{\boldsymbol{\delta}^\top}_{1\times D}) = \boldsymbol{\delta}^\top \boldsymbol{H}\boldsymbol{\delta} \tag{5.171}$$

```
np.einsum(
'ijk,i,j,k',
Df3,d,d,d)
```

$$= \sum_i \sum_j H[i,j]\delta[i]\delta[j] \in \mathbb{R} \tag{5.172}$$

$$k = 3 : D_x^3 f(\boldsymbol{x}_0)\boldsymbol{\delta}^3 = \sum_i \sum_j \sum_k D_x^3 f(\boldsymbol{x}_0)[i,j,k]\delta[i]\delta[j]\delta[k] \in \mathbb{R} \tag{5.173}$$

♢

Example 5.14 (Taylor-Series Expansion of a Function with Two Variables)
Consider the function

$$f(x,y) = x^2 + 2xy + y^3\,. \tag{5.174}$$

We want to compute the Taylor series expansion of f at $(x_0, y_0) = (1, 2)$. Before we start, let us discuss what to expect: The function in (5.174) is a polynomial of degree 3. We are looking for a Taylor series expansion, which itself is a linear combination of polynomials. Therefore, we do not expect the Taylor series expansion to contain terms of fourth or higher order to express a third-order polynomial. This means, it should be sufficient to determine the first four terms of (5.164) for an exact alternative representation of (5.174).

To determine the Taylor series expansion, start of with the constant term and the first-order derivatives, which are given by

$$f(1,2) = 13 \tag{5.175}$$

$$\frac{\partial f}{\partial x} = 2x + 2y \implies \frac{\partial f}{\partial x}(1,2) = 6 \tag{5.176}$$

$$\frac{\partial f}{\partial y} = 2x + 3y^2 \implies \frac{\partial f}{\partial y}(1,2) = 14\,. \tag{5.177}$$

Therefore, we obtain

$$D^1_{x,y} f(1,2) = \nabla_{x,y} f(1,2) = \begin{bmatrix} \frac{\partial f}{\partial x}(1,2) & \frac{\partial f}{\partial y}(1,2) \end{bmatrix} = \begin{bmatrix} 6 & 14 \end{bmatrix} \in \mathbb{R}^{1\times 2} \tag{5.178}$$

such that

$$\frac{D^1_{x,y} f(1,2)}{1!}\boldsymbol{\delta} = \begin{bmatrix} 6 & 14 \end{bmatrix} \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} = 6(x-1) + 14(y-2)\,. \tag{5.179}$$

Note that $D^1_{x,y} f(1,2)\boldsymbol{\delta}$ contains only linear terms, i.e., first-order polynomials.

The second-order partial derivatives are given by

$$\frac{\partial^2 f}{\partial x^2} = 2 \implies \frac{\partial^2 f}{\partial x^2}(1,2) = 2 \tag{5.180}$$

$$\frac{\partial^2 f}{\partial y^2} = 6y \implies \frac{\partial^2 f}{\partial y^2}(1,2) = 12 \tag{5.181}$$

$$\frac{\partial^2 f}{\partial y \partial x} = 2 \implies \frac{\partial^2 f}{\partial y \partial x}(1,2) = 2 \tag{5.182}$$

$$\frac{\partial^2 f}{\partial x \partial y} = 2 \implies \frac{\partial^2 f}{\partial x \partial y}(1,2) = 2\,. \tag{5.183}$$

When we collect the second-order partial derivatives, we obtain the Hessian

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 6y \end{bmatrix}, \tag{5.184}$$

such that

$$\boldsymbol{H}(1,2) = \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \in \mathbb{R}^{2\times 2}\,. \tag{5.185}$$

Therefore, the next term of the Taylor-series expansion is given by

$$\frac{D^2_{x,y} f(1,2)}{2!}\boldsymbol{\delta}^2 = \frac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{H}(1,2)\boldsymbol{\delta} \tag{5.186}$$

$$= \begin{bmatrix} x-1 & y-2 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} \tag{5.187}$$

$$= (x-1)^2 + 2(x-1)(y-2) + 6(y-2)^2\,. \tag{5.188}$$

Here, $D^2_{x,y} f(1,2)\boldsymbol{\delta}^2$ contains only quadratic terms, i.e., second-order polynomials.

The third-order derivatives are obtained as

$$D^3_{x,y} f = \begin{bmatrix} \frac{\partial \boldsymbol{H}}{\partial x} & \frac{\partial \boldsymbol{H}}{\partial y} \end{bmatrix} \in \mathbb{R}^{2\times 2\times 2}\,, \tag{5.189}$$

$$D^3_{x,y} f[:,:,1] = \frac{\partial \boldsymbol{H}}{\partial x} = \begin{bmatrix} \frac{\partial^3 f}{\partial x^3} & \frac{\partial^3 f}{\partial x^2 \partial y} \\ \frac{\partial^3 f}{\partial x \partial y \partial x} & \frac{\partial^3 f}{\partial x \partial y^2} \end{bmatrix}, \tag{5.190}$$

$$D^3_{x,y} f[:,:,2] = \frac{\partial \boldsymbol{H}}{\partial y} = \begin{bmatrix} \frac{\partial^3 f}{\partial y \partial x^2} & \frac{\partial^3 f}{\partial y \partial x \partial y} \\ \frac{\partial^3 f}{\partial y^2 \partial x} & \frac{\partial^3 f}{\partial y^3} \end{bmatrix}. \tag{5.191}$$

Since most second-order partial derivatives in the Hessian in (5.184) are constant the only non-zero third-order partial derivative is

$$\frac{\partial^3 f}{\partial y^3} = 6 \implies \frac{\partial^3 f}{\partial y^3}(1,2) = 6\,. \tag{5.192}$$

Higher-order derivatives and the mixed derivatives of degree 3 (e.g., $\frac{\partial f^3}{\partial x^2 \partial y}$) vanish, such that

$$D^3_{x,y} f[:,:,1] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad D^3_{x,y} f[:,:,2] = \begin{bmatrix} 0 & 0 \\ 0 & 6 \end{bmatrix} \tag{5.193}$$

and

$$\frac{D^3_{x,y} f(1,2)}{3!} \boldsymbol{\delta}^3 = (y-2)^3\,, \tag{5.194}$$

which collects all cubic terms (third-order polynomials) of the Taylor series.

Overall, the (exact) Taylor series expansion of f at $(x_0, y_0) = (1,2)$ is

$$f(x) = f(1,2) + D^1_{x,y} f(1,2)\boldsymbol{\delta} + \frac{D^2_{x,y} f(1,2)}{2!}\boldsymbol{\delta}^2 + \frac{D^3_{x,y} f(1,2)}{3!}\boldsymbol{\delta}^3 \tag{5.195}$$

$$= f(1,2) + \frac{\partial f(1,2)}{\partial x}(x-1) + \frac{\partial f(1,2)}{\partial y}(y-2) \tag{5.196}$$

$$+ \frac{1}{2!}\left(\frac{\partial^2 f(1,2)}{\partial x^2}(x-1)^2 + \frac{\partial^2 f(1,2)}{\partial y^2}(y-2)^2\right. \tag{5.197}$$

$$\left. + 2\frac{\partial^2 f(1,2)}{\partial x \partial y}(x-1)(y-2)\right) + \frac{1}{6}\frac{\partial^3 f(1,2)}{\partial y^3}(y-2)^3 \tag{5.198}$$

$$= 13 + 6(x-1) + 14(y-2) \tag{5.199}$$

$$+ (x-1)^2 + 6(y-2)^2 + 2(x-1)(y-2) + (y-2)^3\,. \tag{5.200}$$

In this case, we obtained an exact Taylor series expansion of the polynomial in (5.174), i.e., the polynomial in (5.200) is identical to the original polynomial in (5.174). In this particular example, this result is not surprising since the original function was a third-order polynomial, which we expressed through a linear combination of constant terms, first-order, second order and third-order polynomials in (5.200).

5.9 Further Reading

Further details of matrix differentials, along with a short review of the required linear algebra can be found in Magnus and Neudecker (2007). Automatic differentiation has had a long history, and the reader is referred to Griewank and Walther (2003, 2008); Elliott (2009) and their references.

In machine learning (and other disciplines), we often need to compute expectations, i.e., we need to solve integrals of the form

$$\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}\,. \tag{5.201}$$

Even if $p(\boldsymbol{x})$ is in a convenient form (e.g., Gaussian), this integral generally cannot be solved analytically. The Taylor series expansion of f is one way of finding an approximate solution: Assuming $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is Gaussian, then the first-order Taylor series expansion around $\boldsymbol{\mu}$ locally linearizes the nonlinear function f. For linear functions, we can compute the mean (and the covariance) exactly if $p(\boldsymbol{x})$ is Gaussian distributed (see Section 6.6). This property is heavily exploited by the *Extended Kalman Filter* (Maybeck, 1979) for online state estimation in nonlinear dynamical systems (also called "state-space models"). Other deterministic ways to approximate the integral in (5.201) are the *unscented transform* (Julier and Uhlmann, 1997), which does not require any gradients, or the *Laplace approximation* (Bishop, 2006), which uses the Hessian for a local Gaussian approximation of $p(\boldsymbol{x})$ at the posterior mean.

Extended Kalman Filter

unscented transform

Laplace approximation

Exercises

5.1 Compute the derivative $f'(x)$ for

$$f(x) = \log(x^4)\sin(x^3)\,. \tag{5.202}$$

5.2 Compute the derivative $f'(x)$ of the logistic sigmoid

$$f(x) = \frac{1}{1+\exp(-x)}\,. \tag{5.203}$$

5.3 Compute the derivative $f'(x)$ of the function

$$f(x) = \exp(-\tfrac{1}{2\sigma^2}(x-\mu)^2)\,, \tag{5.204}$$

where $\mu,\ \sigma \in \mathbb{R}$ are constants.

5.4 Compute the Taylor polynomials T_n, $n = 0, \ldots, 5$ of $f(x) = \sin(x) + \cos(x)$ at $x_0 = 0$.

5.5 Consider the following functions

$$f_1(\boldsymbol{x}) = \sin(x_1)\cos(x_2)\,, \quad \boldsymbol{x} \in \mathbb{R}^2 \tag{5.205}$$

$$f_2(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}\,, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \tag{5.206}$$

$$f_3(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{x}^\top\,, \quad \boldsymbol{x} \in \mathbb{R}^n \tag{5.207}$$

1. What are the dimensions of $\frac{\partial f_i}{\partial \boldsymbol{x}}$?

2982 2. Compute the Jacobians

5.6 Differentiate f with respect to $\boldsymbol{t}$ and g with respect to $\boldsymbol{X}$, where

$$f(\boldsymbol{t}) = \sin(\log(\boldsymbol{t}^\top \boldsymbol{t}))\,, \qquad t \in \mathbb{R}^D \tag{5.208}$$

$$g(\boldsymbol{X}) = \mathrm{tr}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B})\,, \qquad \boldsymbol{A} \in \mathbb{R}^{D\times E}, \boldsymbol{X} \in \mathbb{R}^{E\times F}, \boldsymbol{B} \in \mathbb{R}^{F\times D}\,, \tag{5.209}$$

2983 where tr denotes the trace.

2984 5.7 Compute the derivatives $df/d\boldsymbol{x}$ of the following functions by using the chain
2985 rule. Provide the dimensions of every single partial derivative. Describe your
2986 steps in detail.

1.

$$f(z) = \log(1+z)\,, \quad z = \boldsymbol{x}^\top \boldsymbol{x}\,, \quad \boldsymbol{x} \in \mathbb{R}^D$$

2.

$$f(\boldsymbol{z}) = \sin(\boldsymbol{z})\,, \quad \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\,, \quad \boldsymbol{A} \in \mathbb{R}^{E\times D}, \boldsymbol{x} \in \mathbb{R}^D, \boldsymbol{b} \in \mathbb{R}^E$$

2987 where $\sin(\cdot)$ is applied to every element of $\boldsymbol{z}$.

2988 5.8 Compute the derivatives $df/d\boldsymbol{x}$ of the following functions.
2989 Describe your steps in detail.

1. Use the chain rule. Provide the dimensions of every single partial derivative.

$$\begin{aligned} f(z) &= \exp(-\tfrac{1}{2}z) \\ z &= g(\boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{S}^{-1}\boldsymbol{y} \\ \boldsymbol{y} &= h(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{\mu} \end{aligned}$$

2990 where $\boldsymbol{x}, \boldsymbol{\mu} \in \mathbb{R}^D$, $\boldsymbol{S} \in \mathbb{R}^{D\times D}$.

2.

$$f(\boldsymbol{x}) = \mathrm{tr}(\boldsymbol{x}\boldsymbol{x}^\top + \sigma^2\boldsymbol{I})\,, \quad \boldsymbol{x} \in \mathbb{R}^D$$

2991 Here $\mathrm{tr}(\boldsymbol{A})$ is the trace of $\boldsymbol{A}$, i.e., the sum of the diagonal elements A_{ii}.
2992 *Hint: Explicitly write out the outer product.*

3. Use the chain rule. Provide the dimensions of every single partial derivative. You do not need to compute the product of the partial derivatives explicitly.

$$\begin{aligned} \boldsymbol{f} &= \tanh(\boldsymbol{z}) \in \mathbb{R}^M \\ \boldsymbol{z} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M. \end{aligned}$$

2993 Here, $\tanh$ is applied to every component of $\boldsymbol{z}$.

6

Probability and Distributions

Probability, loosely speaking, is the study of uncertainty. Probability can be thought of as the fraction of times an event occurs, or as a degree of belief about an event. We then would like to use this probability to measure the chance of something occurring in an experiment. As mentioned in the introduction (Chapter 1), we would often like to quantify uncertainty: uncertainty in the data, uncertainty in the machine learning model, and uncertainty in the predictions produced by the model. Quantifying uncertainty requires the idea of a *random variable*, which is a function that maps outcomes of random experiments to real numbers. Associated with the random variable is a number corresponding to each possible mapping of outcomes to real numbers. This set of numbers specifies the probability of occurrence, and is called the *probability distribution*.

random variable

probability distribution

Probability distributions are used as a building block for other concepts, such as model selection (Section 8.4) and graphical models (Section 8.5). In this section, we present the three concepts that define a probability space: the state space, the events and the probability of an event. The presentation is deliberately slightly hand wavy since a rigorous presentation would occlude the main idea.

6.1 Construction of a Probability Space

The theory of probability aims at defining a mathematical structure to describe random outcomes of experiments. For example, when tossing a single coin, one cannot determine the outcome, but by doing a large number of coin tosses, one can observe a regularity in the average outcome. Using this mathematical structure of probability, the goal is to perform automated reasoning, and in this sense probability generalizes logical reasoning (Jaynes, 2003).

6.1.1 Philosophical Issues

When constructing automated reasoning systems, classical Boolean logic does not allow us to express certain forms of plausible reasoning. Consider the following scenario: We observe that A is false. We find B becomes less plausible although no conclusion can be drawn from classical logic. We

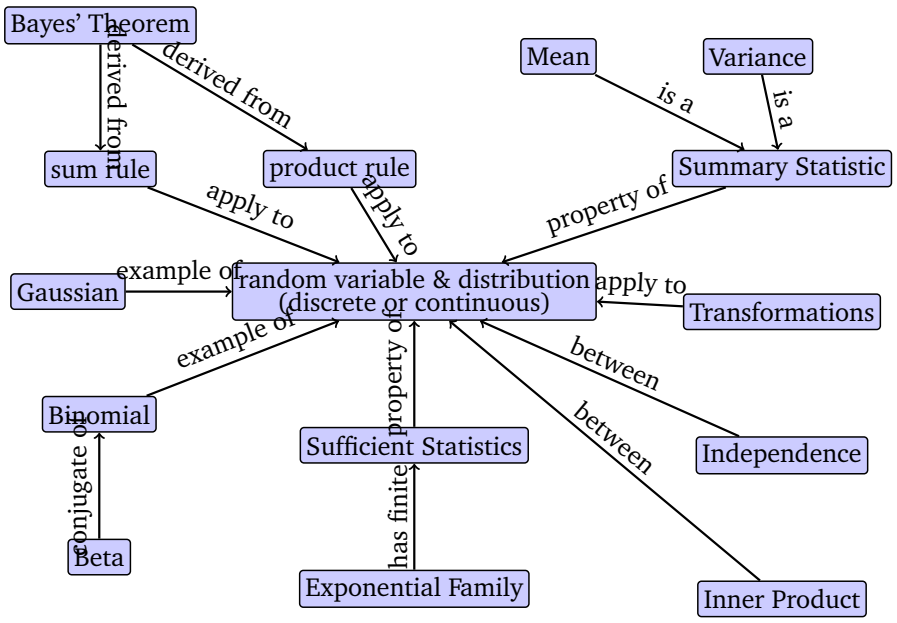


Figure 6.1 A mind map of the concepts related to random variables and probability distributions, as described in this chapter.

observe that B is true. It seems A becomes more plausible. We use this form of reasoning daily: Our friend is late. We have three hypotheses H1, H2, H3. Was she H1 abducted by aliens, H2 abducted by kidnappers or H3 delayed by traffic. How do we conclude H3 is the most plausible answer? Seen in this way, probability theory can be considered a generalization of Boolean logic. In the context of machine learning, it is often applied in this way to formalize the design of automated reasoning systems. Further arguments about how probability theory is the foundation of reasoning systems can be found in (Pearl, 1988).

"For plausible reasoning it is necessary to extend the discrete true and false values of truth to continuous plausibilities."(Jaynes, 2003)

The philosophical basis of probability and how it should be somehow related to what we think should be true (in the logical sense) was studied by Cox (Jaynes, 2003). Another way to think about it is that if we are precise about our common sense constructing probabilities. E.T. Jaynes (1922–1998) identified three mathematical criteria, which must apply to all plausibilities:

1. The degrees of plausibility are represented by real numbers.
2. These numbers must be based on the rules of common sense.
    1. Consistency or non-contradiction: when the same result can be reached through different means, the same plausibility value must be found in all cases.
    2. Honesty: All available data must be taken into account.
    3. Reproducibility: If our state of knowledge about two problems are the same, then we must assign the same degree of plausibility to both of them.

The Cox-Jaynes's theorem proves these plausibilities to be sufficient to define the universal mathematical rules that apply to plausibility p, up to an arbitrary monotonic function. Crucially, these rules *are* the rules of probability.

Remark. In machine learning and statistics, there are two major interpretations of probability: the Bayesian and frequentist interpretations (Bishop, 2006). The Bayesian interpretation uses probability to specify the degree of uncertainty that the user has about an event, and is sometimes referred to as subjective probability or degree of belief. The frequentist interpretation The frequentist interpretation considers probability to be the relative frequencies of events, in the limit when one has infinite data. ♢

It is worth noting that some machine learning literature on probabilistic models use lazy notation and jargon, which is confusing. Multiple distinct concepts are all referred to as "probability distribution", and the reader has to often disentangle the meaning from the context. One trick to help make sense of probability distributions is to check whether we are trying to model something categorical (a discrete random variable) or something continuous (a continous random variable). The kinds of questions we tackle in machine learning are closely related to whether we are considering categorical or continuous models.

6.1.2 Probability and Random Variables

Modern probability is based on a set of axioms proposed by Kolmogorov (Jacod and Protter, 2004, Chapter 1 and 2) that introduce the three concepts of state space, event space and probability measure.

The state space Ω

state space

The *state space* is the set of all possible outcomes of the experiment, usually denoted by Ω. For example, two successive coin tosses have a state space of $\{$hh, tt, ht, th$\}$, where "h" denotes "heads" and "t" denotes "tails".

The events $\mathcal{A}$

events

The *events* can be observed after the experiment is done, i.e., they are realizations of an experiment. The event space is often denoted by $\mathcal{A}$ and is also often the set of all subsets of Ω. In the two coins example, one possible element of $\mathcal{A}$ is the event when both tosses are the same, that is $\{$hh, tt$\}$.

The probability $P(A)$

With each event $A \in \mathcal{A}$, we associate a number $P(A)$ that measures the probability or belief that the event will occur. $P(A)$ is called the *probability* of A.

probability

The probability of a single event must lie in the interval $[0, 1]$, and the total probability over all states in the state space must sum to 1, i.e.,

$\sum_{A\in\mathcal{A}} P(A) = 1$. We associate this number (the probability) to a particular event occurring, and intuitively understand this as the chance that this event occurs. This association or mapping is called a *random variable*. This brings us back to the concepts at the beginning of this chapter, where we can see that a random variable is a map from Ω to $\mathbb{R}$. The name "random variable" is a great source of misunderstanding as it is neither random nor is it a variable. It is a function.

random variable

We omit the definition of a random variable as this will become too technical for the purpose of this book.

Remark. The state space Ω above unfortunately is referred to by different names in different books. Another common name for Ω is sample space (Grinstead and Snell, 1997; Jaynes, 2003), and state space is sometimes reserved for referring to states in a dynamical system (Hasselblatt and Katok, 2003). Other names sometimes used to describe Ω are: sample description space, possibility space and (very confusingly) event space. ♢

We say that a random variable is distributed according to a particular probability distribution, which defines the probability mapping between the event and the probability of the event. The two concepts are intertwined, but for ease of presentation we will discuss some properties with respect to random variables and others with respect to their distributions. An outline of the concepts presented in this chapter are shown in Figure 6.1.

6.1.3 Statistics

Probability theory and statistics are often presented together, and in some sense they are intertwined. One way of contrasting them is by the kinds of problems that are considered. Using probability we can consider a model of some process where the underlying uncertainty is captured by random variables, and we use the rules of probability to derive what happens. Using statistics we observe that something has happened, and try to figure out the underlying process that explains the observations. In this sense machine learning is close to statistics in its goals, that is to construct a model that adequately represents the process that generated the data. When the machine learning model is a probabilistic model, we can use the rules of probability to calculate the "best fitting" model for some data.

Another aspect of machine learning systems is that we are interested in generalization error. This means that we are actually interested in the performance of our system on instances that we will observe in future, which are not identical to the instances that we have seen so far. This analysis of future performance relies on probability and statistics, most of which is beyond what will be presented in this chapter. The interested reader is encouraged to look at the books by Shalev-Shwartz and Ben-David (2014); Boucheron et al. (2013). We will see more about statistics in Chapter 8.

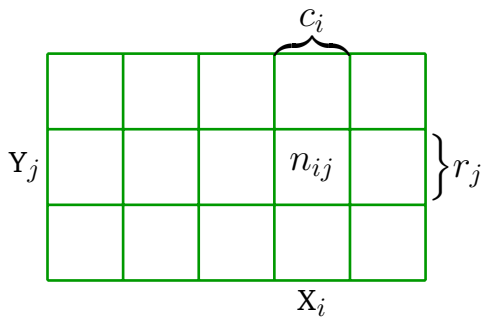


Figure 6.2 Visualization of a discrete bivariate probability mass function, with random variables x and y. This diagram is from Bishop (2006).

6.2 Discrete and Continuous Probabilities

Let us focus our attention on ways to describe the probability of an event, as introduced in Section 6.1. Depending on whether the state space is discrete or continuous, the natural way to refer to distributions is different. When the state space Ω is discrete, we can specify the probability that a random variable x takes a particular value $\mathrm{x} \in \Omega$, denoted as $P(x = \mathrm{x})$. The expression $P(x = \mathrm{x})$ for a discrete random variable x is known as the *probability mass function*. We will discuss discrete random variables in the following subsection. When the state space Ω is continuous, for example the real line $\mathbb{R}$, it is more natural to specify the probability that a random variable x is in an interval. By convention we specify the probability that a random variable x is less than a particular value x, denoted $P(x \leqslant \mathrm{x})$. The expression $P(x \leqslant \mathrm{x})$ for a continuous random variable x is known as the *cumulative distribution function*. We will discuss continuous random variables in Section 6.2.2. We will revisit the nomenclature and contrast discrete and continuous random variables in Section 6.2.3.

Many probability textbooks tend to use capital letters X for random variables, and small letters x for their values.

probability mass function

cumulative distribution function

6.2.1 Discrete Probabilities

When the state space is discrete, we can imagine the probability distribution of multiple random variables as filling out a (multidimensional) array of numbers. We define the *joint probability* as the entry of both values jointly.

joint probability

$$P(x = \mathrm{x}_i, y = \mathrm{y}_i) = \frac{n_{ij}}{N} \,. \tag{6.1}$$

To be precise, the above table defines the *probability mass function* (pmf) of a discrete probability distribution. For two random variables x and y, the probability that $x = \mathrm{x}$ and $y = \mathrm{y}$ is (lazily) written as $p(x, y)$ and is called the joint probability. The *marginal probability* is obtained by summing over a row or column. The *conditional probability* is the fraction of a row or column in a particular cell.

probability mass function

marginal probability

conditional probability

Example 6.1
Consider two random variables x and y, where x has five possible states and y has three possible states, as shown in Figure 6.2. The value c_i is the sum of the individual probabilities for the i$^{\text{th}}$ column, that is $c_i = \sum_{j=1}^{3} n_{ij}$. Simularly, the value r_j is the row sum, that is $r_j = \sum_{i=1}^{5} n_{ij}$. Using these definitions, we can compactly express the distribution of x and y by themselves.

The probability distribution of each random variable, the marginal probability, which can be seen as the sum over a row or column

$$P(x = \mathrm{X}_i) = \frac{c_i}{N} = \frac{\sum_{j=1}^{3} n_{ij}}{N} \tag{6.2}$$

and

$$P(y = \mathrm{Y}_j) = \frac{r_j}{N} = \frac{\sum_{i=1}^{5} n_{ij}}{N}, \tag{6.3}$$

where c_i and r_j are the ith column and jth row of the probability table, respectively. Recall that by the axioms of probability (Section 6.1) we require that the probabilities sum up to one, that is

$$\sum_{i=1}^{3} P(x = \mathrm{X}_i) = 1 \quad \text{and} \quad \sum_{j=1}^{5} P(y = \mathrm{Y}_j) = 1\,. \tag{6.4}$$

The conditional probability is the fraction of a row or column in a particular cell. For example the conditional probability of y given x is

$$p(y = \mathrm{Y}_j \,|\, x = \mathrm{X}_i) = \frac{n_{ij}}{c_i}\,, \tag{6.5}$$

and the conditional probability of x given y is

$$p(x = \mathrm{X}_i \,|\, y = \mathrm{Y}_j) = \frac{n_{ij}}{r_j}\,, \tag{6.6}$$

The marginal probability that x takes the value X irrespective of the value of random variable y is (lazily) written as $p(x)$. If we consider only the instances where $x = \mathrm{X}$, then the fraction of instances (the conditional probability) for which $y = \mathrm{Y}$ is written (lazily) as $p(y \,|\, x)$.

Example 6.2
Consider a statistical experiment where we perform a medical test for cancer two times. There are two possible outcomes for each test, and hence there are four outcomes in total. The state space or sample space Ω of this experiment is then (cancer, cancer), (cancer, healthy), (healthy, cancer), (healthy, healthy). The event we are interested in is the total

This toy example is essentially a coin flip example.

number of times the repeated medical test returns a cancerous answer, where we can see from the above state space can occur in no test, either one of the tests or both tests. Therefore the event space $\mathcal{A}$ is 0, 1, 2. Let variable x denote the number of times the medical test returns "cancer". Then x is a random variable (a function) that counts the number of times "cancer" appears. It can be represented as a table as below

$$x((\text{cancer}, \text{cancer})) = 2 \tag{6.7}$$
$$x((\text{cancer}, \text{healthy})) = 1 \tag{6.8}$$
$$x((\text{healthy}, \text{cancer})) = 1 \tag{6.9}$$
$$x((\text{healthy}, \text{healthy})) = 0. \tag{6.10}$$

Let us assume that this useless test returns at random a value of "cancer" with probability 0.3, ignoring any real world information. This assumption also implies that the two tests are independent of each other, which we will discuss in Section 6.4.3. Note that since there are two states which map to the same event, where only one of the tests say "cancer". Therefore the probability mass function of x is given by the table below

$$P(x = 2) = 0.09 \tag{6.11}$$
$$P(x = 1) = 0.42 \tag{6.12}$$
$$P(x = 0) = 0.49. \tag{6.13}$$

In machine learning, we use discrete probability distributions to model *categorical variables*, i.e., variables that take a finite set of unordered values. These could be categorical features such as the gender of a person when used for predicting the salary of a person, or categorical labels such as letters of the alphabet when doing handwritten recognition. Discrete distributions are often used to construct probabilistic models that combine a finite number of continuous distributions. We will see the Gaussian mixture model in Chapter 11.

categorical variables

6.2.2 Continuous Probabilities

When we consider real valued random variables, that is when we consider state spaces which are intervals of the real line $\mathbb{R}$ we have corresponding definitions to the discrete case (Section 6.2.1). We will sweep measure theoretic considerations under the carpet in this book, and pretend as if we can perform operations as if we have discrete probability spaces with finite states. However this simplification is not precise for two situations: when we repeat something infinitely often, and when we want to draw a point from an interval. The first situation arises when we discuss generalization error in machine learning (Chapter 8). The second situation arises

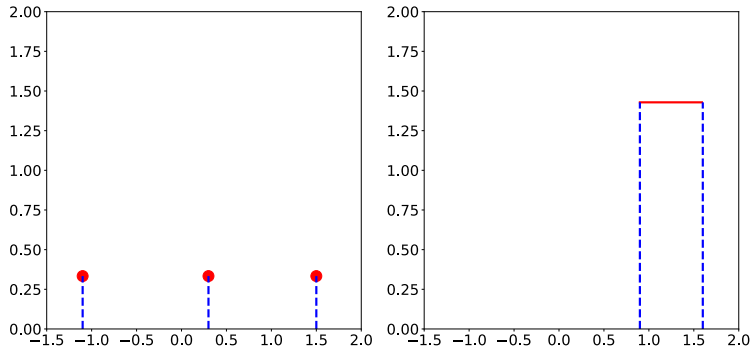


Figure 6.3 Examples of Uniform distributions. (left) discrete, (right) continuous. See example for details of the distributions.

when we want to discuss continuous distributions such as the Gaussian (Section 6.6). For our purposes, the lack of precision allows a more brief introduction to probability. A reader interested a measure based approach is referred to Billingsley (1995).

Definition 6.1 (Probability Density Function)**.** A function $f : \mathbb{R}^D \to \mathbb{R}$ is called a *probability density function* (pdf) if

probability density function

1. $\forall \boldsymbol{x} \in \mathbb{R}^D : f(\boldsymbol{x}) \geqslant 0$
2. Its integral exists and

$$\int_{\mathbb{R}^D} f(\boldsymbol{x}) d\boldsymbol{x} = 1\,. \tag{6.14}$$

Here, $\boldsymbol{x} \in \mathbb{R}^D$ is a (continuous) random variable. For discrete random variables, the integral in (6.14) is replaced with a sum.

Definition 6.2 (Cumulative Distribution Function)**.** A *cumulative distribution function* (cdf) of a multivariate real-valued random variable $\boldsymbol{x} \in \mathbb{R}^D$ is given by

cumulative distribution function

$$F_{\boldsymbol{x}}(\mathrm{x}) = P(x_1 \leqslant \mathrm{x}_1, \ldots, x_D \leqslant \mathrm{x}_D) \tag{6.15}$$

where the right hand side represents the probability that random variable x_i takes the value smaller than x_i. This can be expressed also as the integral of the probability density function,

$$F_{\boldsymbol{x}}(\mathrm{x}) = \int_{-\infty}^{\boldsymbol{x}} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{6.16}$$

6.2.3 Contrasting Discrete and Continuous Distributions

Let us consider both discrete and continuous distributions, and contrast them. The aim here is to see that while both discrete and continuous distributions seem to have similar requirements, such as the total probability

mass is 1, they are subtly different. Since the total probability mass of a discrete random variable is 1 (Equation (6.4)), and there are a finite number of states, the probability of each state must lie in the interval $[0, 1]$. However the analogous requirement for continuous random variables (Equation (6.14)) does not imply that the value of the density is less than 1 for all values. We illustrate this using the *uniform distribution* for both discrete and continuous random variables.

uniform distribution

Example 6.3
We consider two examples of the uniform distribution, where each state is equally likely to occur. This example illustrates the difference between discrete and continuous probability distributions.

Let z be a discrete uniform random variable with three states $\{z = -1.1, z = 0.3, z = 1.5\}$. Note that the actual values of these states are not meaningful here, and we deliberately used numbers to drive home the point that we do not want to use (and should ignore) the ordering of the states. The probability mass function can be represented as a table of probability values.

z	-1.1	0.3	1.5
$P(z = \mathrm{z})$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Alternatively one could think of this as a graph (left of Figure 6.3), where we use the fact that the states can be located on the x-axis, and the y-axis represents the probability of a particular state. The y-axis in the left of Figure 6.3 is deliberately extended such that is it the same as the right figure.

Let x be a continuous random variable taking values in the range $0.9 \leqslant x \leqslant 1.6$, as represented by the graph on the right in Figure 6.3. Observe that the height of the density can be more than 1. However, it needs to hold that

$$\int_{0.9}^{1.6} p(x)\mathrm{d}x = 1\,. \tag{6.17}$$

Very often the literature uses lazy notation and nomenclature that can be confusing to a beginner. For a value x of a state space Ω, $p(x)$ denotes the probability that random variable x takes value x, i.e., $P(x = \mathrm{x})$, which is known as the probability mass function. This is often referred to as the "distribution". For continuous variables, $p(x)$ is called the probability density function (often referred to as a density), and to make things even more confusing the cumulative distribution function $P(x \leqslant \mathrm{x})$ is often also referred to as the "distribution". In this chapter we often will use the

	"point probability"	"interval probability"
discrete	$P(x = \mathrm{x})$ probability mass function	not applicable
continuous	$p(x)$ probability density function	$P(x \leqslant \mathrm{x})$ cumulative distribution function

Table 6.1 Nomenclature for probability distributions.

notation x or $\boldsymbol{x}$ to refer to univariate and multivariate random variables respectively. We summarise the nomenclature in Table 6.1.

Remark. We will be using the expression "probability distribution" not only for discrete distributions but also for continuous probability density functions, although this is technically incorrect. However, this is consistent with the majority of machine learning literature. ♢

6.3 Sum Rule, Product Rule and Bayes' Theorem

When we think of a probabilisitic model as an extension to logical reasoning, as we discussed in Section 6.1.1, the rules of probability presented here follow naturally from fulfilling the desiderata (Jaynes, 2003, Chapter 2). Probabilistic modelling provides a principled foundation for designing machine learning methods. Once we have defined probability distributions (Section 6.2) corresponding to the uncertainties of the data and our problem, it turns out that there are only two fundamental rules, the sum rule and the product rule, that govern probabilistic inference.

Before we define the sum rule and product rule, let us briefly explore how to use probabilistic models to capture uncertainty (Ghahramani, 2015). At the lowest modelling level, measurement noise introduces model uncertainty. for example the measurement error in a camera sensor. We will see in Chapter 9 how to use Gaussian (Section 6.6) noise models for linear regression. At higher modelling levels, we would be interested to model the uncertainty of the coefficients in linear regression. This uncertainty captures which values of these parameters will be good at predicting new data. Finally at the highest levels, we may want to capture uncertainties about the model structure. We discuss model selection issues in Chapter 8. Once we have the probabilistic models, the basic rules of probability presented in this section are used to infer the unobserved quantities given the observed data. The same rules of probability are used for inference (transforming prior probabilities to posterior probabilities) and learning (estimating the likelihood of the model for a given dataset).

Given the definitions of marginal and conditional probability for discrete and continuous random variables in the previous section, we can now present the two fundamental rules in probability theory. These two rules arise naturally (Jaynes, 2003) from the requirements we discussed in Section 6.1.1. Recall that $p(x, y)$ is the joint distribution of the two

random variables x, y, $p(x), p(y)$ are the corresponding marginal distributions, and $p(y \mid x)$ is the conditional distribution of y given x.

sum rule

The first rule, the *sum rule* is expressed for discrete random variables as

$$p(x) = \sum_y p(x, y) \qquad \text{sum rule/marginalization property} . \quad (6.18)$$

marginalization property

The sum above is over the set of states of the random variable Y. The sum rule is also known as the *marginalization property*. For continuous probability distributions, the sum is replaced by an integral

$$p(x) = \int_y p(x, y) dy . \quad (6.19)$$

The sum rule relates the joint distribution to a marginal distribution. In general, when the joint distribution contains more than two random variables, the sum rule can be applied to any subset of the random variables, resulting in a marginal distribution of potentially more than one random variable.

Remark. Many of the computational challenges of probabilistic modelling are due to the application of the sum rule. When there are many variables or discrete variables with many states, the sum rule boils down to performing a high dimensional sum or integral. Performing high dimensional sums or integrals are generally computationally hard, in the sense that there is no known polynomial time algorithm to calculate them exactly. ♢

product rule

The second rule, known as the *product rule*, relates the joint distribution to the conditional distribution

$$p(x, y) = p(y \mid x)p(x) \qquad \text{product rule.} \quad (6.20)$$

The product rule can be interpreted as the fact that every joint distribution of two random variables can be factorized (written as a product) of two other distributions. The two factors are the marginal distribution of the first random variable $p(x)$, and the conditional distribution of the second random variable given the first $p(y \mid x)$. Observe that since the ordering of random variables is arbitrary in $p(x, y)$ the product rule also implies $p(x, y) = p(x \mid y)p(y)$. To be precise, Equation (6.20) is expressed in terms of the probability mass functions for discrete random variables. For continuous random variables, the product rule is expressed in terms of the probability density functions (recall the discussion in Section 6.2.3).

In machine learning and Bayesian statistics, we are often interested in making inferences of random variables given that we have observed other random variables. Let us assume, we have some prior knowledge $p(x)$ about a random variable x and some relationship $p(y \mid x)$ between x and a second random variable y. If we now observe y, we can use Bayes' theorem to draw some conclusions about x given the observed values of y. *Bayes'*

Bayes' theorem is also called the "probabilistic inverse"

Bayes' theorem

theorem or *Bayes' law*

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(y)} \tag{6.21}$$

is a direct consequence of the sum and product rules in (6.18)–(6.20).

Example 6.4 (Applying the Sum and Product Rule)
We prove Bayes' theorem by using the sum and product rule. First we observe that we can apply the product rule in two ways,

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{y})p(\boldsymbol{y})\,. \tag{6.22}$$

Simple algebra then gives us (6.21). Very often in machine learning, the evidence term $p(\boldsymbol{x})$ is hard to estimate, and we rewrite it by using the sum and product rule.

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{y}} p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y})p(\boldsymbol{y})\,. \tag{6.23}$$

We now have an alternative formulation

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{y})p(\boldsymbol{y})}{\sum_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y})p(\boldsymbol{y})}\,. \tag{6.24}$$

In Equation (6.21), $p(y)$ is the *prior*, which encapsulates our prior knowledge of y, $p(x \mid y)$ is the *likelihood* that describes how x and y are related. The quantity $p(x)$ is the marginal likelihood or *evidence* and is a normalizing constant (independent of y). The *posterior* $p(\boldsymbol{x} \mid \boldsymbol{y})$ expresses exactly what we are interested in, i.e., what we know about x if we observe y. We will see an application of this in Maximum-A-Posteriori estimation (Section 9.2.3).

prior
likelihood
The likelihood is sometimes also called the "measurement model".
evidence
posterior

6.4 Summary Statistics and Independence

We are often interested in summarizing and contrasting random variables. A statistic of a random variable is a deterministic function of that random variable. The summary statistics of a distribution provide one useful view how a random variable behaves, and as the name suggests, provides numbers that summarize the distribution. The following describes the mean and the variance, two well known summary statistics. Then we discuss two ways to compare a pair of random variables: first how to say that two random variables are independent, and second how to compute an inner product between them.

6.4.1 Means and Covariances

Mean and (co)variance are often useful to describe properties of probability distributions (expected values and spread). We will see in Section 6.7 that there is a useful family of distributions (called the exponential family) where the statistics of the random variable capture all the possible information. The definitions in this section are stated for a general multivariate continuous random variable, because it is more intuitive to think about means and covariances in terms of real numbers. Analogous definitions exist for discrete random variables where the integral is replaced by a sum.

In one dimension, the mean value is the average value. It is the value obtained by summing up all values and dividing by the number of items. In more than one dimension, the sum becomes vector addition and the idea still holds. To account for the fact that we are dealing with a continuous random variable $\boldsymbol{x} \in \mathbb{R}^D$ with a particular density $p(\boldsymbol{x})$, the sum becomes an integral, and the addition is weighted by the density.

mean

Definition 6.3 (Mean)**.** The *mean* of a random variable $\boldsymbol{x} \in \mathbb{R}^D$ is defined as

$$\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \int \boldsymbol{x} p(\boldsymbol{x}) d\boldsymbol{x} = \begin{bmatrix} \mathbb{E}[x_1] \\ \vdots \\ \mathbb{E}[x_D] \end{bmatrix} \in \mathbb{R}^D \,, \tag{6.25}$$

where the subscript indicates the corresponding dimension of $\boldsymbol{x}$.

median
mode

In one dimension, there are two other intuitive notions of "average" which are the *median* and the *mode*. The median is the "middle" value if we sort the values, that is intuitively it is a typical value. For distributions which are asymmetric or has long tails, the median provides an estimate of a typical value that is closer to human intuition than the mean value. The mode is the most frequently occurring value, which is the highest peak in the density $p(\boldsymbol{x})$. A particular density $p(\boldsymbol{x})$ may have more than one mode, and therefore finding the mode may be computationally challenging in high dimensions.

The generalization of the median to higher dimensions is non-trivial, as there is no obvious way to "sort" in more than one dimension.

The definition of the mean (Definition 6.3), is actually a special case of an incredibly useful concept: the expected value.

expected value

Definition 6.4 (Expected value)**.** The *expected value* of a function g of a random variable $\boldsymbol{x} \sim p(\boldsymbol{x})$ is given by

$$\mathbb{E}_{\boldsymbol{x}}[g(\boldsymbol{x})] = \int g(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \,. \tag{6.26}$$

The mean is recovered if we set the function g in Definition 6.4 to the identity function. This indicates that we can think about functions of random variables, which we will revisit in Section 6.5.

The expected value of a function of a random variable is sometimes referred to as the law of the unconscious statistician (Casella and Berger, 2002, Section 2.2).

Remark. The expected value is a linear operator. For example given a univariate real valued function $f(x) = ag(x) + bh(x)$ where $a, b \in \mathbb{R}$,

$$\begin{aligned}\mathbb{E}_x[f(x)] &= \int f(x)p(x)dx && (6.27)\\ &= \int [ag(x) + bh(x)]p(x)dx && (6.28)\\ &= a\int g(x)p(x)dx + b\int h(x)p(x)dx && (6.29)\\ &= a\mathbb{E}_x[g(x)] + b\mathbb{E}_x[h(x)] && (6.30)\end{aligned}$$

This linear relationship holds in higher dimensions as well. ♢

For two random variables, we may wish to figure out their correspondence to each other.

Definition 6.5 (Covariance (univariate))**.** The covariance between two univariate random variables $x, y \in \mathbb{R}$ is given by the expected product of their deviations from their respective means, that is

$$\mathrm{Cov}[x, y] = \mathbb{E}\big[(x - \mathbb{E}[x])(y - \mathbb{E}[y])\big]\,. \tag{6.31}$$

By using the linearity of expectations, the expression in Definition 6.5 can be rewritten as the expected value of the product minus the product of the expected values

$$\mathrm{Cov}[x, y] = \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]\,. \tag{6.32}$$

The covariance of a variable with itself $\mathrm{Cov}[x, x]$ is called the *variance* and is denoted by $\mathbb{V}[x]$. The square root of the variance is called the *standard deviation* and is denoted $\sigma(x)$. variance standard deviation

The notion of covariance can be generalised to multivariate random variables.

Definition 6.6 (Covariance)**.** If we consider two random variables $\boldsymbol{x} \in \mathbb{R}^D, \boldsymbol{y} \in \mathbb{R}^E$, the *covariance* between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as covariance

$$\mathrm{Cov}[\boldsymbol{x}, \boldsymbol{y}] = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}\boldsymbol{y}^\top] - \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]\mathbb{E}_{\boldsymbol{y}}[\boldsymbol{y}]^\top = \mathrm{Cov}[\boldsymbol{y}, \boldsymbol{x}]^\top \in \mathbb{R}^{D\times E}\,. \tag{6.33}$$

Here, the subscript makes it explicit with respect to which variable we need to average.

Covariance intuitively represents the notion of how dependent random variables are to one another. We will revisit the idea of covariance again in Section 6.4.3

Definition 6.6 can be applied with the same multivariate random variable in both arguments, which results in a useful concept that intuitively captures the "spread" of a random variable.

Definition 6.7 (Variance)**.** The *variance* of a random variable $\boldsymbol{x} \in \mathbb{R}^D$ variance

with mean vector $\boldsymbol{\mu}$ is defined as

$$\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}] = \mathbb{E}_{\boldsymbol{x}}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top] = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}\boldsymbol{x}^\top] - \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]^\top \tag{6.34}$$

$$= \begin{bmatrix} \mathrm{Cov}[x_1,x_1] & \mathrm{Cov}[x_1,x_2] & \dots & \mathrm{Cov}[x_1,x_D] \\ \mathrm{Cov}[x_2,x_1] & \mathrm{Cov}[x_2,x_2] & \dots & \mathrm{Cov}[x_2,x_D] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[x_D,x_1] & \dots & \dots & \mathrm{Cov}[x_D,x_D] \end{bmatrix} \in \mathbb{R}^{D\times D}\,. \tag{6.35}$$

covariance matrix

This matrix is called the *covariance matrix* of the random variable $\boldsymbol{x}$. The covariance matrix is symmetric and positive definite and tells us something about the spread of the data.

The covariance matrix contains the variances of the marginals $p(x_i) = \int p(x_1,\dots,x_D)\mathrm{d}x_{\setminus i}$ on its diagonal, where "$\setminus i$" denotes "all variables but i". The off-diagonal terms contain the *cross-covariance* terms $\mathrm{Cov}[x_i,x_j]$ for $i,j=1,\dots,D,\ i\neq j$.

cross-covariance

It generally holds that

$$\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}] = \mathrm{Cov}_{\boldsymbol{x}}[\boldsymbol{x},\boldsymbol{x}]\,. \tag{6.36}$$

population mean and covariance

The definitions above are often also called the *population mean and covariance*. For a particular set of data we can obtain an estimate of the mean, which is called the *empirical mean* or *sample mean*. The same holds for the empirical covariance.

empirical mean
sample mean

Definition 6.8 (Empirical Mean and Covariance)**.** The *empirical mean* vector is the arithmetic average of the observations for each variable, and is written

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\,. \tag{6.37}$$

empirical covariance

The *empirical covariance* is a $K\times K$ matrix

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n-\bar{\boldsymbol{x}})(\boldsymbol{x}_n-\bar{\boldsymbol{x}})^\top. \tag{6.38}$$

Empirical covariance matrices are positive semi-definite (see Section 3.2.3).

We use the sample covariance in this book. The unbiased (sometimes called corrected) covariance has the factor $N-1$ in the denominator.

6.4.2 Three Expressions for the Variance

We now focus on a single random variable x, and use the empirical formulas above to derive three possible expressions for the variance. The derivation below is the same for the population variance, except that one needs to take care of integrals. The standard definition of variance, corresponding to the definition of covariance (Definition 6.5), is the expectation of

the squared deviation of a random variable x from its expected value. That is

$$\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2 \tag{6.39}$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean. Observe that the variance as expressed above is the mean of a new random variable $z = (x-\mu)^2$.

When estimating this empirically, we need to resort to a two pass algorithm: one pass through the data to calculate the mean μ using (6.37), and then a second pass using this estimate $\hat{\mu}$ calculate the variance. It turns out that we can avoid two passes by rearranging the terms. The formula in (6.39) can be converted to the so called raw score formula for variance

$$\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2 = \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N}x_i\right)^2 . \tag{6.40}$$

This expression in (6.40) can be remembered as "the mean of the square minus the square of the mean". It can be calculated in one pass through data since we can accumulate x_i (to calculate the mean) and x_i^2 simultaneously. Unfortunately if implemented in this way, it is numerically unstable. The raw score version of the variance can be useful in machine learning, for example when deriving the bias-variance decomposition (Bishop, 2006).

The two terms can cancel out, resulting is loss of numerical precision in floating point arithmetic.

A third way to understand the variance is that it is a sum of pairwise differences between all pairs of observations. By expanding the square we can show that the sum of pairwise differences is two times the raw score expression,

$$\frac{1}{N^2}\sum_{i,j=1}^{N}(x_i-x_j)^2 = 2\left[\frac{1}{N}\sum_{i=1}^{N}x_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N}x_i\right)^2\right] \tag{6.41}$$

Observe that (6.41) is twice of (6.40). This means that we can express the sum of pairwise distances (of which there are N^2 of them) as a sum of deviations from the mean (of which there are N). Geometrically, this means that there is an equivalence between the pairwise distances and the distances from the center of the set of points.

6.4.3 Statistical Independence

Definition 6.9 (Independence)**.** Two random variables $\boldsymbol{x}, \boldsymbol{y}$ are *statistically independent* if and only if

statistically independent

$$p(\boldsymbol{x},\boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y})\,. \tag{6.42}$$

Intuitively, two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent if the value

of $\boldsymbol{y}$ (once known) does not add any additional information about $\boldsymbol{x}$ (and vice versa).

If $\boldsymbol{x}, \boldsymbol{y}$ are (statistically) independent then

- $p(\boldsymbol{y} \mid \boldsymbol{x}) = p(\boldsymbol{y})$
- $p(\boldsymbol{x} \mid \boldsymbol{y}) = p(\boldsymbol{x})$
- $\mathbb{V}[\boldsymbol{x} + \boldsymbol{y}] = \mathbb{V}[\boldsymbol{x}] + \mathbb{V}[\boldsymbol{y}]$
- $\mathrm{Cov}[\boldsymbol{x}, \boldsymbol{y}] = \mathbf{0}$

Note that the last point above may not hold in converse, that is two random variables can have covariance zero but are not statistically independent.

correlation

Remark. Let us briefly mention the relationship between *correlation* and *covariance*. The correlation matrix is the covariance matrix of standardized random variables, $x/\sigma(x)$. In other words, each random variable is divided by its standard deviation (the square root of the variance) in the correlation matrix. ♢

Another concept that is important in machine learning is conditional independence.

conditionally independent given $\boldsymbol{z}$

Definition 6.10 (Conditional Independence)**.** Formally, $\boldsymbol{x}$ and $\boldsymbol{y}$ are *conditionally independent given* $\boldsymbol{z}$ if and only if

$$p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{y} \mid \boldsymbol{z}) \,. \tag{6.43}$$

We write $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \mid \boldsymbol{z}$.

Note that the definition of conditional independence above requires that the relation in Equation (6.43) must hold true for every value of $\boldsymbol{z}$. The interpretation of Equation (6.43) above can be understood as "given knowledge about $\boldsymbol{z}$, the distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$ factorizes". Independence can be cast as a special case of conditional independence if we write $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \mid \emptyset$.

By using the product rule of probability (Equation (6.20)), we can expand the left hand side of Equation 6.43 to obtain

$$p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) p(\boldsymbol{y} \mid \boldsymbol{z}). \tag{6.44}$$

By comparing the right hand side of Equation (6.43) with Equation (6.44), we see that $p(\boldsymbol{y} \mid \boldsymbol{z})$ appears in both, and therefore

$$p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}). \tag{6.45}$$

Equation (6.45) above provides an alternative definition of conditional independence, that is $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \mid \boldsymbol{z}$. This alternative presentation provides the interpretation: "given that we know $\boldsymbol{z}$, knowledge about $\boldsymbol{y}$ does not change our knowledge of $\boldsymbol{x}$".

6.4.4 Sums and Transformations of Random Variables

We may want to model a phenomenon that cannot be well explained by textbook distributions (we introduce some in Section 6.6 and 6.7), and hence may perform simple manipulations of random variables (such as adding two random variables).

Consider two random variables $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$. It holds that

$$\mathbb{E}[\boldsymbol{x}+\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}] + \mathbb{E}[\boldsymbol{y}] \tag{6.46}$$
$$\mathbb{E}[\boldsymbol{x}-\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}] - \mathbb{E}[\boldsymbol{y}] \tag{6.47}$$
$$\mathbb{V}[\boldsymbol{x}+\boldsymbol{y}] = \mathbb{V}[\boldsymbol{x}] + \mathbb{V}[\boldsymbol{y}] + \mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] + \mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}] \tag{6.48}$$
$$\mathbb{V}[\boldsymbol{x}-\boldsymbol{y}] = \mathbb{V}[\boldsymbol{x}] + \mathbb{V}[\boldsymbol{y}] - \mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] - \mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}] \tag{6.49}$$

Mean and (co)variance exhibit some useful properties when it comes to affine transformation of random variables. Consider a random variable $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ and a (deterministic) affine transformation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ of $\boldsymbol{x}$. Then $\boldsymbol{y}$ is itself a random variable whose mean vector and covariance matrix are given by

$$\mathbb{E}_{\boldsymbol{y}}[\boldsymbol{y}] = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}] = \boldsymbol{A}\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] + \boldsymbol{b} = \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}\,, \tag{6.50}$$
$$\mathbb{V}_{\boldsymbol{y}}[\boldsymbol{y}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x}] = \boldsymbol{A}\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}]\boldsymbol{A}^\top = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top\,, \tag{6.51}$$

respectively. Furthermore,

This can be shown directly by using the definition of the mean and covariance.

$$\begin{aligned}\mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] &= \mathbb{E}[\boldsymbol{x}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})^\top] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}]^\top && (6.52)\\ &= \mathbb{E}[\boldsymbol{x}]\boldsymbol{b}^\top + \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]\boldsymbol{A}^\top - \boldsymbol{\mu}\boldsymbol{b}^\top - \boldsymbol{\mu}\boldsymbol{\mu}^\top\boldsymbol{A}^\top && (6.53)\\ &= \boldsymbol{\mu}\boldsymbol{b}^\top - \boldsymbol{\mu}\boldsymbol{b}^\top + \left(\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top\right)\boldsymbol{A}^\top && (6.54)\\ &\overset{(6.34)}{=} \boldsymbol{\Sigma}\boldsymbol{A}^\top\,, && (6.55)\end{aligned}$$

where $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top$ is the covariance of $\boldsymbol{x}$.

6.4.5 Inner Products of Random Variables

Recall the definition of inner products from Section 3.2. Another example for defining an inner product between unusual types are random variables or random vectors. If we have two uncorrelated random variables x, y then

$$\mathbb{V}[x+y] = \mathbb{V}[x] + \mathbb{V}[y] \tag{6.56}$$

Since variances are measured in squared units, this looks very much like the Pythagorean theorem for right triangles $c^2 = a^2 + b^2$.

In the following, we see whether we can find a geometric interpretation of the variance relation of uncorrelated random variables in (6.56).

Random variables can be considered vectors in a vector space, and we

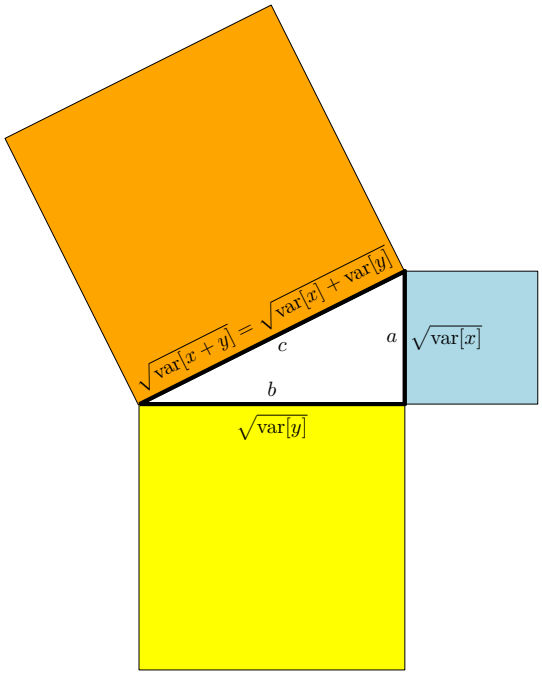


Figure 6.4 Geometry of random variables. If random variables x and y are uncorrelated they are orthogonal vectors in a corresponding vector space, and the Pythagorean theorem applies.

can define inner products to obtain geometric properties of random variables. If we define

$$\langle x, y \rangle := \mathrm{Cov}[x, y] \tag{6.57}$$

we see that the covariance is symmetric, positive definite[1], and linear in either argument[2] The length of a random variable is

$$\|x\| = \sqrt{\mathrm{Cov}[x, x]} = \sqrt{\mathbb{V}[x]} = \sigma[x]\,, \tag{6.58}$$

i.e., its standard deviation. The "longer" the random variable, the more uncertain it is; and a random variable with length 0 is deterministic.

If we look at the angle θ between random two random variables x, y, we get

$$\cos\theta = \frac{\langle x, y \rangle}{\|x\|\,\|y\|} = \frac{\mathrm{Cov}[x, y]}{\sqrt{\mathbb{V}[x]\mathbb{V}[y]}}\,. \tag{6.59}$$

We know from Definition 3.6 that $x \perp y \iff \langle x, y \rangle = 0$. In our case this means that x and y are orthogonal if and only if $\mathrm{Cov}[x, y] = 0$, i.e., they are uncorrelated. Figure 6.4 illustrates this relationship.

Remark. While it is tempting to use the Euclidean distance (constructed from the definition of inner products above) to compare probability distributions, it is unfortunately not the best way to obtain distances between

[1] $\mathrm{Cov}[x, x] > 0$ and $0 \iff x = 0$
[2] $\mathrm{Cov}[\alpha x + z, y] = \alpha\,\mathrm{Cov}[x, y] + \mathrm{Cov}[z, y]$ for $\alpha \in \mathbb{R}$.

distributions. Due to the fact that the probability mass (or density) needs to add up to 1, distributions live in a subspace which is called a manifold. The study of this space of probability distributions is called information geometry. Computing distances between distributions are done using Bregman divergences or f-divergences, which is beyond the scope of this book. Interested readers are referred to a recent book (Amari, 2016) written by one of the founders of the field of information geometry. ♢

6.5 Change of Variables/Inverse transform

It may seem that there are very many known distributions to a beginner, but in reality the set of distributions for which we have names are quite limited. Therefore it is often useful to understand how transformations of random variables are distributed. For example, assume that x is a random variable distributed according to the univariate normal distribution $\mathcal{N}(0, 1)$, what is the distribution of x^2? Another example which is quite common in machine learning is: given that x_1 and x_2 are univariate standard normal, what is the distribution of $\frac{1}{2}(x_1 + x_2)$?

Remark. One option to work out the distribution of $\frac{1}{2}(x_1 + x_2)$ is to calculate the mean and variance of x_1 and x_2 and then combine them. As we saw in Section 6.4.4, we can calculate the mean and covariance of resulting random variables when we consider affine transformations of random variables. However we may not be able to obtain the functional form of the distribution under transformations. Furthermore we may be interested in other transformations (for example nonlinear) of random variables. ♢

In this section, we need to be explicit about random variables and the values they take, and hence we will use small letters x, y to denote random variables and small capital letters X, Y to denote the values that the random variables take. We will look at two approaches for obtaining distributions of transformations of random variables: a direct approach using the definition of a cumulative distribution function; and a change of variable approach that uses the chain rule of calculus (Section 5.2.2). The change of variable approach is widely used because it provides a "recipe" for attempting to compute the resulting distribution due to a transformation. We will explain the techniques for univariate random variables, and will only briefly provide the results for the general case of multivariate random variables.

One can also use the moment generating function to study transformations of random variables (Casella and Berger, 2002, Chapter 2).

As mentioned in the introductory comments in this chapter, random variables and probability distributions are closely associated with each other. It is worth carefully teasing apart the two ideas, and in doing so we will motivate why we need to transform random variables.

Example 6.5
It is worth contrasting this example with the example in Section 6.2.1. Consider a medical test the returns the number of cancerous cells that can be found in the biopsy. The state space the the set of non-negative integers. The random variable x is the *square* of the number of cancerous cells. Given that we know the probability distribution corresponding to the number of cancerous cells in a biopsy, how do we obtain the distribution of random variable x?

Remark. An analogy to object oriented programming may provide an alternative view for computer scientists. The distinction between random variables and probability distributions can be thought of as the distinction between objects and classes. A probability distribution defines the behaviour (the probability) corresponding to a particular statistical experiment, which quantifies the uncertainty associated with the experiment. A random variable is a particular instantiation of this statistical experiment, which follows the probabilities defined by the distribution. ♢

Transformations of discrete random variables can be understood directly. Given a discrete random variable x with probability mass function $p_x(\mathrm{x})$ (Section 6.2.1), and an invertible function $g(x)$ with inverse $h(\cdot)$. Let y be the random variable transformed by $g(x)$, that is $y = g(x)$. Then

$$p_y(\mathrm{y}) = p_x(h(\mathrm{y})). \tag{6.60}$$

This can be seen by the following short derivation,

$$\begin{aligned} p_y(\mathrm{y}) &= P(y = \mathrm{y}) && \text{definition of pmf} && (6.61)\\ &= P(g(x) = \mathrm{y}) && \text{transformation of interest} && (6.62)\\ &= P(x = h(\mathrm{y})) && \text{inverse} && (6.63)\\ &= p_x(h(\mathrm{y})) && \text{definition of pmf.} && (6.64) \end{aligned}$$

Therefore for discrete random variables, transformations directly change the individual probability of events. The following discussion focuses on continuous random variables and we will need both probability density functions $p(x)$ and cumulative distribution functions $P(x \leqslant \mathrm{x})$.

6.5.1 Distribution Function Technique

The distribution function technique goes back to first principles, and uses the definition of a cumulative distribution function (cdf) and the fact that its differential is the probability density function (pdf) (Wasserman, 2004, Chapter 2). For a random variable x, and a function U, we find the pdf of the random variable $y = U(x)$ by

1. finding the cdf:

$$F_y(\mathrm{Y}) = P(y \leqslant \mathrm{Y}) \tag{6.65}$$

2. then differentiating the cdf $F_y(\mathrm{Y})$ to get the pdf $f(y)$.

$$f(y) = \frac{d}{dy} F_y(\mathrm{Y}) \tag{6.66}$$

We also need to keep in mind that the domain of the random variable may have changed due to the transformation.

Example 6.6
Let x be a continuous random variable with the following probability density function on $0 < x < 1$

$$f(x) = 3x^2. \tag{6.67}$$

What is the pdf of $y = x^2$?

Note that the function f is an increasing function of x and also the resulting value of y is in the interval $(0, 1)$.

$$\begin{aligned}
F_y(\mathrm{Y}) &= P(y \leqslant \mathrm{Y}) && \text{definition of cdf} && (6.68)\\
&= P(x^2 \leqslant \mathrm{Y}) && \text{transformation of interest} && (6.69)\\
&= P(x \leqslant \mathrm{Y}^{\frac{1}{2}}) && \text{inverse} && (6.70)\\
&= P_x(\mathrm{Y}^{\frac{1}{2}}) && \text{definition of cdf} && (6.71)\\
&= \int_0^{\mathrm{Y}^{\frac{1}{2}}} 3t^2 \mathrm{d}t && \text{cdf as a definite integral} && (6.72)\\
&= \left[t^3\right]_{t=0}^{t=\mathrm{Y}^{\frac{1}{2}}} && \text{result of integration} && (6.73)\\
&= \mathrm{Y}^{\frac{3}{2}}, \quad 0 < \mathrm{Y} < 1 && && (6.74)
\end{aligned}$$

Therefore the cdf of y is

$$F_y(\mathrm{Y}) = \mathrm{Y}^{\frac{3}{2}} \tag{6.75}$$

for $0 < \mathrm{Y} < 1$. To obtain the pdf, we differentiate the cdf

$$f_y(\mathrm{Y}) = \frac{d}{d\mathrm{Y}} F_y(\mathrm{Y}) = \frac{3}{2}\mathrm{Y}^{\frac{1}{2}} \tag{6.76}$$

for $0 < \mathrm{Y} < 1$.

In the previous example, we considered a monotonically increasing function x^2. This means that we could compute an inverse function. In general we require that the function of interest, $y = U(x)$ has an inverse $x = U^{-1}(y)$. One useful result that can be obtained by applying the technique above when the transformation of interest is the cumulative distri-

Functions that have inverses are called injective functions (Section 2.7).

bution function of the random variable itself (Casella and Berger, 2002, Theorem 2.1.10).

Theorem 6.11. *Let x be a continous random variable with cumulative distribution function $F_x(\cdot)$. Then the random variable y defined as*

$$y = F_x(\mathrm{X}), \tag{6.77}$$

has a uniform distribution.

Proof We need to show that the cumulative distribution function (cdf) of y defines a distribution of a uniform random variable. Recall that by the axioms of probability (Section 6.1) that probabilities must be non-negative and sum to one. Therefore the range of possible values of $y = F_x(\mathrm{X})$ is in the interval $[0, 1]$. Note that for any $F_x(\cdot)$, the inverse $F_x^{-1}(\cdot)$ exists because cdfs are monotone increasing, which we will use in the following proof. Given any continuous random variable x, the definition of a cdf gives

$$\begin{aligned}
F_y(\mathrm{Y}) &= P(y \leqslant \mathrm{Y}) && && (6.78)\\
&= P(F_x(\mathrm{X}) \leqslant \mathrm{Y}) && \text{transformation of interest} && (6.79)\\
&= P(x \leqslant F_x^{-1}(\mathrm{Y})) && \text{inverse exists} && (6.80)\\
&= F_x(F_x^{-1}(\mathrm{Y})) && \text{definition of cdf} && (6.81)\\
&= \mathrm{Y}, && && (6.82)
\end{aligned}$$

where the last line is due to the fact that $F_x(\cdot)$ composed with its inverse results in an identity transformation. The statement $F_y(\mathrm{Y}) = \mathrm{Y}$ along with the fact that y lies in the interval $[0, 1]$ means that $F_y(\cdot)$ is the cdf of the uniform random variable on the unit interval. □

probability integral transform

This result (Theorem 6.11) is known as the *probability integral transform*, and is used to derive algorithms for sampling from distributions by transforming the result of sampling from a uniform random variable. It is also used for hypothesis testing whether a sample comes from a particular distribution (Lehmann and Romano, 2005). The idea that the output of a cdf gives a uniform distribution also forms the basis of copulas (Nelsen, 2006).

6.5.2 Change of Variables

The argument from first principles in the previous section relies on two facts:

1. We can transform the cdf of y into an expression that is a cdf of x.
2. We can differentiate the cdf to obtain the pdf.

Let us break down the reasoning step by step, with the goal of deriving a more general approach called change of variables.

Consider a function of a random variable $y = U(x)$ where x lies in the interval $a < x < b$. By the definition of the cdf, we have

$$F_y(\mathrm{Y}) = P(y \leqslant \mathrm{Y})\,. \tag{6.83}$$

We are interested in a function U of the random variable

$$P(y \leqslant \mathrm{Y}) = P(U(x) \leqslant \mathrm{Y})\,, \tag{6.84}$$

and we assume that the function U is invertible. By multiplying both sides with the inverse

$$P(U(x) \leqslant y) = P(U^{-1}(U(x)) \leqslant U^{-1}(\mathrm{Y})) = P(x \leqslant U^{-1}(\mathrm{Y})) \tag{6.85}$$

we obtain an expression of the cdf of x. Recall the definition of the cdf in terms of the pdf

$$P(x \leqslant U^{-1}(\mathrm{Y})) = \int_a^{U^{-1}(\mathrm{Y})} f(x)\mathrm{d}x\,. \tag{6.86}$$

Now we have an expression of the cdf of y in terms of x.

$$F_y(\mathrm{Y}) = \int_a^{U^{-1}(\mathrm{Y})} f(x)\mathrm{d}x \tag{6.87}$$

To obtain the pdf, we differentiate the expression above with respect to y. Since the expression is in terms of x, we apply the chain rule of calculus from (5.56) and obtain

$$f_y(\mathrm{Y}) = \frac{d}{d\mathrm{Y}} F_y(\mathrm{Y}) = \frac{d}{d\mathrm{Y}} \int_a^{U^{-1}(\mathrm{Y})} f(x)\mathrm{d}x \tag{6.88}$$

$$= f_x(U^{-1}(\mathrm{Y})) \times \left| \det\left(\frac{d}{d\mathrm{Y}} U^{-1}(\mathrm{Y}) \right) \right|. \tag{6.89}$$

The fact that integration and differentiation are somehow "inverses" of each other is due to a deep result called the Fundamental Theorem of Calculus.

This is called the *change of variable* technique. The term $\left|\frac{d}{d\mathrm{Y}} U^{-1}(\mathrm{Y})\right|$ measures how much a unit volume changes when applying U. Recall from Section 4.1 that the existence of the determinant shows that we can invert the Jacobian. Recall further that the determinant arises because our differentials (cubes of volume) are transformed into parallelepipeds by the determinant. In the last expression above, we have introduced the absolute value of the differential. For decreasing functions, it turns out that an additional negative sign is needed, and instead of having two types of change of variable rules, the absolute value unifies both of them.

change of variable

Remark. Observe that in comparison to the discrete case in Equation (6.60), we have an additional factor $\left|\frac{d}{dy} U^{-1}(y)\right|$. The continuous case requires more care because $P(y = \mathrm{Y}) = 0$ for all Y. The probability density function $f_y(\mathrm{Y})$ does not have a description as a probability of an event involving y. ♢

So far in this section we have been studying univariate change of variables. The case for multivariate random variables is analogous, but complicated by fact that the absolute value cannot be used for multivariate functions. Instead we use the determinant of the Jacobian matrix. Recall from Equation (5.68) that the Jacobian is a matrix of partial derivatives. Let us summarize the discussion above in the following theorem which describes the recipe for multivariate change of variables.

Theorem 6.12. *Let $f_{\boldsymbol{x}}(\mathrm{X})$ be the value of the probability density of the multivariate continuous random variable $\boldsymbol{x}$ at X. If the vector valued function $\boldsymbol{y} = U(\boldsymbol{x})$ is differentiable and invertible for all values within the range of $\boldsymbol{x}$, then for corresponding values of $\boldsymbol{y}$, the probability density of $y = U(x)$ is given by*

$$f_{\boldsymbol{y}}(\mathrm{Y}) = f_{\boldsymbol{x}}(U^{-1}(\mathrm{Y})) \times \left| \det \left(\frac{\partial}{\partial \mathrm{Y}} U^{-1}(\mathrm{Y}) \right) \right| . \tag{6.90}$$

The theorem looks intimidating at first glance, but we only need to understand that a change of variable of a multivariate random variable follows the procedure of the univariate change of variable. That is first we need to work out the inverse transform, and substitute that into the density of $\boldsymbol{x}$. Then calculate the determinant of the Jacobian and multiply the result. The following example illustrates the case of a bivariate random variable.

Example 6.7

Consider a bivariate random variable $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ with probability density function

$$f_{\boldsymbol{x}} \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \frac{1}{2\pi} \exp \left(-\frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) . \tag{6.91}$$

We use the change of variable technique (Theorem 6.12) to derive the effect of an linear transformation (Section 2.7) of the random variables. Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ defined as

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} . \tag{6.92}$$

What is the probability density function of the resulting transformed bivariate random variable $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$?

Recall that for change of variables, we require the inverse transformation of $\boldsymbol{x}$ as a function of $\boldsymbol{y}$. Since we are considering linear transformations, the inverse transformation is given matrix inverse from Section 2.2.2. For 2×2 matrices, we can explicitly write out the formula,

given by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{A}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}. \tag{6.93}$$

Observe that $ad - bc$ is the determinant (Section 4.1) of $\boldsymbol{A}$. The corresponding probability density function is given by

$$f_{\boldsymbol{x}}(\mathrm{X}) = f_{\boldsymbol{x}}(\boldsymbol{A}^{-1}\mathrm{Y}) \tag{6.94}$$

$$= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\mathrm{Y}^\top \boldsymbol{A}^{-\top}\boldsymbol{A}^{-1}\mathrm{Y}\right). \tag{6.95}$$

The partial derivative of a matrix times a vector with respect to the vector is the matrix itself (Section 5.5) and therefore

$$\frac{\partial}{\partial \mathrm{Y}} \boldsymbol{A}^{-1}\mathrm{Y} = \boldsymbol{A}^{-1}. \tag{6.96}$$

Recall from Section 4.1 that the determinant of the inverse is the inverse of the determinant, and therefore the determinant of the Jacobian matrix is given by

$$\left|\tfrac{\partial}{\partial \mathrm{Y}} \boldsymbol{A}^{-1}\mathrm{Y}\right| = ad - bc. \tag{6.97}$$

We are now able to apply the change of variable formula from Theorem 6.12, by multiplying Equation (6.95) with Equation (6.97),

$$f_y(\mathrm{Y}) = f_x(\mathrm{X}) \times \left|\left|\tfrac{\partial}{\partial \mathrm{Y}} \boldsymbol{A}^{-1}\mathrm{Y}\right|\right| \tag{6.98}$$

$$= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\mathrm{Y}^\top \boldsymbol{A}^{-\top}\boldsymbol{A}^{-1}\mathrm{Y}\right)(ad - bc). \tag{6.99}$$

While the example above is based on a bivariate random variable so that we can compute the matrix inverse in closed form, the relation above holds true for higher dimensions.

Remark. We will see in Section 6.6 that the density $f_{\boldsymbol{x}}(\mathrm{X})$ above is actually the standard Gaussian distribution, and the transformed density $f_{\boldsymbol{y}}(\mathrm{Y})$ is a bivariate Gaussian with covariance $\boldsymbol{\Sigma} = \boldsymbol{A}^\top \boldsymbol{A}$. The linear transformation $\boldsymbol{A}$ turns out to correspond to the Cholesky factorization (Section 4.3) of $\boldsymbol{\Sigma}$. ♢

6.6 Gaussian Distribution

The Gaussian distribution is the most important probability distribution for continuous-valued random variables. It is also referred to as the *normal distribution*. Its importance originates from the fact that it has many computationally convenient properties, which we will be discussing in the following. In particular, we will use it to define the likelihood and prior for

The Gaussian distribution arises naturally when we consider sums of independent and identically distributed random variables. This is known as the Central Limit Theorem (Grinstead and Snell, 1997).

normal distribution

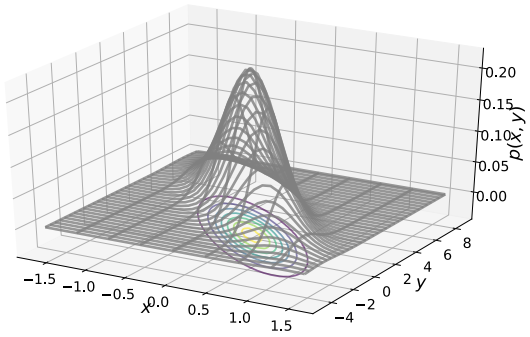


Figure 6.5 Gaussian distribution of two random variables x, y.

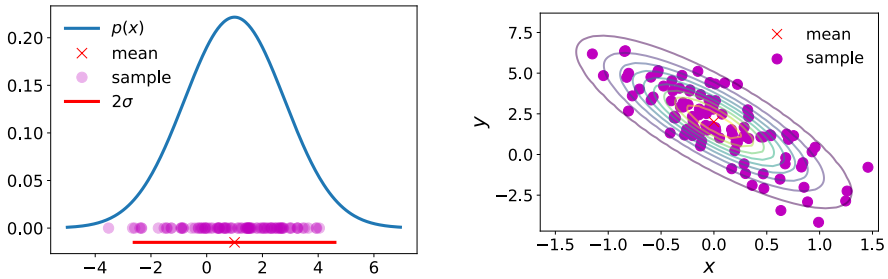


Figure 6.6 Gaussian distributions overlaid with 100 samples. Left: Univariate (1-dimensional) Gaussian; The red cross shows and mean and the red line the extent of the variance. Right: Multivariate (2-dimensional) Gaussian, viewed from top. The red cross shows the mean and the coloured lines contour lines of the density.

linear regression (Chapter 9), and consider a mixture of Gaussians for density estimation (Chapter 11).

There are many other areas of machine learning that also benefit from using a Gaussian distribution, for example Gaussian processes, variational inference and reinforcement learning. It is also widely used in other application areas such as signal processing (e.g., Kalman filter), control (e.g., linear quadratic regulator) and statistics (e.g. hypothesis testing).

For a univariate random variable, the Gaussian distribution has a density that is given by

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{6.100}$$

multivariate Gaussian distribution
Also: multivariate normal distribution
mean vector
covariance matrix

The *multivariate Gaussian distribution* is fully characterized by a *mean vector* $\boldsymbol{\mu}$ and a *covariance matrix* $\boldsymbol{\Sigma}$ and defined as

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{6.101}$$

where $\boldsymbol{x} \in \mathbb{R}^D$ is a random variable. We write $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Figure 6.5 shows a bi-variate Gaussian (mesh), with the corresponding contour plot. The special case of the Gaussian with zero mean and identity variance, that is $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$, is referred to as the *standard normal distribution*.

standard normal distribution

Gaussian distributions are widely used in statistical estimation and ma-

chine learning because they have closed-form expressions for marginal and conditional distributions. In Chapter 9, we use these closed form expressions extensively for linear regression. A major advantage of modelling with Gaussian distributed random variables is that variable transformations (Section 6.5) are often not needed. Since the Gaussian distribution is fully specified by its mean and covariance we often can obtain the transformed distribution by applying the transformation to the mean and covariance of the random variable.

6.6.1 Marginals and Conditionals of Gaussians are Gaussians

In the following, we present marginalization and conditioning in the general case of multivariate random variables. If this is confusing at first reading, the reader is advised to consider two univariate random variables instead. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two multivariate random variables, which may have different dimensions. We would like to consider the effect of applying the sum rule of probability and the effect of conditioning. We therefore explicitly write the Gaussian distribution in terms of the concatenated random variable $[\boldsymbol{x}, \boldsymbol{y}]^\top$,

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy}\end{bmatrix}\right). \tag{6.102}$$

where $\boldsymbol{\Sigma}_{xx} = \mathrm{Cov}[\boldsymbol{x}, \boldsymbol{x}]$ and $\boldsymbol{\Sigma}_{yy} = \mathrm{Cov}[\boldsymbol{y}, \boldsymbol{y}]$ are the marginal covariance matrices of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and $\boldsymbol{\Sigma}_{xy} = \mathrm{Cov}[\boldsymbol{x}, \boldsymbol{y}]$ is the cross-covariance matrix between $\boldsymbol{x}$ and $\boldsymbol{y}$.

The conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ is also Gaussian (illustrated on the bottom right of Figure 6.7) and given by

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{\mu}_{x \mid y}, \boldsymbol{\Sigma}_{x \mid y}\big) \tag{6.103}$$

$$\boldsymbol{\mu}_{x \mid y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y) \tag{6.104}$$

$$\boldsymbol{\Sigma}_{x \mid y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\,. \tag{6.105}$$

Note that in the computation of the mean in (6.104) the $\boldsymbol{y}$-value is an observation and no longer random.

Remark. The conditional Gaussian distribution shows up in many places, where we are interested in posterior distributions:

- The Kalman filter (Kalman, 1960), one of the most central algorithms for state estimation in signal processing, does nothing but computing Gaussian conditionals of joint distributions (Deisenroth and Ohlsson, 2011).
- Gaussian processes (Rasmussen and Williams, 2006), which are a practical implementation of a distribution over functions. In a Gaussian process, we make assumptions of joint Gaussianity of random variables. By

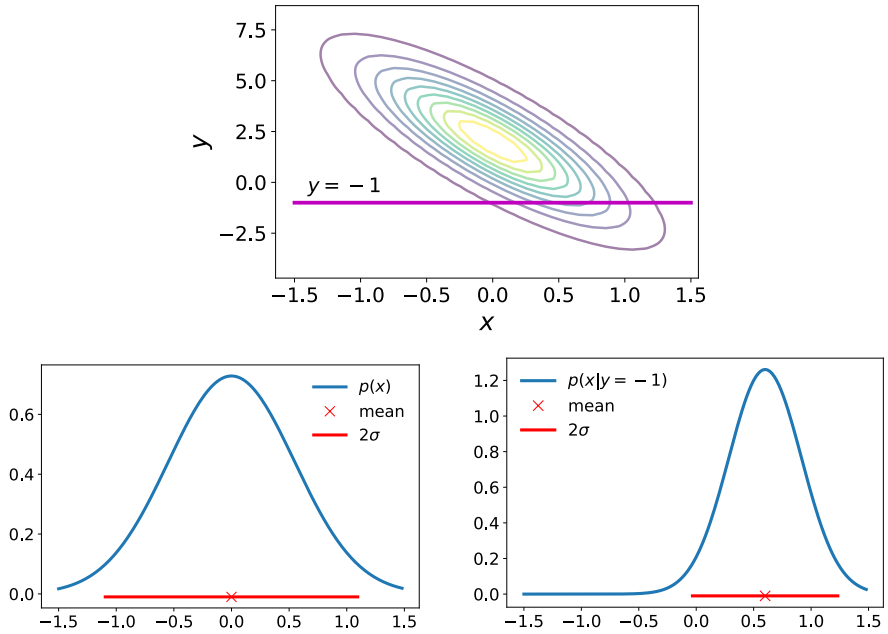


Figure 6.7 Top: Bivariate Gaussian; Bottom left: Marginal of a joint Gaussian distribution is Gaussian; Bottom right: The conditional distribution of a Gaussian is also Gaussian

(Gaussian) conditioning on observed data, we can determine a posterior distribution over functions.

- Latent linear Gaussian models (Roweis and Ghahramani, 1999; Murphy, 2012), which include probabilistic PCA (Tipping and Bishop, 1999).

♢

The marginal distribution $p(\boldsymbol{x})$ of a joint Gaussian distribution $p(\boldsymbol{x}, \boldsymbol{y})$, see (6.102), is itself Gaussian and computed by applying the sum-rule in (6.18) and given by

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} = \mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{\mu}_x, \, \boldsymbol{\Sigma}_{xx}\right). \tag{6.106}$$

The corresponding result holds for $p(\boldsymbol{y})$, which is obtained by marginalizing with respect to $\boldsymbol{x}$. Intuitively, looking at the joint distribution in (6.102), we ignore (i.e., integrate out) everything we are not interested in. This is illustrated on the bottom left of Figure 6.7.

Example 6.8

Consider the bivariate Gaussian distribution (illustrated in Figure 6.7)

$$p(x, y) = \mathcal{N}\left(\begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0.3 & -1\\-1 & 5\end{bmatrix}\right). \tag{6.107}$$

We can compute the parameters of the univariate Gaussian, conditioned

on $y = -1$, by applying (6.104) and (6.105) to obtain the mean and variance respectively. Numerically, this is

$$\mu_{x\,|\,y=-1} = 0 + (-1)(0.2)(-1-2) = 0.6 \tag{6.108}$$

and

$$\sigma^2_{x\,|\,y=-1} = 0.3 - (-1)(0.2)(-1) = 0.1. \tag{6.109}$$

Therefore the conditional Gaussian is given by

$$p(x\,|\,y=-1) = \mathcal{N}\big(0.6,\, 0.1\big). \tag{6.110}$$

The marginal distribution $p(x)$ in contrast can be obtained by applying (6.106), which is essentially using the mean and variance of the random variable x, giving us

$$p(x) = \mathcal{N}\big(0,\, 0.3\big) \tag{6.111}$$

6.6.2 Product of Gaussians

In machine learning, we often assume that examples are perturbed by Gaussian noise, leading to a Gaussian likelihood for linear regression. Furthermore we may wish to assume a Gaussian prior (Section 9.3). The application of Bayes rule to compute the posterior results in a multiplication of the likelihood and the prior, that is the multiplication of two Gaussians. The *product* of two Gaussians $\mathcal{N}\big(\boldsymbol{x}\,|\,\boldsymbol{a},\,\boldsymbol{A}\big)\mathcal{N}\big(\boldsymbol{x}\,|\,\boldsymbol{b},\,\boldsymbol{B}\big)$ is an unnormalized Gaussian distribution $c\mathcal{N}\big(\boldsymbol{x}\,|\,\boldsymbol{c},\,\boldsymbol{C}\big)$ with

$$\boldsymbol{C} = (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1} \tag{6.112}$$

$$\boldsymbol{c} = \boldsymbol{C}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}) \tag{6.113}$$

$$c = (2\pi)^{-\frac{D}{2}}|\boldsymbol{A}+\boldsymbol{B}|^{-\frac{1}{2}} \exp\big(-\tfrac{1}{2}(\boldsymbol{a}-\boldsymbol{b})^\top(\boldsymbol{A}+\boldsymbol{B})^{-1}(\boldsymbol{a}-\boldsymbol{b})\big)\,. \tag{6.114}$$

Note that the normalizing constant c itself can be considered a (normalized) Gaussian distribution either in $\boldsymbol{a}$ or in $\boldsymbol{b}$ with an "inflated" covariance matrix $\boldsymbol{A}+\boldsymbol{B}$, i.e., $c = \mathcal{N}\big(\boldsymbol{a}\,|\,\boldsymbol{b},\,\boldsymbol{A}+\boldsymbol{B}\big) = \mathcal{N}\big(\boldsymbol{b}\,|\,\boldsymbol{a},\,\boldsymbol{A}+\boldsymbol{B}\big)$.

Remark. For notation convenience, we will sometimes use $\mathcal{N}\big(\boldsymbol{x}\,|\,\boldsymbol{m},\,\boldsymbol{S}\big)$ to describe the functional form of a Gaussian even if $\boldsymbol{x}$ is not a random variable. We have just done this above when we wrote

$$c = \mathcal{N}\big(\boldsymbol{a}\,|\,\boldsymbol{b},\,\boldsymbol{A}+\boldsymbol{B}\big) = \mathcal{N}\big(\boldsymbol{b}\,|\,\boldsymbol{a},\,\boldsymbol{A}+\boldsymbol{B}\big)\,. \tag{6.115}$$

Here, neither $\boldsymbol{a}$ nor $\boldsymbol{b}$ are random variables. However, writing c in this way is more compact than (6.114). ♢

6.6.3 Sums and Linear Transformations

If $\boldsymbol{x}, \boldsymbol{y}$ are independent Gaussian random variables (i.e., the joint is given as $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y})$) with $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ and $p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, then $\boldsymbol{x} + \boldsymbol{y}$ is also Gaussian distributed and given by

$$p(\boldsymbol{x} + \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y) . \tag{6.116}$$

Knowing that $p(\boldsymbol{x}+\boldsymbol{y})$ is Gaussian, the mean and covariance matrix can be determined immediately using the results from (6.46)–(6.49). This property will be important when we consider i.i.d. Gaussian noise acting on random variables as is the case for linear regression (Chapter 9).

Example 6.9
Since expectations are linear operations, we can obtain the weighted sum of independent Gaussian random variables

$$p(a\boldsymbol{x} + b\boldsymbol{y}) = \mathcal{N}(a\boldsymbol{\mu}_x + b\boldsymbol{\mu}_y, a\boldsymbol{\Sigma}_x + b\boldsymbol{\Sigma}_y) . \tag{6.117}$$

Remark. A case which will be useful in Chapter 11 is the weighted sum of Gaussian densities. This is different from the weighted sum of Gaussian random variables. ♢

In Theorem 6.13, the random variable z is from the mixture density of the two random variables x and y. The theorem can be generalized to the multivariate random variable case, since linearity of expectations holds also for multivariate random variables. However the idea of a squared random variable requires more care.

Theorem 6.13. *Consider a weighted sum of two univariate Gaussian densities*

$$p(z) = \alpha p(x) + (1 - \alpha)p(y) \tag{6.118}$$

where the scalar $0 < \alpha < 1$ is the mixture weight, and $p(x)$ and $p(y)$ are univariate Gaussian densities (Equation (6.100)) with different parameters, that is $(\mu_x, \sigma_x^2) \neq (\mu_y, \sigma_y^2)$.

The mean of the mixture z is given by the weighted sum of the means of each random variable,

$$\mathbb{E}[z] = \alpha\mu_x + (1 - \alpha)\mu_y. \tag{6.119}$$

The variance of the mixture z is the mean of the conditional variance and the variance of the conditional mean,

$$\mathbb{V}[z] = \left[\alpha\sigma_x^2 + (1 - \alpha)\sigma_y^2\right] + \left(\left[\alpha\mu_x^2 + (1 - \alpha)\mu_y^2\right] \left[\alpha\mu_x + (1 - \alpha)\mu_y\right]^2\right) . \tag{6.120}$$

Proof The mean of the mixture z is given by the weighted sum of the

means of each random variable. We apply the definition of the mean (Definition 6.3), and plug in our mixture (Equation (6.118)) above

$$\mathbb{E}[z] = \int_{-\infty}^{\infty} zp(z)\mathrm{d}z \tag{6.121}$$

$$= \int_{-\infty}^{\infty} \alpha zp(x) + (1-\alpha)zp(y)\mathrm{d}z \tag{6.122}$$

$$= \alpha\int_{-\infty}^{\infty} zp(x)\mathrm{d}z + (1-\alpha)\int_{-\infty}^{\infty} zp(y)\mathrm{d}z \tag{6.123}$$

$$= \alpha\mu_x + (1-\alpha)\mu_y. \tag{6.124}$$

To compute the variance, we can use the raw score version of the variance (Equation (6.40)), which requires an expression of the expectation of the squared random variable. Here we use the definition of an expectation of a function (the square) of a random variable (Definition 6.4).

$$\mathbb{E}[z^2] = \int_{-\infty}^{\infty} z^2p(z)\mathrm{d}z \tag{6.125}$$

$$= \int_{-\infty}^{\infty} \alpha z^2p(x) + (1-\alpha)z^2p(y)\mathrm{d}z \tag{6.126}$$

$$= \alpha\int_{-\infty}^{\infty} z^2p(z)\mathrm{d}z + (1-\alpha)\int_{-\infty}^{\infty} z^2p(y)\mathrm{d}z \tag{6.127}$$

$$= \alpha(\mu_x^2+\sigma_x^2) + (1-\alpha)(\mu_y^2+\sigma_y^2). \tag{6.128}$$

where in the last equality, we again used the raw score version of the variance and rearranged terms such that the expectation of a squared random variable is the sum of the squared mean and the variance.

Therefore the variance is given by subtracting the two terms above

$$\mathbb{V}[z] = \mathbb{E}[z^2] - (\mathbb{E}[z])^2 \tag{6.129}$$

$$= \alpha(\mu_x^2+\sigma_x^2) + (1-\alpha)(\mu_y^2+\sigma_y^2) - (\alpha\mu_x + (1-\alpha)\mu_y)^2 \tag{6.130}$$

$$= \left[\alpha\sigma_x^2 + (1-\alpha)\sigma_y^2\right] + \left(\left[\alpha\mu_x^2 + (1-\alpha)\mu_y^2\right]\left[\alpha\mu_x + (1-\alpha)\mu_y\right]^2\right). \tag{6.131}$$

Observe for a mixture, the individual components can be considered to be conditional distributions (conditioned on the component identity). The last line is an illustration of the conditional variance formula: "The variance of a mixture is the mean of the conditional variance and the variance of the conditional mean". □

Remark. The derivation above holds for any density, but in the case of the Gaussian since it is fully determined by the mean and variance, the mixture density can be determined in closed form. ♢

Recall the example in Section 6.5, where we considered a bivariate standard Gaussian random variable X and performed a linear transformation $\boldsymbol{A}X$ on it. The outcome was a Gaussian random variable with zero mean

and covariance $\boldsymbol{A}^\top\boldsymbol{A}$. Observe that adding a constant vector will change the mean of the distribution, without affecting its variance, that is the random variable $\boldsymbol{x} + \boldsymbol{\mu}$ is Gaussian with mean $\boldsymbol{\mu}$ and identity covariance. Therefore, a linear (or affine) transformation of a Gaussian random variable is Gaussian distributed.

Consider a Gaussian distributed random variable $\boldsymbol{x} \sim \mathcal{N}\big(\boldsymbol{x} \,|\, \boldsymbol{\mu},\, \boldsymbol{\Sigma}\big)$. For a given matrix $\boldsymbol{A}$ of appropriate shape, let $\boldsymbol{y}$ be a random variable $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ which is a transformed version of $\boldsymbol{x}$. We can compute the mean of $\boldsymbol{y}$ by using the fact that the expectation is a linear operator (Equation (6.50)) as follows:

$$\mathbb{E}[\boldsymbol{A}\boldsymbol{x}] = \boldsymbol{A}\mathbb{E}[\boldsymbol{x}] = \boldsymbol{A}\boldsymbol{\mu}. \tag{6.132}$$

Similarly the variance of $\boldsymbol{y}$ can be found by using Equation (6.51):

$$\mathbb{V}[\boldsymbol{A}\boldsymbol{x}] = \boldsymbol{A}\mathbb{V}[\boldsymbol{x}]\boldsymbol{A}^\top = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top. \tag{6.133}$$

This means that the random variable $\boldsymbol{y}$ is distributed according to

$$p(\boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{x} \,|\, \boldsymbol{A}\boldsymbol{\mu},\, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top\big). \tag{6.134}$$

Let us now consider the reverse transformation: when we know that a random variable has a mean that is a linear transformation of another random variable. For a given matrix $\boldsymbol{A}$ of appropriate shape, let $\boldsymbol{y}$ be a Gaussian random variable with mean $\boldsymbol{A}\boldsymbol{x}$, i.e.,

$$p(\boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{y} \,|\, \boldsymbol{A}\boldsymbol{x},\, \boldsymbol{\Sigma}\big). \tag{6.135}$$

What is the corresponding probability distribution $p(\boldsymbol{x})$? If $\boldsymbol{A}$ is invertible, then we can write $\boldsymbol{x} = \boldsymbol{A}^{-1}y$ and apply the transformation in the previous paragraph. However in general $\boldsymbol{A}$ is not invertible, and we use an approach similar to the that of the pseudo-inverse (Equation 3.54). That is we pre-multiply both sides with $\boldsymbol{A}^\top$ and then invert $\boldsymbol{A}^\top\boldsymbol{A}$ which is symmetric and positive definite, giving us the relation

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \iff (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{y} = \boldsymbol{x}\,. \tag{6.136}$$

Hence, $\boldsymbol{x}$ is a linear transformation of $\boldsymbol{y}$, and we obtain

$$p(\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{x} \,|\, (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{y},\, (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{\Sigma}\boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\big)\,. \tag{6.137}$$

6.6.4 Sampling from Multivariate Gaussian Distributions

We will not explain the subtleties of random sampling on a computer. In the case of a multivariate Gaussian, this process consists of three stages: first we need a source of pseudo-random numbers that provide a uniform sample in the interval [0,1], second we use a non-linear transformation such as the Box-Müller transform (Devroye, 1986) to obtain a sample from a univariate Gaussian, and third we collate a vector of these samples to obtain a sample from a multivariate standard normal $\mathcal{N}\big(\boldsymbol{0},\, \boldsymbol{I}\big)$.

For a general multivariate Gaussian, that is where the mean is non-zero and the covariance is not the identity matrix, we use the properties of linear transformations of a Gaussian random variable. Assume we are interested in generating samples $\boldsymbol{x}_i, i = 1, \ldots, n$, from a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. We would like to construct the sample from a sampler that provides samples from the multivariate standard normal $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

To compute the Cholesky factorization of a matrix, it is required that the matrix is symmetric and positive definite (Section 3.2.3). Covariance matrices possess this property.

To obtain samples from a multivariate normal $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we can use the properties of a linear transformation of a Gaussian random variable: If $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ then $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\mu}$, where $\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{\Sigma}$, is Gaussian distributed with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Recall from Section 4.3 that $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^\top$ is the Cholesky factorization of $\boldsymbol{\Sigma}$.

6.7 Conjugacy and the Exponential Family

Many of the probability distributions "with names" that we find in statistics textbooks were discovered to model particular types of phenomena. The distributions are also related to each other in complex ways (Leemis and McQueston, 2008). For a beginner in the field, it can be overwhelming to figure out which distribution to use. In addition, many of these distributions were discovered at a time that statistics and computation was done by pencil and paper. It is natural to ask what are meaningful concepts in the computing age (Efron and Hastie, 2016). In the previous section, we saw that many of the operations required for inference can be conveniently calculated when the distribution is Gaussian. It is worth recalling at this point the desiderata for manipulating probability distributions.

"Computers" were a job description.

1. There is some "closure property" when applying the rules of probability, e.g., Bayes' theorem.
2. As we collect more data, we do not need more parameters to describe the distribution.
3. Since we are interested in learning from data, we want parameter estimation to behave nicely.

It turns out that the class of distributions called the *exponential family* provides the right balance of generality while retaining favourable computation and inference properties. Before we introduce the exponential family, let us see three more members of "named" probability distributions.

exponential family

Example 6.10

The *Bernoulli distribution* is a distribution for a single binary variable $x \in \{0, 1\}$ and is governed by a single continuous parameter $\mu \in [0, 1]$ that represents the probability of $x = 1$. The Bernoulli distribution is defined

Bernoulli distribution

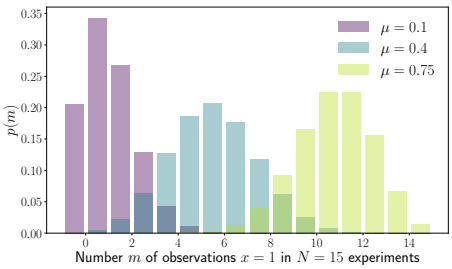


Figure 6.8 Examples of the Binomial distribution for $\mu \in \{0.1, 0.4, 0.75\}$ and $N = 15$.

as

$$p(x \mid \mu) = \mu^x (1-\mu)^{1-x}, \quad x \in \{0, 1\}, \tag{6.138}$$

$$\mathbb{E}[x] = \mu, \tag{6.139}$$

$$\mathbb{V}[x] = \mu(1-\mu), \tag{6.140}$$

where $\mathbb{E}[x]$ and $\mathbb{V}[x]$ are the mean and variance of the binary random variable x.

An example where the Bernoulli distribution can be used is when we are interested in modeling the probability of "head" when flipping a coin.

Example 6.11

Binomial distribution

The *Binomial distribution* is a generalization of the Bernoulli distribution to a distribution over integers. In particular, the Binomial can be used to describe the probability of observing m occurrences of $x = 1$ in a set of N samples from a Bernoulli distribution where $p(x = 1) = \mu \in [0, 1]$. The Binomial distribution is defined as

$$p(m \mid N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m}, \tag{6.141}$$

$$\mathbb{E}[m] = N\mu, \tag{6.142}$$

$$\mathbb{V}[m] = N\mu(1-\mu) \tag{6.143}$$

where $\mathbb{E}[m]$ and $\mathbb{V}[m]$ are the mean and variance of m, respectively.

An example where the Binomial could be used is if we want to describe the probability of observing m "heads" in N coin-flip experiments if the probability for observing head in a single experiment is μ.

Example 6.12

The Beta distribution is a distribution over a continuous variable $\mu \in [0, 1]$, which is often used to represent the probability for some binary event

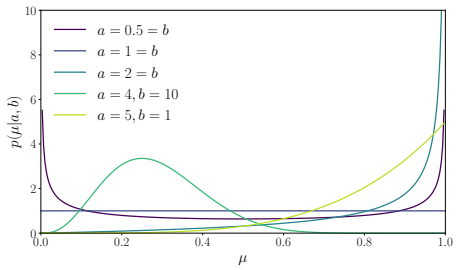


Figure 6.9 Examples of the Beta distribution for different values of α and β.

(e.g., the parameter governing the Bernoulli distribution). The Beta distribution (illustrated in Figure 6.9) itself is governed by two parameters $\alpha > 0,\ \beta > 0$ and is defined as

$$p(\mu \,|\, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{6.144}$$

$$\mathbb{E}[\mu] = \frac{\alpha}{\alpha + \beta}\,, \qquad \mathbb{V}[\mu] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{6.145}$$

where $\Gamma(\cdot)$ is the Gamma function defined as

$$\Gamma(t) := \int_0^\infty x^{t-1} \exp(-x) dx, \qquad t > 0\,. \tag{6.146}$$

$$\Gamma(t+1) = t\Gamma(t)\,. \tag{6.147}$$

Note that the fraction of Gamma functions in (6.144) normalizes the Beta distribution.

Intuitively, α moves probability mass toward 1, whereas β moves probability mass toward 0. There are some special cases (Murphy, 2012):

- For $\alpha = 1 = \beta$ we obtain the uniform distribution $\mathcal{U}[0, 1]$.
- For $\alpha, \beta < 1$, we get a bimodal distribution with spikes at 0 and 1.
- For $\alpha, \beta > 1$, the distribution is unimodal.
- For $\alpha, \beta > 1$ and $\alpha = \beta$, the distribution is unimodal, symmetric and centered in the interval $[0, 1]$, i.e., the mode/mean is at $\frac{1}{2}$.

Remark. There is a whole zoo of distributions with names, and they are related in different ways to each other (Leemis and McQueston, 2008). It is worth keeping in mind that each named distribution is created for a particular reason, but may have other applications. Knowing the reason behind the creation of a particular distribution often allows insight into how to best use it. We introduced the above three distributions to be able to illustrate the concepts of conjugacy (Section 6.7.1) and exponential families (Section 6.153). ♢

6.7.1 Conjugacy

According to Bayes' theorem (6.21), the posterior is proportional to the product of the prior and the likelihood. The specification of the prior can be tricky for two reasons: First, the prior should encapsulate our knowledge about the problem before we see some data. This is often difficult to describe. Second, it is often not possible to compute the posterior distribution analytically. However, there are some priors that are computationally convenient: *conjugate priors*.

conjugate priors

conjugate

Definition 6.14 (Conjugate Prior)**.** A prior is *conjugate* for the likelihood function if the posterior is of the same form/type as the prior.

Conjugacy is particularly convenient because we can algebraically calculate our posterior distribution by updating the parameters of the prior distribution.

Remark. When considering the geometry of probability distributions, conjugate priors retain the same distance structure as the likelihood (Agarwal and III, 2010). ♢

To introduce a concrete example of conjugate priors, we describe below the Binomial distribution (defined on discrete random variables) and the Beta distribution (defined on continuous random variables).

Example 6.13 (Beta-Binomial Conjugacy)
Consider a Binomial random variable $x \sim \text{Bin}(m \,|\, N, \mu)$ where

$$p(x \,|\, \mu, N) = \binom{N}{m} \mu^m (1-\mu)^{N-m} \;\propto\; \mu^a (1-\mu)^b \tag{6.148}$$

for some constants a, b. We place a Beta prior on the parameter μ:

$$\text{Beta}(\mu \,|\, \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1}(1-\mu)^{\beta-1} \;\propto\; \mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{6.149}$$

If we now observe some outcomes $\boldsymbol{x} = (x_1, \ldots, x_N)$ of a repeated coin-flip experiment with h heads and t tails, we compute the posterior distribution on μ as

$$p(\mu \,|\, \boldsymbol{x} = h) \;\propto\; p(\boldsymbol{x} \,|\, \mu) p(\mu \,|\, \alpha, \beta) = \mu^h (1-\mu)^t \mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{6.150}$$

$$= \mu^{h+\alpha-1}(1-\mu)^{t+\beta-1} \;\propto\; \text{Beta}(h+\alpha, t+\beta) \tag{6.151}$$

i.e., the posterior distribution is a Beta distribution as the prior, i.e., the Beta prior is conjugate for the parameter μ in the Binomial likelihood function.

Table 6.2 lists examples for conjugate priors for the parameters of some of standard likelihoods used in probabilistic modeling. Distributions such

Likelihood	Conjugate prior	Posterior
Bernoulli	Beta	Beta
Binomial	Beta	Beta
Gaussian	Gaussian/inverse Gamma	Gaussian/inverse Gamma
Gaussian	Gaussian/inverse Wishart	Gaussian/inverse Wishart
Multinomial	Dirichlet	Dirichlet

Table 6.2 Examples of conjugate priors for common likelihood functions.

as Multinomial, inverse Gamma, inverse Wishart, and Dirichlet can be found in any statistical text, and is for example described in Bishop (2006).

The Beta distribution is the conjugate prior for the parameter μ in both the Binomial and the Bernoulli likelihood. For a Gaussian likelihood function, we can place a conjugate Gaussian prior on the mean. The reason why the Gaussian likelihood appears twice in the table is that we need distinguish the univariate from the multivariate case. In the univariate (scalar) case, the inverse Gamma is the conjugate prior for the variance. In the multivariate case, we use a conjugate inverse Wishart distribution as a prior on the covariance matrix. The Dirichlet distribution is the conjugate prior for the multinomial likelihood function. For further details, we refer to Bishop (2006).

Alternatively, the Gamma prior is conjugate for the precision (inverse variance) in the Gaussian likelihood.

Alternatively, the Wishart prior is conjugate for the precision matrix (inverse covariance matrix) in the Gaussian likelihood.

6.7.2 Sufficient Statistics

Recall that a statistic of a random variable is a deterministic function of that random variable. For example if $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top$ is a vector of univariate Gaussian random variables, that is $x_n \sim \mathcal{N}(\mu, \sigma^2)$, then the sample mean $\hat{\mu} = \frac{1}{N}(x_1 + \cdots + x_N)$ is a statistic. Sir Ronald Fisher discovered the notion of *sufficient statistics*: the idea that there are statistics that will contain all available information that can be inferred from data corresponding to the distribution under consideration. In other words sufficient statistics carry all the information needed to make inference about the population, that is they are the statistics that are sufficient to represent the distribution.

sufficient statistics

For a set of distributions parameterized by θ, let x be a random variable with distribution given an unknown θ_0. A vector $\phi(x)$ of statistics are called sufficient statistics for θ_0 if they contain all possible information about θ_0. To be more formal about "contain all possible information": this means that the probability of x given θ can be factored into a part that does not depend on θ, and a part that depends on θ only via $\phi(x)$. The Fisher-Neyman factorization theorem formalizes this notion, which we state below without proof.

Theorem 6.15 (Fisher-Neyman)**.** *Let x have probability density function $p(x \mid \theta)$. Then the statistics $\phi(x)$ are sufficient for θ if and only if $p(x \mid \theta)$ can be written in the form*

$$p(x \mid \theta) = h(x)g_\theta(\phi(x)). \tag{6.152}$$

where $h(x)$ is a distribution independent of θ and g_θ captures all the dependence on θ via sufficient statistics $\phi(x)$.

Note that if $p(x \mid \theta)$ does not depend on θ then $\phi(x)$ is trivially a sufficient statistic for any function ϕ. The more interesting case is that $p(x \mid \theta)$ is dependent only on $\phi(x)$ and not x itself. In this case, $\phi(x)$ is a sufficient statistic for x.

A natural question to ask is as we observe more data, do we need more parameters θ to describe the distribution? It turns out that the answer is yes in general, and this is studied in non-parametric statistics (Wasserman, 2007). A converse question is to consider which class of distributions have finite dimensional sufficient statistics, that is the number of parameters needed to describe them do not increase arbitrarily. The answer is exponential family distributions, described in the following section.

6.7.3 Exponential Family

At this point it is worth being a bit careful by discussing three possible levels of abstraction we can have when considering distributions (of discrete or continuous random variables). At the most concrete end of the spectrum, we have a particular named distribution with fixed parameters, for example a univariate Gaussian $\mathcal{N}(0, 1)$ with zero mean and unit variance. In machine learning, we often fix the parametric form (the univariate Gaussian) and infer the parameters from data. For example, we assume a univariate Gaussian $\mathcal{N}(\mu, \sigma^2)$ with unknown mean μ and unknown variance σ^2, and use a maximum likelihood fit to determine the best parameters (μ, σ^2). We will see an example of this when considering linear regression in Chapter 9. A third level of abstraction is to consider families of distributions, and in this book, we consider the exponential family. The univariate Gaussian is an example of a member of the exponential family. Many of the widely used statistical models, including all the "named" models in Table 6.2, are members of the exponential family. They can all be unified into one concept (Brown, 1986).

Remark. A brief historical anecdote: like many concepts in mathematics and science, exponential families were independently discovered at the same time by different researchers. In the years 1935–1936, Edwin Pitman in Tasmania, Georges Darmois in Paris, and Bernard Koopman in New York, independently showed that the exponential families are the only families that enjoy finite-dimensional sufficient statistics under repeated independent sampling (Lehmann and Casella, 1998). ♢

exponential family

An *exponential family* is a family of probability distributions, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^D$, of the form

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = h(\boldsymbol{x}) \exp\left(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta})\right) , \tag{6.153}$$

where $\boldsymbol{\phi}(\boldsymbol{x})$ is the vector of sufficient statistics. In general, any inner prod-

uct (Section 3.2) can be used in (6.153), and for concreteness we will use the standard dot product here. Note that the form of the exponential family is essentially a particular expression of $g_\theta(\phi(x))$ in the Fisher-Neyman theorem (Theorem 6.15).

The factor $h(\boldsymbol{x})$ can be absorbed into the dot product term by adding another entry to the vector of sufficient statistics $\log h(\boldsymbol{x})$, and constraining the corresponding parameter $\theta = 1$. The term $A(\boldsymbol{\theta})$ is the normalization constant that ensures that the distribution sums up or integrates to one and is called the *log partition function*. A good intuitive notion of exponential families can be obtained by ignoring these two terms and considering exponential families as distributions of the form

log partition function

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) \propto \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x})\right). \tag{6.154}$$

For this form of parameterization, the parameters $\boldsymbol{\theta}$ are called the *natural paramters*. At first glance it seems that exponential families is a mundane transformation by adding the exponential function to the result of a dot product. However, there are many implications that allow for convenient modelling and efficient computation to the fact that we can capture information about data in $\boldsymbol{\phi}(\boldsymbol{x})$.

natural paramters

Example 6.14 (Gaussian as Exponential Family)

Consider the univariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Let $\boldsymbol{\phi}(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$. Then by using the definition of the exponential family,

$$p(x \mid \boldsymbol{\theta}) \propto \exp(\theta_1 x + \theta_2 x^2). \tag{6.155}$$

Setting

$$\theta = \left[\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}\right]^\top \tag{6.156}$$

and substituting into (6.155) we obtain

$$p(x \mid \boldsymbol{\theta}) \propto \exp\left(\frac{\mu x}{\sigma^2} - \frac{x^2}{2\sigma^2}\right) \propto \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right). \tag{6.157}$$

Therefore, the univariate Gaussian distribution is a member of the exponential family with sufficient statistic $\boldsymbol{\phi}(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$.

Exponential families also provide a convenient way to find conjugate pairs of distributions. In the following example, we will derive a result that is similar to the Beta-Binomial conjugacy result of Section 6.7.1. Here we will show that the Beta distribution is a conjugate prior for the Bernoulli distribution.

Example 6.15 (Beta-Bernoulli Conjugacy)
Let $x \in \{0, 1\}$ be distributed according to the Bernoulli distribution with parameter $\theta \in [0, 1]$, that is $P(x = 1 \,|\, \theta) = \theta$. This can also be expressed as $P(x \,|\, \theta) = \theta^x(1 - \theta)^{1-x}$. Let θ be distributed according to a Beta distribution with parameters α, β, that is $p(\theta \,|\, \alpha, \beta) \propto \theta^{\alpha-1}(1 - \theta)^{\beta-1}$.

Multiplying the Beta and the Bernoulli distributions, we get

$$p(\theta \,|\, x, \alpha, \beta) = P(x \,|\, \theta) \times p(\theta \,|\, \alpha, \beta) \tag{6.158}$$
$$\propto \theta^x(1 - \theta)^{1-x} \times \theta^{\alpha-1}(1 - \theta)^{\beta-1} \tag{6.159}$$
$$= \theta^{\alpha+x-1}(1 - \theta)^{\beta+(1-x)-1} \tag{6.160}$$
$$\propto p(\theta \,|\, \alpha + x, \beta + (1 - x)). \tag{6.161}$$

The last line above is the Beta distribution with parameters $(\alpha + x, \beta + (1 - x))$.

Remark. The rewriting above of the Bernoulli distribution, where we use Boolean variables as numerical 0 or 1 and express them in the exponents, is a trick that is often used in machine learning textbooks. Another occurence of this is when expressing the Multinomial distribution. ♢

As mentioned in the previous section, the main motivation for exponential families is that they have finite-dimensional sufficient statistics. Additionally, conjugate distributions are easy to write down, and the conjugate distributions also come from an exponential family. From an inference perspective, maximum likelihood estimation behaves nicely because empirical estimates of sufficient statistics are optimal estimates of the population values of sufficient statistics (recall the mean and covariance of a Gaussian). From an optimization perspective, the log-likelihood function is concave allowing for efficient optimization approaches to be applied (Chapter 7).

6.8 Further Reading

Probabilistic models in machine learning Bishop (2006); Murphy (2012) provide a way for users to capture uncertainty about data and predictive models in a principled fashion. Ghahramani (2015) presents a short review of probabilistic models in machine learning. This chapter is rather terse at times, and Grinstead and Snell (1997) provides a more relaxed presentation that is suitable for self study. Readers interested in more philosophical aspects of probability should consider Hacking (2001), whereas a more software engineering approach is presented by Downey (2014).

Given a probabilistic model, we may be lucky enough to be able to compute parameters of interest analytically. However in general analytic solutions are rare and computational methods such as sampling (Brooks et al.,

2011) and variational inference (Blei et al., 2017) are used. Ironically the recent surge in interest in neural networks has resulted in a broader appreciation of probabilisitic models. For example the idea of normalizing flows (Rezende and Mohamed, 2015) relies on change of variables for transforming random variables. An overview of methods for variational inference as applied to neural networks is described in Chapters 16 to 20 of Goodfellow et al. (2016).

A more technical audience interested in the details of probability theory have many options (Jacod and Protter, 2004; Jaynes, 2003; Mackay, 2003) including some very technical discussions (Dudley, 2002; Shiryayev, 1984; Lehmann and Casella, 1998; Bickel and Doksum, 2006). We side stepped a large part of the difficulty by glossing over measure theoretic questions (Billingsley, 1995; Pollard, 2002), and by assuming without construction that we have real numbers, and ways of defining sets on real numbers as well as their appropriate frequency of occurrence. As machine learning allows us to model move intricate distributions on ever move complex types of data, a developer of probabilistic machine learning models would have to understand these more technical aspects. Machine learning books with a probabilistic modelling focus includes Mackay (2003); Bishop (2006); Murphy (2012); Barber (2012); Rasmussen and Williams (2006).

Exercises

6.1 You have written a computer program that sometimes compiles and sometimes not (code does not change). You decide to model the apparent stochasticity (success vs no success) x of the compiler using a Bernoulli distribution with parameter μ:

$$p(x|\mu) = \mu^x(1-\mu)^{1-x}\,, \quad x \in \{0, 1\}$$

Choose a conjugate prior for the Bernoulli likelihood and compute the posterior distribution $p(\mu|x_1, \ldots, x_N)$.

6.2 Consider the following time-series model:

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{w}\,, \quad \boldsymbol{w} \sim \mathcal{N}\big(\boldsymbol{0},\, \boldsymbol{Q}\big) \\ \boldsymbol{y}_t &= \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}\,, \quad \boldsymbol{v} \sim \mathcal{N}\big(\boldsymbol{0},\, \boldsymbol{R}\big) \end{aligned}$$

where $\boldsymbol{w}, \boldsymbol{v}$ are i.i.d. Gaussian noise variables. Further, assume that $p(\boldsymbol{x}_0) = \mathcal{N}\big(\boldsymbol{\mu}_0,\, \boldsymbol{\Sigma}_0\big)$.

1. What is the form of $p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$? Justify your answer (you do not have to explicitly compute the joint distribution). (1–2 sentences)
2. Assume that $p(\boldsymbol{x}_t|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t) = \mathcal{N}\big(\boldsymbol{\mu}_t,\, \boldsymbol{\Sigma}_t\big)$.
    1. Compute $p(\boldsymbol{x}_{t+1}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$
    2. Compute $p(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$
    3. At time $t{+}1$, we observe the value $\boldsymbol{y}_{t+1} = \hat{\boldsymbol{y}}$. Compute $p(\boldsymbol{x}_{t+1}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t+1})$.

6.3 Prove the relationship in Equation 6.40, which relates the standard definition of the variance to the raw score expression for the variance.

6.4 Prove the relationship in Equation 6.41, which relates the pairwise difference between examples in a dataset with the raw score expression for the variance.

6.5 Express the Bernoulli distribution in the natural parameter form of the exponential family (Equation (6.153)).

6.6 Express the Binomial distribution as an exponential family distribution. Also express the Beta distribution is an exponential family distribution. Show that the product of the Beta and the Binomial distribution is also a member of the exponential family.

6.7 **Iterated Expectations.**
Consider two random variables x, y with joint distribution $p(x, y)$. Show that:

$$\mathbb{E}_x[x] = \mathbb{E}_y\big[\mathbb{E}_x[x|y]\big]$$

Here, $\mathbb{E}_x[x|y]$ denotes the expected value of x under the conditional distribution $p(x|y)$.

6.8 **Manipulation of Gaussian Random Variables.**
Consider a Gaussian random variable $\boldsymbol{x} \sim \mathcal{N}\big(\boldsymbol{x} \,|\, \boldsymbol{\mu}_x,\, \boldsymbol{\Sigma}_x\big)$, where $\boldsymbol{x} \in \mathbb{R}^D$. Furthermore, we have

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{w}\,, \tag{6.162}$$

where $\boldsymbol{y} \in \mathbb{R}^E$, $\boldsymbol{A} \in \mathbb{R}^{E\times D}$, $\boldsymbol{b} \in \mathbb{R}^E$, and $\boldsymbol{w} \sim \mathcal{N}\big(\boldsymbol{w} \,|\, \boldsymbol{0},\, \boldsymbol{Q}\big)$ is independent Gaussian noise. "Independent" implies that $\boldsymbol{x}$ and $\boldsymbol{w}$ are independent random variables and that $\boldsymbol{Q}$ is diagonal.

1. Write down the likelihood $p(\boldsymbol{y}|\boldsymbol{x})$.
2. The distribution $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$ is Gaussian.[3] Compute the mean $\boldsymbol{\mu}_y$ and the covariance $\boldsymbol{\Sigma}_y$. Derive your result in detail.
3. The random variable $\boldsymbol{y}$ is being transformed according to the measurement mapping

$$\boldsymbol{z} = \boldsymbol{C}\boldsymbol{y} + \boldsymbol{v}\,, \tag{6.163}$$

where $\boldsymbol{z} \in \mathbb{R}^F$, $\boldsymbol{C} \in \mathbb{R}^{F\times E}$, and $\boldsymbol{v} \sim \mathcal{N}\big(\boldsymbol{v} \,|\, \boldsymbol{0},\, \boldsymbol{R}\big)$ is independent Gaussian (measurement) noise.

   - Write down $p(\boldsymbol{z}|\boldsymbol{y})$.
   - Compute $p(\boldsymbol{z})$, i.e., the mean $\boldsymbol{\mu}_z$ and the covariance $\boldsymbol{\Sigma}_z$. Derive your result in detail.

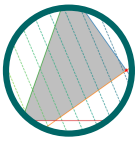
4. Now, a value $\hat{\boldsymbol{y}}$ is measured. Compute the posterior distribution $p(\boldsymbol{x}|\hat{\boldsymbol{y}})$.[4]
   Hint for solution: Start by explicitly computing the joint Gaussian $p(\boldsymbol{x}, \boldsymbol{y})$. This also requires to compute the cross-covariances $\mathrm{Cov}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}, \boldsymbol{y}]$ and $\mathrm{Cov}_{\boldsymbol{y},\boldsymbol{x}}[\boldsymbol{y}, \boldsymbol{x}]$. Then, apply the rules for Gaussian conditioning.

[3] An affine transformation of the Gaussian random variable $\boldsymbol{x}$ into $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ preserves Gaussianity. Furthermore, the sum of this Gaussian random variable and the independent Gaussian random variable $\boldsymbol{w}$ is Gaussian.

[4] This posterior is also Gaussian, i.e., we need to determine only its mean and covariance matrix.

7

Continuous Optimization



Since machine learning algorithms are implemented on a computer, the mathematical formulations are expressed as numerical optimization methods. This chapter describes the basic numerical methods for training machine learning models. Training a machine learning model often boils down to finding a good set of parameters. The notion of "good" is determined by the objective function or the probabilistic model, which we will see examples of in the second part of this book. Given an objective function finding the best value is done using optimization algorithms.

Since we consider data and models in $\mathbb{R}^D$ the optimization problems we face are *continuous* optimization problems, as opposed to *combinatorial* optimization problems for discrete variables.

This chapter covers two main branches of continuous optimization (Figure 7.1): unconstrained and constrained optimization. We will assume in this chapter that our objective function is differentiable (see Chapter 5), hence we have access to a gradient at each location in the space to help us find the optimum value. By convention most objective functions in machine learning are intended to be minimized, that is the best value is the minimum value. Intuitively finding the best value is like finding the valleys of the objective function, and the gradients point us uphill. The idea is to move downhill (opposite to the gradient) and hope to find the deepest point. For unconstrained optimization, this is the only concept we need, but there are several design choices which we discuss in Section 7.1. For constrained optimization, we need to introduce other concepts to manage the constraints (Section 7.2). We will also introduce a special class of problems (convex optimization problems in Section 7.3) where we can make statements about reaching the global optimum.

global minimum

Consider the function in Figure 7.2. The function has a *global minimum* around the value $x = -4.5$ which has the objective function value of around -47. Since the function is "smooth" the gradients can be used to help find the minimum by indicating whether we should take a step to the right or left. This assumes that we are in the correct bowl, as there exists another *local minimum* around the value $x = 0.7$. Recall that we can solve for all the stationary points of a function by calculating its derivative and setting it to zero. Let

local minimum

Stationary points are points that have zero gradient.

$$\ell(x) = x^4 + 7x^3 + 5x^2 - 17x + 3. \tag{7.1}$$

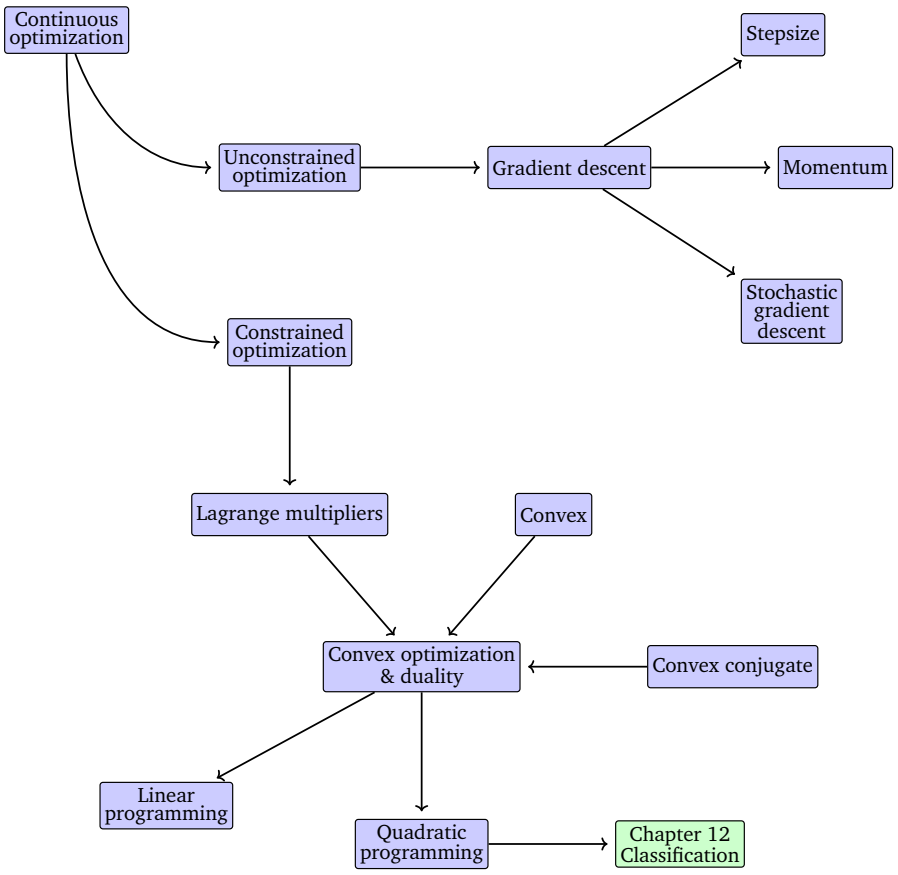


Figure 7.1 A mind map of the concepts related to optimization, as presented in this chapter. There are two main ideas: gradient descent and convex optimization.

Its gradient is given by

$$\frac{\mathrm{d}\ell(x)}{\mathrm{d}x} = 4x^3 + 21x^2 + 10x - 17. \tag{7.2}$$

Since this is a cubic equation, it has three solutions when set to zero. Two of them are minima and one is a maximum (around $x = -1.4$). Recall that to check whether a stationary point is a minimum or maximum we need to take the derivative a second time and check whether the second derivative is positive or negative at the stationary point.

$$\frac{\mathrm{d}^2\ell(x)}{\mathrm{d}x^2} = 12x^2 + 42x + 10 \tag{7.3}$$

By substituting our visually estimated values of $x = -4.5, -1.4, 0.7$ we will observe that as expected the middle point is a maximum $\left(\frac{\mathrm{d}^2\ell(x)}{\mathrm{d}x^2} < 0\right)$ and the other two stationary points are minimums.

Note that we have avoided analytically solving for values of x in the previous discussion, although for low order polynomials such as the above we could. In general, we are unable to find analytic solutions, and hence we

In fact according to the Abel-Ruffini theorem, also known as Abel's impossibility theorem, there is in general no algebraic solution for polynomials of degree 5 or more.

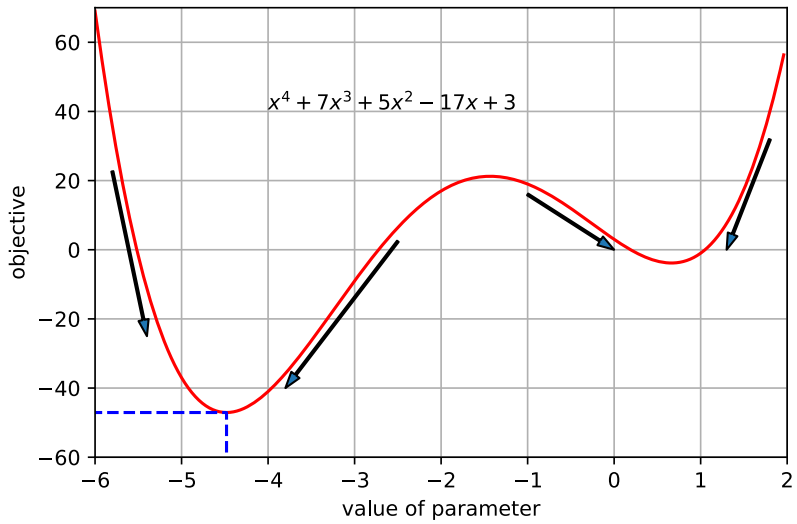


Figure 7.2 Example objective function. Gradients are indicated by arrows, and the global minimum is indicated by the dashed blue line.

need to start at some value, say $x_0 = -10$ and follow the gradient. The gradient indicates that we should go right, but not how far (this is called the step size). Furthermore, if we had started at the right side (e.g. $x_0 = 0$) the gradient would have led us to the wrong minimum. Figure 7.2 illustrates the fact that for $x > -1$, the gradient points towards the minimum on the right of the figure, which has a larger objective value.

We will see in Section 7.3 a class of functions called convex functions that do not exhibit this tricky dependency on the starting point of the optimization algorithm. For *convex functions* all local minima are global minimum. It turns out that many machine learning objective functions are designed such that they are convex, and we will see an example in Chapter 12.

convex functions

The discussion in this chapter so far was about a one dimensional function, where we are able to visualize the ideas of gradients, descent directions and optimal values. In the rest of this chapter we develop the same ideas in high dimensions. Unfortunately we can only visualize the concepts in one dimension, but some concepts do not generalize directly to higher dimensions, therefore some care needs to be taken when reading.

7.1 Optimization using Gradient Descent

We now consider the problem of solving for the minimum of a real-valued function

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{7.4}$$

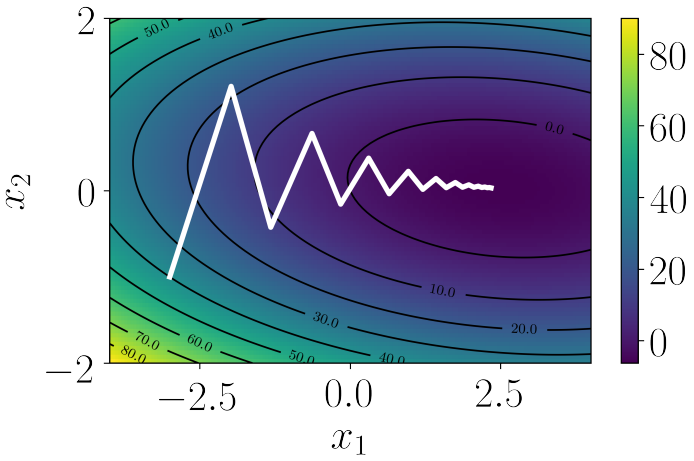


Figure 7.3 Gradient descent on a 2 dimensional quadratic surface (shown as a heatmap). See Example 7.1 for a description.

where $f : \mathbb{R}^d \to \mathbb{R}$ is an objective function that captures the machine learning problem at hand. We assume that our function f is differentiable, and we are unable to analytically find a solution in closed form.

Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point. Recall from Chapter 5 that the gradient points in the direction of the steepest ascent and it is orthogonal to the contour lines of the function we wish to optimize.

We use the convention of row vectors for gradients.

Let us consider multivariate functions. Imagine a surface (described by the function $f(\boldsymbol{x})$) with a ball starting at a particular location $\boldsymbol{x}_0$. When the ball is released, it will move downhill in the direction of steepest descent. Gradient descent exploits the fact that $f(\boldsymbol{x}_0)$ decreases fastest if one moves from $\boldsymbol{x}_0$ in the direction of the negative gradient $-((\nabla f)(\boldsymbol{x}_0))^\top$ of f at $\boldsymbol{x}_0$. We assume in this book that the functions are differentiable, and refer the reader to more general settings in Section 7.4. Then, if

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \gamma((\nabla f)(\boldsymbol{x}_0))^\top \tag{7.5}$$

for a small *step size* $\gamma \geqslant 0$ then $f(\boldsymbol{x}_1) \leqslant f(\boldsymbol{x}_0)$. Note that we use the transpose for the gradient since otherwise the dimensions will not work out.

This observation allows us to define a simple gradient-descent algorithm: If we want to find a local optimum $f(\boldsymbol{x}_*)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, we start with an initial guess $\boldsymbol{x}_0$ of the parameters we wish to optimize and then iterate according to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top . \tag{7.6}$$

For suitable step size γ_i, the sequence $f(\boldsymbol{x}_0) \geqslant f(\boldsymbol{x}_1) \geqslant \ldots$ converges to a local minimum.

Example 7.1
Consider a quadratic function in two dimensions

$$f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \begin{bmatrix}x_1\\x_2\end{bmatrix}^\top \begin{bmatrix}2 & 1\\1 & 20\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - \begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \tag{7.7}$$

with gradient

$$\nabla f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \begin{bmatrix}2 & 1\\1 & 20\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - \begin{bmatrix}5\\3\end{bmatrix}. \tag{7.8}$$

Starting at the initial location $\boldsymbol{x}_0 = [-3, -1]^\top$, we iteratively apply (7.6) to obtain a sequence of estimates that converge to the minimum value (illustrated in Figure 7.3). We can see (both from the figure and by plugging $\boldsymbol{x}_0$ into (7.8)) that the the gradient at $\boldsymbol{x}_0$ points north and east, leading to $\boldsymbol{x}_1 = [-1.98, 1.21]^\top$. Repeating that argument gives us $\boldsymbol{x}_2 = [-1.32, -0.42]^\top$, and so on.

Remark. Gradient descent can be relatively slow close to the minimum: Its asymptotic rate of convergence is inferior to many other methods. Using the ball rolling down the hill analogy, when the surface is a long thin valley the problem is poorly conditioned (Trefethen and Bau III, 1997). For poorly conditioned convex problems, gradient descent increasingly 'zigzags' as the gradients point nearly orthogonally to the shortest direction to a minimum point, see Fig. 7.3. ♢

7.1.1 Stepsize

As mentioned earlier, choosing a good stepsize is important in gradient descent. If the stepsize is too small, gradient descent can be slow. If the stepsize is chosen too large, gradient descent can overshoot, fail to converge, or even diverge. We will discuss the use of momentum in the next section. It is a method that smoothes out erratic behavior of gradient updates and dampens oscillations.

The stepsize is also called the learning rate

Adaptive gradient methods rescale the stepsize at each iteration, depending on local properties of the function. There are two simple heuristics (Toussaint, 2012):

- When the function value increases after a gradient step, the step size was too large. Undo the step and decrease the stepsize.
- When the function value decreases the step could have been larger. Try to increase the stepsize.

Although the "undo" step seems to be a waste of resources, using this heuristic guarantees monotonic convergence.

Example 7.2 (Solving a Linear Equation System)
When we solve linear equations of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, in practice we solve $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$ approximately by finding $\boldsymbol{x}_*$ that minimizes the the squared error

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{7.9}$$

if we use the Euclidean norm. The gradient of (7.9) with respect to $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} = 2(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top \boldsymbol{A}\,. \tag{7.10}$$

We can use this gradient directly in a gradient descent algorithm. However for this particular special case, it turns out that there is an analytic solution, which can be found by setting the gradient to zero. We can see that this analytic solution is given by $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We will see more on solving squared error problems in Chapter 9.

Remark. When applied to the solution of linear systems of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ gradient descent may converge slowly. The speed of convergence of gradient descent is dependent on the *condition number* $\kappa = \frac{\sigma(\boldsymbol{A})_{\max}}{\sigma(\boldsymbol{A})_{\min}}$, which is the ratio of the maximum to the minimum singular value (Section 4.5) of $\boldsymbol{A}$. The condition number essentially measures the ratio of the most curved direction versus the least curved direction, which corresponds to our imagery that poorly conditioned problems are long thin valleys: they are very curved in one direction, but very flat in the other. Instead of directly solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, one could instead solve $\boldsymbol{P}^{-1}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{0}$, where $\boldsymbol{P}$ is called the *preconditioner*. The goal is to design $\boldsymbol{P}^{-1}$ such that $\boldsymbol{P}^{-1}\boldsymbol{A}$ has a better condition number, but at the same time $\boldsymbol{P}^{-1}$ is easy to compute. For further information on gradient descent, pre-conditioning and convergence we refer to (Boyd and Vandenberghe, 2004, Chapter 9). ♢

condition number

preconditioner

7.1.2 Gradient Descent with Momentum

As illustrated in Figure 7.3, the convergence of gradient descent may be very slow if the curvature of the optimization surface is such that the there are regions which are poorly scaled. The curvature is such that the gradient descent steps hops between the walls of the valley, and approaches the optimum in small steps. The proposed tweak to improve convergence is to give gradient descent some memory.

Gradient descent with momentum (Rumelhart et al., 1986) is a method that introduces an additional term to remember what happened in the previous iteration. This memory dampens oscillations and smoothes out the gradient updates. Continuing the ball analogy, the momentum term emulates the phenomenon of a heavy ball which is reluctant to change directions. The idea is to have a gradient update with memory to imple-

Goh (2017) wrote an intuitive blog post on gradient descent with momentum.

ment a moving average. The momentum-based method remembers the update $\Delta \boldsymbol{x}_i$ at each iteration i and determines the next update as a linear combination of the current and previous gradients

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top + \alpha \Delta \boldsymbol{x}_i \tag{7.11}$$

$$\Delta \boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} = -\gamma_{i-1}((\nabla f)(\boldsymbol{x}_{i-1}))^\top, \tag{7.12}$$

where $\alpha \in [0, 1]$. Sometimes we will only know the gradient approximately. In such cases the momentum term is useful since it averages out different noisy estimates of the gradient. One particularly useful way to obtain an approximate gradient is using a stochastic approximation, which we discuss next.

7.1.3 Stochastic Gradient Descent

Computing the gradient can be very time consuming. However, often it is possible to find a "cheap" approximation of the gradient. Approximating the gradient is still useful as long as it points in roughly the same direction as the true gradient.

Stochastic gradient descent

Stochastic gradient descent (often shortened in SGD) is a stochastic approximation of the gradient descent method for minimizing an objective function that is written as a sum of differentiable functions. The word stochastic here refers to the fact that we acknowledge that we do not know the gradient precisely, but instead only know a noisy approximation to it. By constraining the probability distribution of the approximate gradients, we can still theoretically guarantee that SGD will converge.

In machine learning given $n = 1, \ldots, N$ data points, we often consider objective functions which are the sum of the losses L_n incurred by each example n. In mathematical notation we have the form

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} L_n(\boldsymbol{\theta}) \tag{7.13}$$

where $\boldsymbol{\theta}$ is the vector of parameters of interest, i.e., we want to find $\boldsymbol{\theta}$ that minimizes L. An example from regression (Chapter 9), is the negative log-likelihood, which is expressed as a sum over log-likelihoods of individual examples,

$$L(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n | \boldsymbol{x}_n, \boldsymbol{\theta}) \tag{7.14}$$

where $\boldsymbol{x}_n \in \mathbb{R}^D$ are the training inputs, y_n are the training targets and $\boldsymbol{\theta}$ are the parameters of the regression model.

Standard gradient descent, as introduced previously, is a "batch" optimization method, i.e., optimization is performed using the full training set

by updating the vector of parameters according to

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma_i (\nabla L(\boldsymbol{\theta}_i))^\top = \boldsymbol{\theta}_i - \gamma_i \sum_{n=1}^{N} (\nabla L_n(\boldsymbol{\theta}_i))^\top \tag{7.15}$$

for a suitable stepsize parameter γ_i. Evaluating the sum-gradient may require expensive evaluations of the gradients from all individual functions L_n. When the training set is enormous and/or no simple formulas exist, evaluating the sums of gradients becomes very expensive.

Consider the term $\sum_{n=1}^{N}(\nabla L_n(\boldsymbol{\theta}_i))$ in (7.15) above: we can reduce the amount of computation by taking a sum over a smaller set of L_n. In contrast to batch gradient descent, which uses all L_n for $n = 1, \ldots, N$, we randomly choose a subset of L_n for mini-batch gradient descent. In the extreme case, we randomly select only a single L_n to estimate the gradient. The key insight about why taking a subset of data is sensible is to realise that for gradient descent to converge, we only require that the gradient is an unbiased estimate of the true gradient. In fact the term $\sum_{n=1}^{N}(\nabla L_n(\boldsymbol{\theta}_i))$ in (7.15) is an empirical estimate of the expected value (Section 6.4.1) of the gradient. Therefore any other unbiased empirical estimate of the expected value, for example using any subsample of the data, would suffice for convergence of gradient descent.

Why should one consider using an approximate gradient? A major reason is practical implementation constraints, such as the size of CPU/GPU memory or limits on computational time. We can think of the size of the subset used to estimate the gradient in the same way that we thought of the size of a sample when estimating empirical means 6.4.1. In practice, it is good to keep the size of the mini-batch as large as possible. Large mini-batches reduce the variance in the parameter update. Furthermore large mini-batches take advantage of highly optimized matrix operations in vectorized implementations of the cost and gradient. However when we choose the mini-batch size, we need to make sure it fits into CPU/GPU memory. Typical mini-batch sizes are 64, 128, 256, 512, 1024, which depends on the way computer memory is laid out and accessed.

This often leads to more stable convergence since the gradient estimator is less noisy.

Remark. When the learning rate decreases at an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to local minimum (Bottou, 1998). ♢

If we keep the mini-batch size small, the noise in our gradient estimate will allow us to get out of some bad local optima, which we may otherwise get stuck in.

Stochastic gradient descent is very effective in large-scale machine learning problems (Bottou et al., 2018), such as training deep neural networks on millions of images (Dean et al., 2012), topic models (Hoffman et al., 2013), reinforcement learning (Mnih et al., 2015) or training large-scale Gaussian process models (Hensman et al., 2013; Gal et al., 2014).

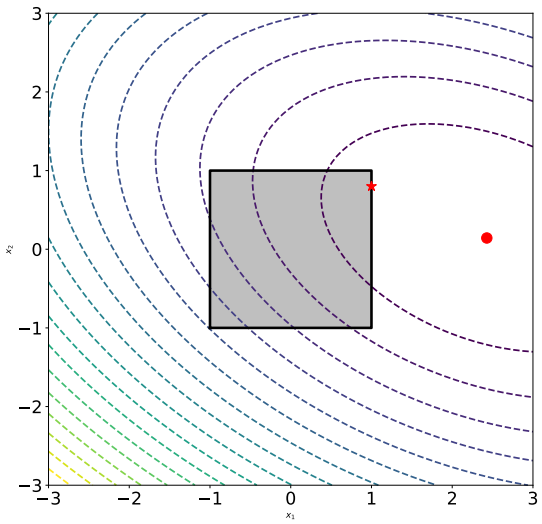


Figure 7.4 Illustration of constrained optimization. The unconstrained problem (indicated by the contour lines) has a minimum on the right side (indicated by the circle). The box constraints ($-1 \leqslant x \leqslant 1$ and $-1 \leqslant y \leqslant 1$) require that the optimal solution are within the box, resulting in an optimal value indicated by the star.

7.2 Constrained Optimization and Lagrange Multipliers

In the previous section, we considered the problem of solving for the minimum of a function

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{7.16}$$

where $f : \mathbb{R}^D \to \mathbb{R}$.

In this section we have additional constraints. That is for real valued functions $g_i : \mathbb{R}^D \to \mathbb{R}$ for $i = 1, \ldots, m$ we consider the constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \end{aligned} \tag{7.17}$$

It is worth pointing out that the functions f and g_i could be non-convex in general, and we will consider the convex case in the next section.

One obvious, but not very practical, way of converting the constrained problem (7.17) into an unconstrained one is to use an indicator function

$$J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \mathbf{1}(g_i(\boldsymbol{x})) \tag{7.18}$$

where $\mathbf{1}(z)$ is an infinite step function

$$\mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leqslant 0 \\ \infty & \text{otherwise} \end{cases} . \tag{7.19}$$

This gives infinite penalty if the constraint is not satisfied, and hence would provide the same solution. However, this infinite step function is equally difficult to optimize. We can overcome this difficulty by introducing *Lagrange multipliers*. The idea of Lagrange multipliers is to replace the step function with a linear function. Lagrange multipliers

We associate to problem (7.17) the *Lagrangian* by introducing the Lagrange multipliers $\lambda_i \geqslant 0$ corresponding to each inequality constraint respectively (Boyd and Vandenberghe, 2004, Chapter 4). Lagrangian

$$\begin{aligned}\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \\ &= f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x}) \end{aligned} \tag{7.20}$$

where in the last line we have a concatenated all constraints $g_i(\boldsymbol{x})$ into a vector $\boldsymbol{g}(\boldsymbol{x})$, and all the Lagrange multipliers into a vector $\boldsymbol{\lambda} \in \mathbb{R}^m$.

We now introduce the idea of Lagrangian duality. In general, duality in optimization is the idea of converting an optimization problem in one set of variables $\boldsymbol{x}$ (called the primal variables), into another optimization problem in a different set of variables $\boldsymbol{\lambda}$ (called the dual variables). We introduce two different approaches to duality: in this section we discuss Lagrangian duality, and in Section 7.3.3 we discuss Legendre-Fenchel duality.

Theorem 7.1. *The problem in* (7.17)

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \textit{for all} \quad i = 1, \ldots, m \end{aligned}$$

is known as the primal problem, corresponding to the primal variables x. *The associated Lagrangian dual problem is given by* primal problem; Lagrangian dual problem

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad \mathfrak{D}(\boldsymbol{\lambda}) \tag{7.21}$$

$$\text{subject to} \quad \boldsymbol{\lambda} \geqslant 0, \tag{7.22}$$

where $\boldsymbol{\lambda}$ *are the dual variables and* $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.

Proof Recall that the difference between $J(\boldsymbol{x})$ in (7.18) and the Lagrangian in (7.20) is that we have relaxed the indicator function to a linear function. Therefore when $\lambda \geqslant 0$, the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a lower bound of $J(\boldsymbol{x})$. Hence the maximum of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is $J(\boldsymbol{x})$

$$J(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geqslant 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{7.23}$$

Recall that the original problem was minimising $J(\boldsymbol{x})$,

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geqslant \boldsymbol{0}} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{7.24}$$

By the minimax inequality (Boyd and Vandenberghe, 2004) it turns out

that, for any function swapping the order of the minimum and maximum above results in a smaller value.

$$\min_{\boldsymbol{x}\in\mathbb{R}^d}\max_{\boldsymbol{\lambda}\geqslant\mathbf{0}}\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda})\geqslant\max_{\boldsymbol{\lambda}\geqslant\mathbf{0}}\min_{\boldsymbol{x}\in\mathbb{R}^d}\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda}) \tag{7.25}$$

weak duality

This is also known as *weak duality*. Note that the inner part of the right hand side is the dual objective function $\mathfrak{D}(\boldsymbol{\lambda})$ and the theorem follows. □

In contrast to the original optimization problem which has constraints, $\min_{\boldsymbol{x}\in\mathbb{R}^d}\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda})$ is an unconstrained optimization problem for a given value of $\boldsymbol{\lambda}$. If solving $\min_{\boldsymbol{x}\in\mathbb{R}^d}\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda})$ is easy, then the overall problem is easy to solve. The reason is that the outer problem (maximization over $\boldsymbol{\lambda}$) is a maximum over a set of affine functions, and hence is a concave function, even though $f(\cdot)$ and $g_i(\cdot)$ may be non-convex. The maximum of a concave function can be efficiently computed.

Assuming $f(\cdot)$ and $g_i(\cdot)$ are differentiable, we find the Lagrange dual problem by differentiating the Lagrangian with respect to $\boldsymbol{x}$ and setting the differential to zero and solving for the optimal value. We will discuss two concrete examples in Section 7.3.1 and 7.3.2, where $f(\cdot)$ and $g_i(\cdot)$ are convex.

Remark (Equality constraints). Consider (7.17) with additional equality constraints

$$\begin{aligned}\min_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{subject to } & g_i(\boldsymbol{x})\leqslant 0 \quad \text{for all} \quad i=1,\dots,m \\ & h_j(\boldsymbol{x})=0 \quad \text{for all} \quad j=1,\dots,n\end{aligned} \tag{7.26}$$

We can model equality constraints by replacing them with two inequality constraints. That is for each equality constraint $h_j(\boldsymbol{x})=0$ we equivalently replace it by two constraints $h_j(\boldsymbol{x})\leqslant 0$ and $h_j(\boldsymbol{x})\geqslant 0$. It turns out that the resulting Lagrange multipliers are then unconstrained.

Therefore we constrain the Lagrange multipliers corresponding to the inequality constraints in (7.26) to be non-negative, and leave the Lagrange multipliers corresponding to the equality constraints unconstrained. ♢

7.3 Convex Optimization

We focus our attention of a particularly useful class of optimization problems, where we can guarantee global optimality. When $f(\cdot)$ is a convex function, and when the constraints involving $g(\cdot)$ and $h(\cdot)$ are convex sets, this is called a *convex optimization problem*. In this setting, we have *strong duality*: The optimal solution of the dual problem is the same as the optimal solution of the primal problem. The distinction between *convex functions* and *convex sets* are often not strictly presented in machine learning literature, but one can often infer the implied meaning from context.

convex optimization problem
strong duality
convex functions
convex sets

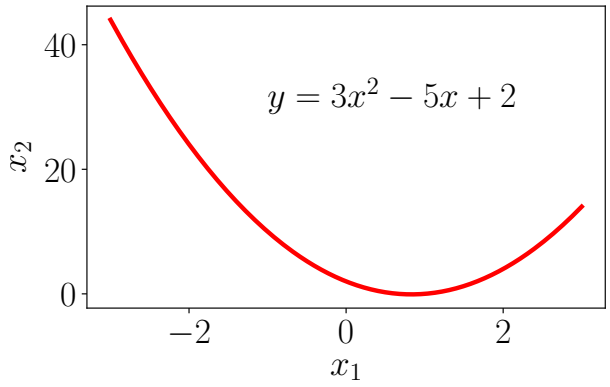


Figure 7.5 Example of a convex function.

Convex functions are functions such that a straight line between any two points of the function lie above the function. Figure 7.2 shows a nonconvex function and Figure 7.3 shows a convex function. Another convex function is shown in Figure 7.5.

Definition 7.2. A function $f : \mathbb{R}^D \to \mathbb{R}$ is a *convex function* if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f(\boldsymbol{x}) + (1-\theta)f(\boldsymbol{y}) \tag{7.27}$$

convex function

Technically, the domain of the function f must also be a convex set.

Remark. A *concave function* is the negative of a convex function. ♢

concave function

The constraints involving $g(\cdot)$ and $h(\cdot)$ in (7.26) truncate functions at a scalar value, resulting in sets. Another relation between convex functions and convex sets is to consider the set obtained by "filling in" a convex function. A convex function is a bowl like object, and we imagine pouring water into it to fill it up. This resulting filled in set, called the epigraph of the convex function, is a convex set. Convex sets are sets such that a straight line connecting any two elements of the set lie inside the set. Figure 7.6 and Figure 7.7 illustrates convex and nonconvex sets respectively.

Figure 7.6 Example of a convex set

Definition 7.3. A set C is a *convex set* if for any $x, y \in C$ and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$\theta x + (1-\theta)y \in C \tag{7.28}$$

Figure 7.7 Example of a nonconvex set

convex set

If a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, we can specify convexity in terms of its gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ (Section 5.2). A function $f(\boldsymbol{x})$ is convex if and only if for any two points $\boldsymbol{x}, \boldsymbol{y}$,

$$f(\boldsymbol{y}) \geqslant f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}). \tag{7.29}$$

If we further know that a function $f(\boldsymbol{x})$ is twice differentiable, that is the Hessian (5.144) exists for all values in the domain of $\boldsymbol{x}$, then the function $f(\boldsymbol{x})$ is convex if and only if $\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$ is positive semi-definite (Boyd and Vandenberghe, 2004).

Example 7.3

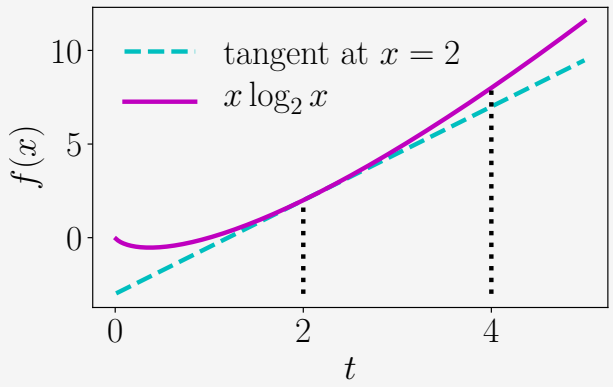


Figure 7.8 The negative entropy function (which is convex), and its tangent at $x = 2$.

The negative entropy $f(x) = x \log_2 x$ is convex for $x > 0$. A visualization of the function is show in Figure 7.8, and we can see that the function is convex. To illustrate the above definitions of convexity, let us check the calculations for two points $x = 2$ and $x = 4$. Note that to prove convexity of $f(x)$ we would need to check for all points $x \in \mathbb{R}$.

Recall Definition 7.2. Consider a point midway between the two points (that is $\theta = 0.5$), then the left hand side is $f(0.5 \times 2 + 0.5 \times 4) = 3 \log_2 3 \approx 4.75$. The right hand side is $0.5(2 \log_2 2) + 0.5(4 \log_2 4) = 1 + 4 = 5$. And therefore the definition is satisfied.

Since $f(x)$ is differentiable, we can alternatively use (7.29). Calculating the derivative of $f(x)$, we obtain

$$\nabla_x (x \log_2 x) = 1 \times \log_2 x + x \times \frac{1}{x} \tag{7.30}$$

$$= \log_2 x + 1. \tag{7.31}$$

Using the same two test points $x = 2$ and $x = 4$, the left hand side of (7.29) is given by $f(4) = 8$. The right hand side is

$$f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}^\top (\boldsymbol{y} - \boldsymbol{x}) = f(2) + \nabla f(2) \times (4 - 2) \tag{7.32}$$

$$= 2 + 2 \times 2 = 6. \tag{7.33}$$

We can check that a function or set is convex from first principles by recalling the definitions. In practice we often rely on operations that preserve convexity to check that a particular function or set is convex. Although the details are vastly different, this is again the idea of closure that we introduced in Chapter 2 for vector spaces.

Example 7.4
A nonnegative weighted sum of convex functions is convex. Observe that if f is a convex function, and $\alpha \geqslant 0$ is a nonnegative scalar, then the function αf is convex. We can see this by multiplying α to both sides of equation in Definition 7.2, and recalling that multiplying a nonnegative number does not change the inequality.

If f_1 and f_2 are convex functions, then we have by the definition

$$f_1(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f_1(\boldsymbol{x}) + (1-\theta) f_1(\boldsymbol{y}) \tag{7.34}$$
$$f_2(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f_2(\boldsymbol{x}) + (1-\theta) f_2(\boldsymbol{y}). \tag{7.35}$$

Summing up both sides gives us

$$\begin{aligned} &f_1(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) + f_2(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \\ &\leqslant \theta f_1(\boldsymbol{x}) + (1-\theta) f_1(\boldsymbol{y}) + \theta f_2(\boldsymbol{x}) + (1-\theta) f_2(\boldsymbol{y}) \end{aligned} \tag{7.36}$$

where the right hand side can be rearranged to

$$\theta(f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})) + (1-\theta)(f_1(\boldsymbol{y}) + f_2(\boldsymbol{y})) \tag{7.37}$$

completing the proof that the sum of convex functions is convex.

Combining the two facts above, we see that $\alpha f_1(\boldsymbol{x}) + \beta f_2(\boldsymbol{x})$ is convex for $\alpha, \beta \geqslant 0$. This closure property can be extended using a similar argument for nonnegative weighted sums of more than two convex functions.

Remark. The inequality defining convex functions, see 7.27, is sometimes called *Jensen's inequality*. In fact a whole class of inequalities for taking nonnegative weighted sums of convex functions are all called Jensen's inequality. ♢

Jensen's inequality

In summary, a constrained optimization problem is called a *convex optimization problem* if

convex optimization problem

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\text{subject to } g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \\ &\qquad\qquad h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \ldots, n \end{aligned} \tag{7.38}$$

where all the functions $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ are convex functions, and all $h_j(\boldsymbol{x}) = 0$ are convex sets. In the following two subsections, we will describe two classes convex optimization problems that are widely used and well understood.

7.3.1 Linear Programming

Consider the special case when all the functions above are linear, that is

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \boldsymbol{c}^\top \boldsymbol{x} \tag{7.39}$$

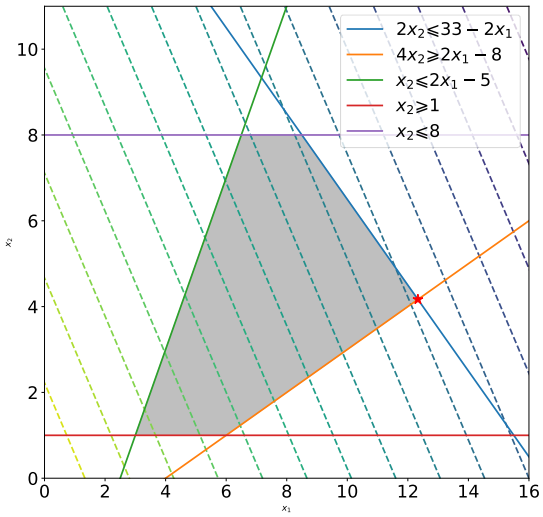


Figure 7.9 Illustration of a linear program. The unconstrained problem (indicated by the contour lines) has a minimum on the right side. The optimal value given the constraints are shown by the star.

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$$

Linear programs are one of the most widely used approaches in industry.

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$. This is known as a *linear program*. It has d variables and m linear constraints.

Example 7.5
An example of a linear program is illustrated in Figure 7.9, which has two variables. The objective function is linear, resulting in linear contour lines. The constraint set in standard form is translated into the legend. The optimal value must lie in the shaded (feasible) region, and is indicated by the star.

$$\min_{\boldsymbol{x} \in \mathbb{R}^2} -\begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \tag{7.40}$$

$$\text{subject to } \begin{bmatrix}2 & 2\\2 & -4\\-2 & 1\\0 & -1\\0 & 1\end{bmatrix} \begin{bmatrix}x_1\\x_2\end{bmatrix} \leqslant \begin{bmatrix}33\\8\\5\\-1\\8\end{bmatrix} \tag{7.41}$$

The Lagrangian is given by

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

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of non-negative Lagrange multipliers. It is easier to see what is going on by rearranging the terms corresponding to $\boldsymbol{x}$.

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

Taking the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ and setting it to zero gives us

$$\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = \boldsymbol{0}.$$

Therefore the dual Lagrangian is $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^\top \boldsymbol{b}$. Recall we would like to maximize $\mathfrak{D}(\boldsymbol{\lambda})$. In addition to the constraint due to the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ being zero, we also have the fact that $\boldsymbol{\lambda} \geqslant \boldsymbol{0}$, resulting in the following dual optimization problem

It is convention to minimize the primal and maximize the dual.

$$\begin{aligned} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} & -\boldsymbol{b}^\top \boldsymbol{\lambda} \\ \text{subject to } & \boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = \boldsymbol{0} \\ & \boldsymbol{\lambda} \geqslant \boldsymbol{0}. \end{aligned} \tag{7.42}$$

This is also a linear program, but with m variables. We have the choice of solving the primal (7.39) or the dual (7.42) program depending on whether m or d is larger. Recall that d is the number of variables and m is the number of constraints in the primal linear program.

7.3.2 Quadratic Programming

Consider when the objective function is a convex quadratic function, and the constraints are affine,

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^d} & \frac{1}{2} \boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to } & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \end{aligned} \tag{7.43}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^d$. The square symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{d \times d}$ is positive definite, and therefore the objective function is convex. This is known as a *quadratic program*. Observe that it has d variables and m linear constraints.

Example 7.6
An example of a quadratic program is illustrated in Figure 7.4, which has two variables. The objective function is quadratic with a positive semidefinite matrix $\boldsymbol{Q}$, resulting in elliptical contour lines. The optimal value must lie in the shaded (feasible) region, and is indicated by the star.

$$\min_{\boldsymbol{x} \in \mathbb{R}^2} \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{7.44}$$

$$\text{subject to } \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leqslant \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{7.45}$$

The Lagrangian is given by

$$\begin{aligned} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \\ &= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda})^\top \boldsymbol{x} - \boldsymbol{\lambda}^\top \boldsymbol{b} \end{aligned}$$

where again we have rearranged the terms. Taking the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ and setting it to zero gives

$$\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda}) = 0$$

Assuming that $\boldsymbol{Q}$ is invertible, we get

$$\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda}) \tag{7.46}$$

Substituting (7.46) into the primal Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ we get the dual Lagrangian

$$\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + A^\top \boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b}$$

Therefore the dual optimization problem is given by

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b} \tag{7.47}$$

$$\text{subject to } \boldsymbol{\lambda} \geqslant 0. \tag{7.48}$$

We will see an application of Quadratic Programming in machine learning in Chapter 12.

7.3.3 Legendre-Fenchel Transform and Convex Conjugate

Let us revisit the idea of duality, which we saw in Section 7.2, without considering constraints. One useful fact about convex sets is that a convex set can be equivalently described by its supporting hyperplanes. A hyperplane is called a *supporting hyperplane* of a convex set if it intersects the convex set and the convex set is contained on just one side of it. Recall that for a convex function, we can fill it up to obtain the epigraph which is a convex set. Therefore we can also describe convex functions in terms of their supporting hyperplanes. Furthermore observe that the supporting hyperplane just touches the convex function, and is in fact the tangent to the function at that point. And recall that the tangent of a function $f(\boldsymbol{x})$ at

supporting hyperplane

a given point $\boldsymbol{x}_0$ is the evaluation of the gradient of that function at that point $\frac{\mathrm{d}f(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{x}_0}$. In summary, because convex sets can be equivalently described by its supporting hyperplanes, convex functions can be equivalently described by a function of their gradient. The *Legendre transform* formalizes this concept .

Legendre transform

Physics students are often introduced to the Legendre transform as relating the Lagrangian and the Hamiltonian in classical mechanics.

We begin with the most general definition which unfortunately has a counterintuitive form, and look at special cases to try to relate the definition to the intuition above. The *Legendre-Fenchel transform* is a transformation (in the sense of a Fourier transform) from a convex differentiable function $f(\boldsymbol{x})$ to a function that depends on the tangents $s(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. It is worth stressing that this is a transformation of the function $f(\cdot)$ and not the variable $\boldsymbol{x}$ or the function evaluated at a value. The Legendre-Fenchel transform is also known as the convex conjugate (for reasons we will see soon) and is closely related to duality (Hiriart-Urruty and Lemaréchal, 2001, Chapter 5).

Legendre-Fenchel transform

Definition 7.4. The *convex conjugate* of a function $f : \mathbb{R}^D \to \mathbb{R}$ is a function f^* defined by

convex conjugate

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}\in\mathbb{R}^D} \langle \boldsymbol{s}, \boldsymbol{x}\rangle - f(\boldsymbol{x}) \tag{7.49}$$

Note that the convex conjugate definition above does not need the function f to be convex nor differentiable. In the definition above, we have used a general inner product (Section 3.2) but in the rest of this section we will consider the standard dot product between finite dimensional vectors ($\langle \boldsymbol{s}, \boldsymbol{x}\rangle = \boldsymbol{s}^\top \boldsymbol{x}$) to avoid too many technical details.

To understand the above definition in a geometric fashion, consider an nice simple one dimensional convex and differentiable function, for example $f(x) = x^2$. Note that since we are looking at a one dimensional problem, hyperplanes reduce to a line. Consider a line $y = sx + c$. Recall that we are able to describe convex functions by their supporting hyperplanes, so let us try to describe this function $f(x)$ by its supporting lines. Fix the gradient of the line $s \in \mathbb{R}$ and for each point $(x_0, f(x_0))$ on the graph of f, find the minimum value of c such that the line still intersects $(x_0, f(x_0))$. Note that the minimum value of c is the place where a line with slope s "just touches" the function $f(x) = x^2$. The line passing through $(x_0, f(x_0))$ with gradient s is given by

This derivation is easiest to understand by drawing the reasoning as it progresses.

$$y - f(x_0) = s(x - x_0) \tag{7.50}$$

The y-intercept of this line is $-sx_0 + f(x_0)$. The minimum of c for which $y = sx + c$ intersects with the graph of f is therefore

$$\inf_{x_0} -sx_0 + f(x_0). \tag{7.51}$$

The convex conjugate above is by convention defined to be the negative of this. The reasoning in this paragraph did not rely on the fact that we

chose a one dimensional convex and differentiable function, and holds for $f : \mathbb{R}^D \to \mathbb{R}$ which are nonconvex and non differentiable.

The classical Legendre transform is defined on convex differentiable functions in $\mathbb{R}^D$

Remark. Convex differentiable functions such as the example $f(x) = x^2$ is a nice special case, where there is no need for the supremum, and there is a one to one correspondence between a function and its Legendre transform. Let us derive this from first principles. For a convex differentiable function, we know that at x_0 the tangent touches $f(x_0)$, therefore

$$f(x_0) = sx_0 + c. \tag{7.52}$$

Recall that we want to describe the convex function $f(x)$ in terms of its gradient $\nabla_x f(x)$, and that $s = \nabla_x f(x_0)$. We rearrange to get an expression for $-c$ to obtain

$$-c = sx_0 - f(x_0). \tag{7.53}$$

Note that $-c$ changes with x_0 and therefore with s, which is why we can think of it as a function of s, which we call $f^*(s)$.

$$f^*(s) = sx_0 - f(x_0). \tag{7.54}$$

Compare (7.54) with Definition 7.4, and observe that (7.54) is a special case (without the supremum). ♢

The conjugate function has nice properties, for example for convex functions, applying the Legendre transform again gets us back to the original function. In the same way that the slope of $f(x)$ is s, the slope of $f^*(s)$ is x. The following two examples show common uses of convex conjugates in machine learning.

Example 7.7
To illustrate the application of convex conjugates, consider the quadratic function based on a positive definite matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$. We denote the primal variable to be $\boldsymbol{y} \in \mathbb{R}^n$ and the dual variable to be $\boldsymbol{\alpha} \in \mathbb{R}^n$.

$$f(\boldsymbol{y}) = \frac{\lambda}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y} \tag{7.55}$$

Applying Definition 7.4, we obtain the function

$$f^*(\boldsymbol{\alpha}) = \sup_{\boldsymbol{y}\in\mathbb{R}^n} \langle \boldsymbol{y}, \boldsymbol{\alpha}\rangle - \frac{\lambda}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y}. \tag{7.56}$$

Observe that the function is differentiable, and hence we can find the maximum by taking the derivative and with respect to $\boldsymbol{y}$ setting it to zero.

$$\frac{\partial\left[\langle \boldsymbol{y}, \boldsymbol{\alpha}\rangle - \frac{\lambda}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\boldsymbol{y}\right]}{\partial \boldsymbol{y}} = (\boldsymbol{\alpha} - \lambda \boldsymbol{K}^{-1}\boldsymbol{y})^\top \tag{7.57}$$

and hence when the gradient is zero we have $\boldsymbol{y} = \frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}$. Substituting

into (7.56) yields

$$f^*(\boldsymbol{\alpha}) = \frac{1}{\lambda}\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} - \frac{\lambda}{2}\left(\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}\right)^\top \boldsymbol{K}^{-1}\left(\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}\right) \tag{7.58}$$

$$= \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha}\,. \tag{7.59}$$

Example 7.8
In machine learning we often use sums of functions, for example the objective function of the training set includes a sum of the losses for each example in the training set. In the following, we derive the convex conjugate of a sum of losses $\ell(t)$, where $\ell : \mathbb{R} \to \mathbb{R}$. This also illustrates the application of the convex conjugate to the vector case. Let $\mathcal{L}(\boldsymbol{t}) = \sum_{i=1}^n \ell_i(t_i)$,

$$\mathcal{L}^*(\boldsymbol{z}) = \sup_{\boldsymbol{t}\in\mathbb{R}^n} \langle \boldsymbol{z}, \boldsymbol{t}\rangle - \sum_{i=1}^n \ell_i(t_i) \tag{7.60}$$

$$= \sup_{\boldsymbol{t}\in\mathbb{R}^n} \sum_{i=1}^n z_i t_i - \ell_i(t_i) \qquad \text{definition of dot product} \tag{7.61}$$

$$= \sum_{i=1}^n \sup_{\boldsymbol{t}\in\mathbb{R}^n} z_i t_i - \ell_i(t_i) \tag{7.62}$$

$$= \sum_{i=1}^n \ell_i^*(z_i) \qquad \text{definition of conjugate} \tag{7.63}$$

Recall that in Section 7.2 we derived a dual optimization problem using Lagrange multipliers. Furthermore for convex optimization problems we have strong duality, that is the solutions of the primal and dual problem match. The Fenchel-Legendre transform described here also can be used to derive a dual optimization problem. Furthermore then the function is convex and differentiable, the supremum is unique. To further investigate the relation between these two approaches, let us consider a linear equality constrained convex optimization problem.

Example 7.9
Let $f(\boldsymbol{y})$ and $g(\boldsymbol{x})$ be convex functions, and $\boldsymbol{A}$ a real matrix of appropriate dimensions such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. Then

$$\min_{\boldsymbol{x}} f(\boldsymbol{A}\boldsymbol{x}) + g(\boldsymbol{x}) = \min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}). \tag{7.64}$$

By introducing the Lagrange multiplier $\boldsymbol{u}$ for the constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$,

$$\min_{\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) = \min_{\boldsymbol{x},\boldsymbol{y}} \max_{\boldsymbol{u}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.65}$$

$$= \max_{\boldsymbol{u}} \min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.66}$$

where the last step of swapping max and min is due to the fact that $f(\boldsymbol{y})$ and $g(\boldsymbol{x})$ are convex functions. By splitting up the dot product term and collecting $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$\max_{\boldsymbol{u}} \min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.67}$$

$$= \max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y})\right] + \left[\min_{\boldsymbol{x}} (\boldsymbol{A}\boldsymbol{x})^\top \boldsymbol{u} + g(\boldsymbol{x})\right] \tag{7.68}$$

$$= \max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y})\right] + \left[\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{u} + g(\boldsymbol{x})\right] \tag{7.69}$$

For general inner products, $\boldsymbol{A}^\top$ is replaced by the adjoint $\boldsymbol{A}^*$.

Recall the convex conjugate (Definition 7.4) and the fact that dot products are symmetric,

$$\max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y})\right] + \left[\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{u} + g(\boldsymbol{x})\right] \tag{7.70}$$

$$= \max_{\boldsymbol{u}} -f^*(\boldsymbol{u}) - g^*(-\boldsymbol{A}^\top \boldsymbol{u}). \tag{7.71}$$

Therefore we have shown that

$$\min_{\boldsymbol{x}} f(\boldsymbol{A}\boldsymbol{x}) + g(\boldsymbol{x}) = \max_{\boldsymbol{u}} -f^*(\boldsymbol{u}) - g^*(-\boldsymbol{A}^\top \boldsymbol{u}). \tag{7.72}$$

The Legendre-Fenchel conjugate turns out to be quite useful for machine learning problems that can be expressed as convex optimization problems. In particular for convex loss functions that apply independently to each example, the conjugate loss is a convenient way to derive a dual problem.

7.4 Further Reading

Continuous optimization is an active area of research, and we do not try to provide a comprehensive account of recent advances.

From a gradient descent perspective, there are two major weaknesses which each have their own set of literature. The first challenge is the fact that gradient descent is a first order algorithm, and does not use information about the curvature of the surface. When there are long valleys, the gradient points perpendicularly to the direction of interest. Conjugate gradient methods avoid the issues faced by gradient descent by taking previous directions into account (Shewchuk, 1994). Second order methods

such as Newton methods use the Hessian to provide information about the curvature. Many of the choices for choosing stepsizes and ideas like momentum arise by considering the curvature of the objective function (Goh, 2017; Bottou et al., 2018). Quasi-Newton methods such as L-BFGS try to use cheaper computational methods to approximate the Hessian (Nocedal and Wright, 2006). Recently there has been interest in other metrics for computing descent directions, resulting in approaches such as mirror descent (Beck and Teboulle, 2003) and natural gradient (Toussaint, 2012).

The second challenge are non-differentiable functions. Gradient methods are not well defined when there are kinks in the function. In these cases, *subgradient methods* can be used (Shor, 1985). For further information and algorithms for optimizing non-differentiable functions, we refer to the book by Bertsekas (1999). There is a vast amount of literature on different approaches for numerically solving continuous optimization problems, including algorithms for constrained optimization problems. A good starting point to appreciate this literature are Luenberger (1969) and Bonnans et al. (2006). A recent survey of continuous optimization is Bubeck (2015).

Modern applications of machine learning often mean that the size of datasets prohibit the use of batch gradient descent, and hence stochastic gradient descent is the current workhorse of large scale machine learning methods. Recent surveys of the literature include (Hazan, 2015; Bottou et al., 2018).

For duality and convex optimization, the book by Boyd and Vandenberghe (Boyd and Vandenberghe, 2004) includes lectures and slides online. A more mathematical treatment is provided by Bertsekas (2009). Convex optimization is based upon convex analysis, and the reader interested in more foundational results about convex functions is referred to Hiriart-Urruty and Lemaréchal (2001); Rockafellar (1970); Borwein and Lewis (2006). Legendre-Fenchel transforms are also covered in the above books on convex analysis, but more beginner friendly presentations are available at Zia et al. (2009); Gonçalves (2014). The role of Legendre-Fenchel transforms in the analysis of convex optimization algorithms is surveyed in Polyak (2016).

Exercises

7.1 Consider the univariate function

$$f(x) = x^3 + 2x^2 + 5x - 3.$$

Find its stationary points and indicate whether they are maximum, minimum or saddle points.

7.2 Consider the update equation for stochastic gradient descent (Equation (7.15)). Write down the update when we use a mini-batch size of one.

7.3 Express the following optimization problem as a standard linear program in

matrix notation

$$\max_{\boldsymbol{x}\in\mathbb{R}^2,\xi\in\mathbb{R}} \boldsymbol{p}^\top \boldsymbol{x} + \xi$$

4203 subject to the constraints that $\xi \geqslant 0$, $x_0 \leqslant 0$ and $x_1 \leqslant 3$.

7.4 The hinge loss (which is the loss used by the Support Vector Machine) is given by

$$L(\alpha) = \max\{0, 1 - \alpha\}$$

If we are interested in applying gradient methods such as L-BFGS, and do not want to resort to subgradient methods, we need to smooth the kink in the hinge loss. Compute the convex conjugate of the hinge loss $L^*(\beta)$ where β is the dual variable. Add a ℓ_2 proximal term, and compute the conjugate of the resulting function

$$L^*(\beta) + \frac{\gamma}{2}\beta^2$$

4204 where γ is a given hyperparameter.

8

When Models meet Data

In the first part of the book, we introduced the mathematics that form the foundations of many machine learning methods. The hope is that a reader would be able to learn the rudimentary forms of the language of mathematics, which we will now use to describe and discuss machine learning. The second part of the book introduces four pillars of machine learning:

- Regression (Chapter 9)
- Dimensionality reduction (Chapter 10)
- Density estimation (Chapter 11)
- Classification (Chapter 12)

Recall from Table 1.1 that these problems illustrate two supervised and two unsupervised learning methods — one discrete and another continuous. The main aim of this part of the book is to illustrate how the mathematical concepts introduced in the first part of the book can be used to design machine learning algorithms that can be used to solve tasks within the remit of the four pillars. We do not intend to introduce advanced machine learning concepts, but instead to provide a set of practical methods that allow the reader to apply the knowledge they had gained from the first part of the book. It also provides a gateway to the wider machine learning literature for readers already familiar with the mathematics.

It is worth at this point to pause and consider the problem that a machine learning algorithm is designed to solve. As discussed in Chapter 1, there are three major components of a machine learning system: data, models and learning. The main question of machine learning is "what do we mean by good models?". That is we are interested to find models that perform well on future data. The word *model* has many subtleties and we will revisit it multiple times in this chapter. It is also not entirely obvious how to objectively define the word "good", and one of the guiding principles of machine learning is that good models should perform well on unseen data. This requires us to define some performance metrics, such as accuracy or distance from ground truth, as well as figuring out ways to do well (under these performance metrics).

model

This chapter covers a few necessary bits and pieces of mathematical and statistical language that are commonly used to talk about machine

Table 8.1 Example data from a fictitious human resource database that is not in a numerical format.

Name	Gender	Degree	Postcode	Age	Annual Salary
Aditya	M	MSc	W21BG	36	89563
Bob	M	PhD	EC1A1BA	47	123543
Chloé	F	BEcon	SW1A1BH	26	23989
Daisuke	M	BSc	SE207AT	68	138769
Elisabeth	F	MBA	SE10AA	33	113888

Table 8.2 Example data from a fictitious human resource database (see Table 8.1), converted to a numerical format.

Gender ID	Degree	Latitude (in degrees)	Longitude (in degrees)	Age	Annual Salary (in thousands)
-1	2	51.5073	0.1290	36	89.563
-1	3	51.5074	0.1275	47	123.543
+1	1	51.5071	0.1278	26	23.989
-1	1	51.5075	0.1281	68	138.769
+1	2	51.5074	0.1278	33	113.888

learning models. By doing so, we briefly outline the current best practices for training a model such that we do well on data that we have not yet seen. We will introduce the framework for non-probabilistic models in Section 8.1, the principle of maximum likelihood in Section 8.2, and the idea of probabilistic models in Section 8.3. We briefly outline a graphical language for specifying probabilistic models in Section 8.4 and finally discuss model selection in Section 8.5. The rest of this section expands upon the three main components of machine learning: data, models and learning.

Data as Vectors

We assume that our data can be read by a computer, and represented adequately in a numerical format. Furthermore, data is assumed to be tabular, where we think of each row of the table to represent a particular instance or example, and each column to be a particular feature/representation of the instance. We do not discuss the important and challenging aspects of identifying good representations (features). Many of these aspects depend on domain expertise and require careful engineering, which in recent years have been put under the umbrella of data science (Stray, 2016; Adhikari and DeNero, 2018). For example, in Table 8.1, the gender column (a categorical variable) may be converted into numbers 0 representing "Male" and 1 representing "Female". Alternatively, the gender could be represented by numbers $-1, +1$, respectively (as shown in Table 8.2). Furthermore, it is often important to use domain knowledge when constructing the representation, such as knowing that university degrees progress from Bachelor's to Master's to PhD or realizing that the postcode provided is not just a string of characters but actually encodes an area in London. In Table 8.2, we converted the data from Table 8.1 to a numerical format, and each postcode is represented as two numbers, a latitude and longitude. Even numerical data that could potentially be directly read into a machine learning algorithm should be carefully con-

Data is assumed to be in a tidy format (Wickham, 2014; Codd, 1990).

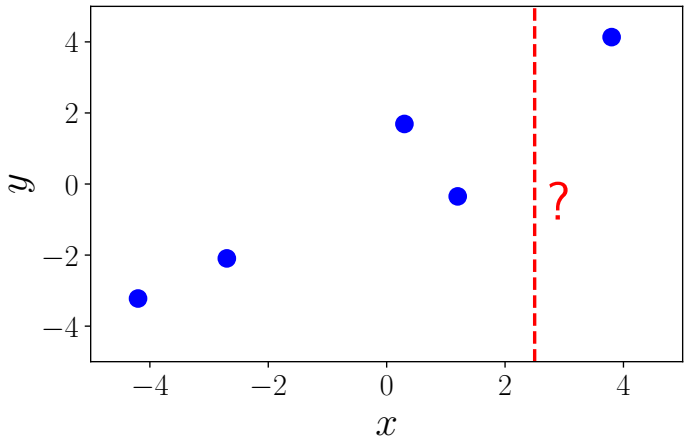


Figure 8.1 Toy data for linear regression. Training data in (x_n, y_n) pairs: $\{(-4.200, -3.222), (-2.700, -2.093), (+0.300, +1.690), (+1.200, -0.348), (+3.800, +4.134)\}$. We are interested in the value of the function at $x = 2.5$, which is not part of the training data.

sidered for units, scaling, and constraints. For the purposes of this book we assume that a domain expert already converted data appropriately, i.e., each input $\boldsymbol{x}_n$ is a D-dimensional vector of numbers, which are called *features*, *attributes* or *covariates*. In general, however, $\boldsymbol{x}_n$ could be a complex structured object (e.g., an image, a sentence, an email message, a time series, a molecular shape, a graph, etc).

Without additional information, one should shift and scale all columns of the dataset such that they mean 0 and variance 1.

features
attributes
covariates

In this part of the book, we will use N to denote the number of examples in a dataset and index the examples with lowercase $n = 1, \ldots, N$. We assume that we are given a set of numerical data, represented as an array of vectors, e.g., as illustrated in Figure 8.2. Each row is a particular individual $\boldsymbol{x}_n$ often referred to as an *example* or *data point* in machine learning. The subscript n refers to the fact that this is the n^{th} example out of a total of N examples in the dataset. Each column represents a particular *feature* of interest about the example, and we index the features as $d = 1, \ldots, D$. Recall that data is represented as vectors, which means that each example (each data point) is a D dimensional vector.

example
data point

feature

For supervised learning problems we have a label y_n associated with each example $\boldsymbol{x}_n$. A dataset is written as a set of example-label pairs $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n), \ldots, (\boldsymbol{x}_N, y_N)\}$. The table of examples $\{\boldsymbol{x}_1, \ldots \boldsymbol{x}_N\}$ is often concatenated, and written as $\boldsymbol{X} \in \mathbb{R}^{N \times D}$. Figure 8.1 illustrates an example of a one dimensional input x and corresponding labels y.

The orientation of the table originates from the database community, although it would actually be more convenient in machine learning for vectors representing examples to be columns.

Representing data as vectors $\boldsymbol{x}_n$ allows us to use concepts from linear algebra (introduced in Chapter 2). In many machine learning algorithms, we need to additionally be able to compare two vectors. As we will see in Chapters 9 and 12, computing the similarity or distance between two examples allows us to formalize the intuition that examples with similar features should have similar labels. The comparison of two vectors requires that we construct a geometry (explained in Chapter 3), and allows us to optimize the resulting learning problem using techniques in Chapter 7.

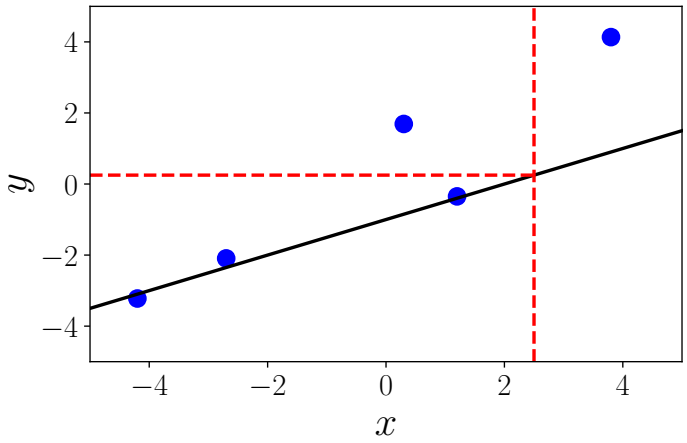


Figure 8.2 Example function (black solid diagonal line) and its prediction at $x = 2.5$. That is $f(2.5) = 0.25$.

Since we have vector representations of data, we can manipulate data to find potentially better representations of it. We will discuss finding good representations in two ways: finding lower-dimensional approximations of the original feature vector, and using nonlinear higher-dimensional combinations of the original feature vector. In Chapter 10 we will see an example of finding a low-dimensional approximation of the original data space by finding the principal components. Finding principal components is closely related to concepts of eigenvalue and singular value decomposition as introduced in Chapter 4. For the high-dimensional representation, we will see an explicit *feature map* $\phi(\cdot)$ that allows us to represent inputs $\boldsymbol{x}_n$ using a higher dimensional representation $\phi(\boldsymbol{x}_n)$. The main motivation for higher dimensional representations is that we can construct new features as non-linear combinations of the original features, which in turn may make the learning problem easier. We will discuss the feature map in Section 9.2 and show how this feature map leads to a *kernel* in Section 12.3.3. In recent years, deep learning methods (Goodfellow et al., 2016) have shown promise in using the data itself to learn the features, and has been very successful in areas such as computer vision, speech recognition and natural language processing. We will not cover neural networks in this part of the book, but the reader is referred to Section 5.6 for the mathematical description of backpropagation, a key concept for training neural networks.

feature map

kernel

Models are Functions

Once we have data in an appropriate vector representation, we can get to the business of constructing a predictive function (known as a *predictor*). In Chapter 1 we did not yet have the language to be precise about models. Using the concepts from the first part of the book, we can now introduce what ”model” means. We present two major approaches in this book: a

predictor

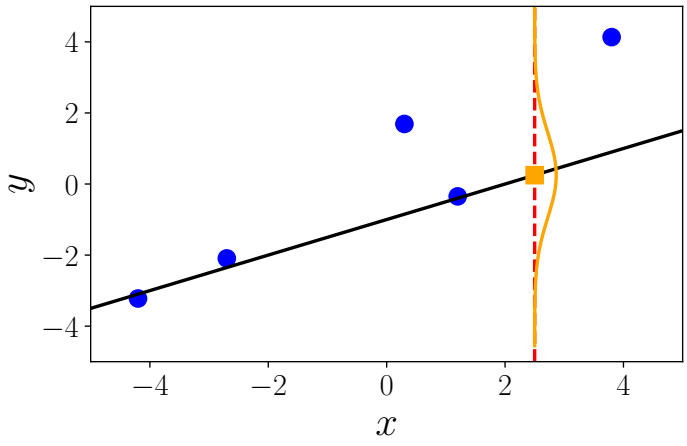


Figure 8.3 Example function (black solid diagonal line) and its predictive uncertainty at $x = 2.5$ (drawn as a Gaussian).

predictor as a function, and a predictor as a probabilistic model. We describe the former here and the latter in the next subsection.

A predictor is a function that, when given a particular input example (in our case a vector of features), produces an output. For now consider the output to be a single number, i.e., a real-valued scalar output. This can be written as

$$f : \mathbb{R}^D \to \mathbb{R}\,, \tag{8.1}$$

where the input vector $\boldsymbol{x}$ is D-dimensional (has D features), and the function f then applied to it (written as $f(\boldsymbol{x})$) returns a real number. Figure 8.2 illustrates a possible function that can be used to compute the value of the prediction for input values x.

In this book, we do not consider the general case of all functions, which would involve the need for functional analysis. Instead we consider the special case of linear functions

$$f(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x} + \theta_0\,. \tag{8.2}$$

This restriction means that the contents of Chapter 2 and 3 suffice for precisely stating the notion of a predictor for the non-probabilistic (in contrast to the probabilistic view described next) view of machine learning. Linear functions strike a good balance between the generality of the problems that can be solved and the amount of background mathematics that is needed.

Models are Probability Distributions

We often consider data to be noisy observations of some true underlying effect, and hope that by applying machine learning we can identify the signal from the noise. This requires us to have a language for quantifying the effect of noise. We often would also like to have predictors that express some sort of uncertainty, e.g., to quantify the confidence we have

about the value of the prediction for a particular test data point. As we have seen in Chapter 6 probability theory provides a language for quantifying uncertainty. Figure 8.3 illustrates the predictive uncertainty of the function as a Gaussian distribution.

Instead of considering a predictor as a single function, we could consider predictors to be probabilistic models, i.e., models describing the distribution of possible functions. We limit ourselves in this book to the special case of distributions with finite dimensional parameters, which allows us to describe probabilistic models without needing stochastic processes and random measures. For this special case we can think about probabilistic models as multivariate probability distributions, which already allow for a rich class of models.

We will introduce how to use concepts from probability (Chapter 6) to define machine learning models in Section 8.3, and introduce a graphical language for describing probabilistic models in a compact way in Section 8.4.

Learning is Finding Parameters

The goal of learning is to find a model and its corresponding parameters such that the resulting predictor will perform well on unseen data. There are conceptually three distinct algorithmic phases when discussing machine learning algorithms:

1. prediction or inference
2. training or parameter estimation
3. hyperparameter tuning or model selection

The prediction phase is when we use a trained predictor on previously unseen test data. In other words, the parameters and model choice is already fixed and the predictor is applied to new vectors representing new input data points. As outlined in Chapter 1 and the previous subsection, we will consider two schools of machine learning in this book, corresponding to whether the predictor is a function or a probabilistic model. When we have a probabilistic model (discussed further in Section 8.3) the prediction phase is called inference.

The training or parameter estimation phase is when we adjust our predictive model based on training data. We would like to find good predictors given training data, and there are two main strategies for doing so: finding the best predictor based on some measure of quality (sometimes called finding a point estimate), or using Bayesian inference. Finding a point estimate can be applied to both types of predictors, but Bayesian inference requires probabilistic models. For the non-probabilistic model, we follow the principle of *empirical risk minimization*, which we describe in Section 8.1. Empirical risk minimization directly provides an optimization problem for finding good parameters. With a statistical model the principle of *maximum likelihood* is used to find a good set of parameters (Sec-

empirical risk minimization

maximum likelihood

tion 8.2). We can additionally model the uncertainty of parameters using a probabilisitic model, which we will look at in more detail in Section 8.3.

We use numerical methods to find good parameters that "fit" the data, and most training methods can be thought of as hill climbing approaches to find the maximum of an objective, for example the maximum of a likelihood. To apply hill-climbing approaches we use the gradients described Chapter 5 and implement numerical optimization approaches from Chapter 7.

The convention in optimization is to minimize objectives. Hence, there is often an extra minus sign in machine learning objectives.

As mentioned in Chapter 1, we are interested in learning a model based on data such that it performs well on future data. It is not enough for the model to only fit the training data well, the predictor needs to perform well on unseen data. We simulate the behaviour of our predictor on future unseen data using cross validation (Section 8.1.4). As we will see in this chapter, to achieve the goal of performing well on unseen data, we will need to balance between fitting well on training data and finding "simple" explanations of the phenomenon. This trade off is achieved using regularization (Section 8.1.3) or by adding a prior (Section 8.2.2). In philosophy, this is considered to be neither induction or deduction, and is called *abduction*. According to the Stanford Encyclopedia of Philosophy, abduction is the process of inference to the best explanation (Douven, 2017).

abduction

A good movie title is "AI abduction".

We often need to make high level modeling decisions about the structure of the predictor, such as the number of components to use or the class of probability distributions to consider. The choice of the number of components is an example of a *hyperparameter*, and this choice can affect the performance of the model significantly. The problem of choosing between different models is called *model selection*, which we describe in Section 8.5. For non-probabilistic models, model selection is often done using *cross validation*, which is described in Section 8.1.4. We also use model selection to choose hyperparameters of our model.

hyperparameter

model selection

cross validation

Remark. The distinction between parameters and hyperparameters is somewhat arbitrary, and is mostly driven by the distinction between what can be numerically optimized versus what needs to utilize search techniques. Another way to consider the distinction is to consider parameters as the explicit parameters of a probabilistic model, and to consider hyperparameters (higher level parameters) as parameters that control the distribution of these explicit parameters. ♢

8.1 Empirical Risk Minimization

After having all the mathematics under our belt, we are now in a position to introduce what it means to learn. The "learning" part of machine learning boils down to estimating parameters based on training data.

In this section we consider the case of a predictor that is a function,

and consider the case of probabilistic models in Section 8.2. We describe the idea of empirical risk minimization, which was originally popularized by the proposal of the support vector machine (described in Chapter 12). However, its general principles are widely applicable and allows us to ask the question of what is learning without explicitly constructing probabilistic models. There are four main design choices, which we will cover in detail in the following subsections:

Section 8.1.1 What is the set of functions we allow the predictor to take?

Section 8.1.2 How do we measure how well the predictor performs on the training data?

Section 8.1.3 How do we construct predictors from only training data that performs well on unseen test data?

Section 8.1.4 What is the procedure for searching over the space of models?

8.1.1 Hypothesis Class of Functions

Assume we are given N examples $\boldsymbol{x}_n \in \mathbb{R}^D$ and corresponding scalar labels $y_n \in \mathbb{R}$. We consider the supervised learning setting, where we obtain pairs $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)$. Given this data, we would like to estimate a predictor $f(\cdot, \boldsymbol{\theta}) : \mathbb{R}^D \to \mathbb{R}$, parameterized by $\boldsymbol{\theta}$. We hope to be able to find a good parameter $\boldsymbol{\theta}^*$ such that we fit the data well

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}^*) \approx y_n \quad \text{for all} \quad n = 1, \dots, N\,. \tag{8.3}$$

In this section, we use the notation $\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta}^*)$ to represent the output of the predictor.

Example 8.1
We introduce the problem of least squares regression to illustrate empirical risk minimization. A more comprehensive account of regression is given in Chapter 9. When the label y_n is real valued, a popular choice of function class for predictors is the set of linear functions,

$$f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{x}_n + \theta_0\,. \tag{8.4}$$

Observe that the predictor takes the vector of features representing a single example $\boldsymbol{x}_n$ as input and produces a real valued output. That is $f : \mathbb{R}^D \to \mathbb{R}$. The previous figures in this chapter had a straight line as a predictor, which means that we have assumed a linear function. For notational convenience we often concatenate an additional unit feature to $\boldsymbol{x}_n$, that is $\tilde{\boldsymbol{x}}_n = \begin{bmatrix} \boldsymbol{x}_n \\ 1 \end{bmatrix}$. This is so that we can correspondingly concatenate the

parameter vector $\tilde{\boldsymbol{\theta}} = \begin{bmatrix} \boldsymbol{\theta} \\ \theta_0 \end{bmatrix}$, and write the linear predictor as

$$f(\tilde{\boldsymbol{x}}_n, \tilde{\boldsymbol{\theta}}) = \tilde{\boldsymbol{\theta}}^\top \tilde{\boldsymbol{x}}_n \,. \tag{8.5}$$

We will often overload the notation in this book to have tidier presentation: $\boldsymbol{x}_n$ is used to mean the new concatenated vector.

Instead of a linear function, we may wish to consider non-linear functions as predictors. Recent advances in neural network frameworks allowed for efficient computation of more complex non-linear function classes.

Remark. For ease of presentation we will describe empirical risk minimization in terms of supervised learning. This simplifies the definition of the hypothesis class and the loss function. ♢

Given the class of functions we want to search for a good predictor, we now move on to the second ingredient of empirical risk minimization: how to measure how well the predictor fits the training data.

8.1.2 Loss Function for Training

Consider the label y_n for particular example; and the corresponding prediction $\hat{y}_n$ that we make based on $\boldsymbol{x}_n$. To define what it means to fit the data well, we need to specify a *loss function* $\ell(y_n, \hat{y}_n)$ that takes two values as input and produces a non-negative number (referred to as the loss) representing how much error we have made on this particular prediction. Our goal for finding a good parameter vector $\boldsymbol{\theta}^*$ is to minimize the average loss on the set of N training examples.

loss function

The word error is often used to mean loss.

One assumption that is commonly made in machine learning is that the set of examples $(x_1, y_1), \ldots, (x_N, y_N)$ are *independent and identically distributed*. The word independent (Section 6.4.3) means that two data points (x_i, y_i) and (x_j, y_j) do not statistically depend on each other, meaning that the empirical mean is a good estimate of the population mean (Section 6.4.1). This implies that we can use the empirical mean of the loss on the training data. For a given *training set* $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ which we collect into an example matrix $\boldsymbol{X}$ and label vector $\boldsymbol{y}$, the average loss is given by

independent and identically distributed

training set

$$\mathbf{R}_{\text{emp}}(f, \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n) \tag{8.6}$$

where $\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta}^*)$. Equation (8.6) is called the *empirical risk*. Note that the empirical risk depends on three arguments, the predictor f and the data $\boldsymbol{X}, \boldsymbol{y}$. This general strategy for learning is called *empirical risk minimization*.

empirical risk

empirical risk minimization

Example 8.2
Continuing the example of least squares regression, we specify that we measure cost of making an error during training using the squared loss $\ell(y_n, \hat{y}_n) = (y_n - \hat{y}_n)^2$. We wish to minimize the empirical risk, which is the average of the losses over the data

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} (y_n - f(\boldsymbol{x}_n, \boldsymbol{\theta}))^2, \tag{8.7}$$

where we have substituted the predictor $\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta})$. By using our choice of a linear predictor $f(\boldsymbol{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{x}_n$ we obtain the optimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} (y_n - \boldsymbol{\theta}^\top \boldsymbol{x}_n)^2 \,. \tag{8.8}$$

This equation can be equivalently expressed in matrix form by collecting the labels into a vector $\boldsymbol{y} := [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$ and collecting the dataset into a matrix $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$.

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2 \,. \tag{8.9}$$

This is known as the least squares problem. There is a closed-form analytic solution for this, by solving the normal equations, which we will discuss in Section 9.2.

Note that we are not interested in a predictor that only performs well on the training data. We are actually interested in a predictor that performs well (has low risk) on unseen test data. More formally we are interested in finding a predictor f (with parameters fixed) that minimizes *expected risk*

expected risk

$$\mathbf{R}_{\text{true}}(f) = \mathbb{E}_{\boldsymbol{x},y} \ell(y, f(\boldsymbol{x})) \tag{8.10}$$

where y is the ground truth label, and $f(\boldsymbol{x})$ is the prediction based on the data $\boldsymbol{x}$. The notation $\mathbf{R}_{\text{true}}(f)$ indicates that this is the true risk if we had access to an infinite amount of data. The expectation is over the (infinite) set of all possible data and labels. There are two practical questions that arise from our desire to minimize expected risk which we address in the following two subsections:

Another phrase commonly used for expected risk is the population risk.

- How should we change our training procedure to generalize well?
- How do we estimate expected risk from (finite) data?

Remark. Many machine learning tasks are specified with an associated performance measure, e.g., accuracy of prediction or root mean squared error. The performance measure could be more complex, be cost sensitive

and capture details about the particular application. In principle, the design of the loss function for empirical risk minimization should correspond directly to the performance measure specified by the machine learning task. In practice there is a often mismatch between the design of the loss function and the performance measure. This could be due to issues such as ease of implementation or efficiency of optimization. ♢

8.1.3 Regularization to Reduce Overfitting

This section describes an addition to empirical risk minimization that allows it to generalize well (minimizing expected risk). Recall that the aim of training a machine learning predictor is so that we can perform well on unseen data, that is the predictor generalizes well. This unseen data is referred to as the *test set*. Given a sufficiently rich class of functions for the predictor f, we can essentially memorize the training data to obtain zero empirical risk. While this is great to minimize the loss (and therefore the risk) on the training data, we would not expect the predictor to generalize well to unseen data. In practice we have only a finite set of data, and hence we split our data into a training and a test set. The training set is used to fit the model, and the test set (not seen by the machine learning algorithm during training) is used to evaluate generalization performance. We use the subscript $_{\text{train}}$ and $_{\text{test}}$ to denote the training and test set respectively. We will revisit this idea of using a finite dataset to evaluate expected risk in Section 8.1.4.

test set

It turns out that empirical risk minimization can lead to *overfitting*, that is the predictor fits too closely to the training data and does not generalize well to new data (Mitchell, 1997). This general phenomenon of having very small training loss but large test loss tends to occur when we have little data and a complex hypothesis class. For a particular predictor f (with parameters fixed), the phenomenon of overfitting occurs when the risk estimate from the training data $\mathbf{R}_{\text{emp}}(f, \boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ underestimates the expected risk $\mathbf{R}_{\text{true}}(f)$. Since we estimate the expected risk $\mathbf{R}_{\text{true}}(f)$ by using the empirical risk on the test set $\mathbf{R}_{\text{emp}}(f, \boldsymbol{X}_{\text{test}}, \boldsymbol{y}_{\text{test}})$ if the test risk is much larger than the training risk, this is an indication of overfitting.

overfitting

Therefore, we need to somehow bias the search for the minimizer of empirical risk by introducing a penalty term, which makes it harder for the optimizer to return an overly flexible predictor. In machine learning, the penalty term is referred to as *regularization*. Regularization is a way to compromise between accurate solution of empirical risk and the size or complexity of the solution.

regularization

Example 8.3
Regularization is used to improve the conditioning of ill-conditioned least squares problems. The simplest regularization strategy is to replace the

least squares problem in the previous example

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2 \ . \tag{8.11}$$

with the "regularized" problem by adding a penalty term involving only $\boldsymbol{\theta}$

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \ . \tag{8.12}$$

The additional term $\|\boldsymbol{\theta}\|^2$ is known as a regularizer, and the parameter λ is known as the regularization parameter. The regularization parameter trades off minimizing the loss on the training set and the size of the parameters $\boldsymbol{\theta}$.

The constant $\frac{1}{2}$ in front of the regularizer is so that when we take the derivative, the square and the half cancels.

The regularization term is sometimes called the penalty term, what biases the vector $\boldsymbol{\theta}$ to be closer to the origin. The idea of regularization also appears in probabilisitic models as the prior probability of the parameters. Recall from Section 6.7 that for the posterior distribution to be of the same form as the prior, the prior distribution and the likelihood need to be conjugate distributions. We will revisit this idea in Section 8.2.2. We will see in Chapter 12 that the idea of the regularizer is equivalent to the idea of a large margin.

8.1.4 Cross Validation to Assess the Generalization Performance

We mentioned in the previous section that we measure generalization error by estimating it by applying the predictor on test data. This data is also sometimes referred to as the *validation set*. The validation set is a subset of the available training data that we keep aside. A practical issue with this approach is that the amount of data is limited, and ideally we would use as much of the data available to train the model. This would require to keep our validation set $\mathcal{V}$ small, which then would lead to a noisy estimate (with high variance) of the predictive performance. One solution to these contradictory objectives (large training set, large validation set) is to use *cross validation*. K-fold cross validation effectively partitions the data into K chunks, $K-1$ of which form the training set $\tilde{\mathcal{D}}$, and the last chunk serves as the validation set $\mathcal{V}$ (similar to the idea outlined above). Cross-validation iterates through (ideally) all combinations of assignments of chunks to $\tilde{\mathcal{D}}$ and $\mathcal{V}$, see Figure 8.4. This procedure is repeated for all K choices for the validation set, and the performance of the model from the K runs is averaged.

validation set

cross validation

We partition our training set into two sets $\mathcal{D} = \tilde{\mathcal{D}} \cup \mathcal{V}$, such that they do not overlap $\tilde{\mathcal{D}} \cap \mathcal{V} = \emptyset$, where $\mathcal{V}$ is the validation set, and train our model on $\tilde{\mathcal{D}}$. After training, we assess the performance of the predictor f on the validation set $\mathcal{V}$ (e.g., by computing root mean square error (RMSE)

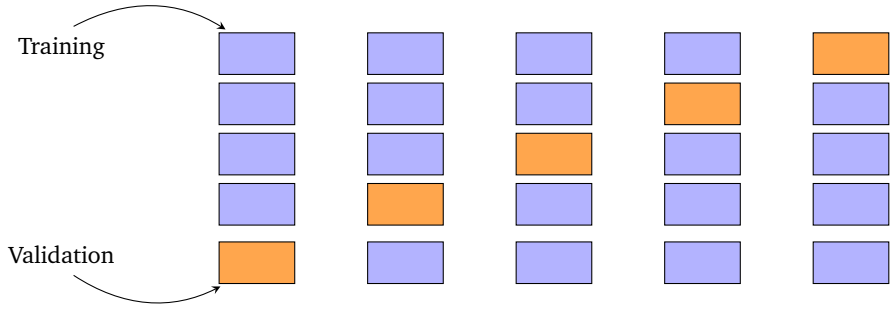


Figure 8.4 K-fold cross validation. The data set is divided into $K = 5$ chunks, $K - 1$ of which serve as the training set (blue) and one as the validation set (orange).

of the trained model on the validation set). We cycle through all possible partitionings of validation and training sets and compute the average generalization error of the predictor. Cross-validation effectively computes the expected generalization error

$$\mathbb{E}_{\mathcal{V}}[\mathbf{R}(f, \mathcal{V})] \approx \frac{1}{K} \sum_{k=1}^{K} \mathbf{R}(f, \mathcal{V}^{(k)}) \,, \tag{8.13}$$

where $\mathbf{R}(f, \mathcal{V})$ is the risk (e.g., RMSE) on the validation set $\mathcal{V}$ for predictor f.

A potential disadvantage of K-fold cross validation is the computational cost of training the model K times, which can be burdensome if the training cost is computationally expensive. In practice, it is often not sufficient to look at the direct parameters alone. For example, we need to explore multiple complexity parameters (e.g., multiple regularization parameters), which may not be direct parameters of the model. Evaluating the quality of the model, depending on these hyper-parameters may result in a number of training runs that is exponential in the number of model parameters.

However, cross validation is an *embarrassingly parallel* problem, i.e., little effort is needed to separate the problem into a number of parallel tasks. Given sufficient computing resources (e.g., cloud computing, server farms), cross validation does not require longer than a single performance assessment.

embarrassingly parallel

Further Reading

Due to the fact that the original development of empirical risk minimization (Vapnik, 1998) was couched in heavily theoretical language, many of the subsequent developments have been theoretical. The area of study is called *statistical learning theory* (von Luxburg and Schölkopf, 2011; Vapnik, 1999; Evgeniou et al., 2000). A recent machine learning textbook that builds on the theoretical foundations and develops efficient learning algorithms is Shalev-Shwartz and Ben-David (2014).

statistical learning theory

The idea of regularization has its roots in the solution of ill-posed in-

verse problems (Neumaier, 1998). It has deep relationships to the bias variance tradeoff and feature selection (Bühlmann and Geer, 2011).

An alternative to cross validation is bootstrap and jackknife (Efron and Tibshirani, 1993; Davidson and Hinkley, 1997; Hall, 1992).

8.2 Parameter Estimation

In Section 8.1 we did not explicitly model our problem using probability distributions. In this section, we will see how to use probability distributions to model our uncertainty due to the observation process and our uncertainty in the parameters of our predictors.

8.2.1 Maximum Likelihood Estimation

maximum likelihood estimation

likelihood

The idea behind *maximum likelihood estimation* (MLE) is to define a function of the parameters that enables us to find a model that fits the data well. The estimation problem is focused on the *likelihood* function, or more precisely its negative logarithm. For data represented by random variable $\boldsymbol{x}$ and for a family of probability densities $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, the *negative log likelihood* is given by

negative log likelihood

$$\mathcal{L}_{\boldsymbol{x}}(\boldsymbol{\theta}) = -\log p(\boldsymbol{x} \mid \boldsymbol{\theta}). \tag{8.14}$$

The notation $\mathcal{L}_{\boldsymbol{x}}(\boldsymbol{\theta})$ emphasizes the fact that the parameter $\boldsymbol{\theta}$ is varying and the data $\boldsymbol{x}$ is fixed. We very often drop the reference to $\boldsymbol{x}$ when writing the negative log likelihood, as it is really a function of $\boldsymbol{\theta}$, and write it as $\mathcal{L}(\boldsymbol{\theta})$ when the random variable representing the uncertainty in the data is clear from the context.

Let us intepret what the probability density $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is modelling for a fixed value of $\boldsymbol{\theta}$. It is a distribution that models the uncertainty of the data. In other words, once we have chosen the type of function we want as a predictor, the likelihood provides the probability of observing data $\boldsymbol{x}$.

In a complementary view, if we consider the data to be fixed (because it has been observed), and we vary the parameters $\boldsymbol{\theta}$, what does $\mathcal{L}(\boldsymbol{\theta})$ tell us? It tells us the (negative log) likelihood of that parameter setting. Based on this second view, the maximum likelihood estimator is the parameter setting that maximizes the function.

We consider the supervised learning setting, where we obtain pairs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and labels $y_n \in \mathbb{R}$. We are interested in constructing a predictor that takes a feature vector $\boldsymbol{x}_n$ as input and produces a prediction y_n (or something close to it). That is given a vector $\boldsymbol{x}_n$ we want the probability distribution of the label y_n. In other words we specify the conditional probability distribution of the labels given the examples for the particular parameter setting $\boldsymbol{\theta}$.

Example 8.4
The first example that is often used is to specify that the conditional probability of the labels given the examples is a Gaussian distribution. In other words we assume that we can explain our observation uncertainty by independent Gaussian noise (refer to Section 6.6) with zero mean, $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$. We further assume that the linear model $\boldsymbol{x}_n^\top \boldsymbol{\theta}$ is used for prediction. This means we specify a Gaussian likelihood for each example label pair $\boldsymbol{x}_n, y_n$,

$$p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = \mathcal{N}(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2) . \tag{8.15}$$

An illustration of a Gaussian likelihood for a given parameter $\boldsymbol{\theta}$ is shown in Figure 8.3. We will see in Section 9.2 how to explicitly expand the expression above out in terms of the Gaussian distribution.

We assume that the set of examples $(x_1, y_1), \ldots, (x_N, y_N)$ are *independent and identically distributed*. The word independent (Section 6.4.3) implies that the likelihood of the whole dataset ($\boldsymbol{y} = [y_1, \ldots, y_N]^\top$ and $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$) factorizes into a product of the likelihoods of each individual example

independent and identically distributed

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) , \tag{8.16}$$

where $p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})$ is a particular distribution (which was Gaussian in the example above (8.15)). The word identically distributed means that each term in the product above is the same and all of them share the same parameters. It is often easier from an optimization viewpoint to compute functions that can be decomposed into sums of simpler functions, and hence in machine learning we often consider the negative log-likelihood

Recall $\log(ab) = \log(a) + \log(b)$

$$\mathcal{L}(\boldsymbol{\theta}) = -\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}). \tag{8.17}$$

While it is temping to interpret the fact that $\boldsymbol{\theta}$ is on the right of the conditioning in $p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$ (8.15), and hence should be intepreted as observed and fixed, this interpretation is incorrect. The negative log likelihood $\mathcal{L}(\boldsymbol{\theta})$ is a function of $\boldsymbol{\theta}$.

Therefore, to find a good parameter vector $\boldsymbol{\theta}$ that explains the data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ well, we look for a $\boldsymbol{\theta}$ that minimizes the negative log likelihood

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) . \tag{8.18}$$

Remark. The negative sign in (8.17) is a historical artefact that is due to the convention that we want to maximize likelihood, but numerical optimization literature tends to study minimization of functions. ♢

Example 8.5
Continuing on our example of Gaussian likelihoods (8.15), the negative log likelihood can be rewritten as

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) \tag{8.19}$$

$$= -\sum_{n=1}^{N} \log \mathcal{N}\left(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta},\, \sigma^2\right) \tag{8.20}$$

$$= -\sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2}\right) \tag{8.21}$$

$$= -\sum_{n=1}^{N} \log \exp\left(-\frac{(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2}\right) - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \tag{8.22}$$

$$= \sum_{n=1}^{N} \frac{(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2} - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}}. \tag{8.23}$$

$$\tag{8.24}$$

Observe that the first term in the last equation above is the least squares problem.

It turns out that for Gaussian likelihoods the resulting optimization problem corresponding to maximum likelihood estimation has a closed-form solution. We will see more details on this in Chapter 9. For other likelihood functions, i.e., if we model our noise with non-Gaussian distributions, maximum likelihood estimation may not have a closed-form analytic solution. In this case, we resort to numerical optimization methods discussed in Chapter 7.

8.2.2 Maximum A Posteriori Estimation

If we have prior knowledge about the distribution of the parameters $\boldsymbol{\theta}$ of our distribution we can multiply an additional term to the likelihood. This additional term is a prior probability distribution on parameters $p(\boldsymbol{\theta})$. For a given prior, after observing some data $\boldsymbol{x}$, how should we update the distribution of $\boldsymbol{\theta}$? In other words, how should we represent the fact that we have more specific knowledge after observing data $\boldsymbol{x}$? Bayes' theorem, as discussed in Section 6.3, gives us a principled tool to update our probability distributions of random variables. It allows us to compute a *posterior* distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ (the more specific knowledge) on the paramters $\boldsymbol{\theta}$ from general *prior* statements (prior distribution) $p(\boldsymbol{\theta})$ and the function $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ that links the parameters $\boldsymbol{\theta}$ and the observed data $\boldsymbol{x}$ (called the *likelihood*):

posterior

prior

likelihood

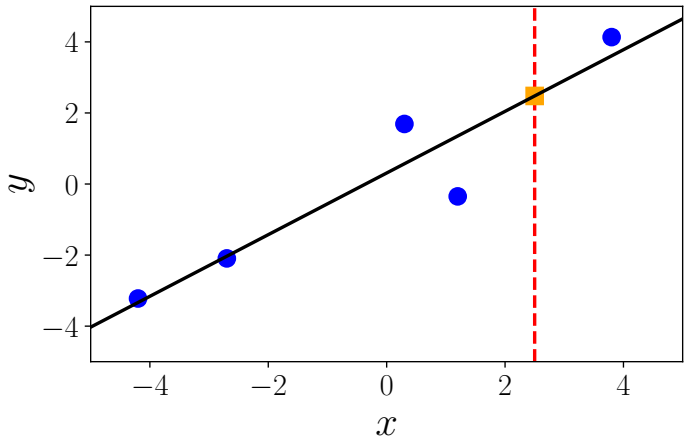


Figure 8.5 For the given data, the maximum likelihood estimate of the parameters results in the black diagonal line. The orange square shows the value of the maximum likelihood prediction at $x = 2.5$.

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} . \tag{8.25}$$

Recall that we are interested in finding the parameter $\boldsymbol{\theta}$ that maximizes likelihood, and the distribution $p(\boldsymbol{x})$ affects the value of the likelihood, but does not affect the value of the parameter that achieves the maximum likelihood. Therefore we can ignore the value of the denominator,

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta}) . \tag{8.26}$$

The proportion relation above hides the density of the data $p(\boldsymbol{x})$ which may be difficult to estimate. Instead of estimating the minimum of the negative log likelihood, we now estimate the minimum of the negative log posterior, which is referred to as *maximum a posteriori estimation* (MAP).

maximum a posteriori estimation

Example 8.6
In addition to the assumption of Gaussian likelihood in the previous example, we assume that the parameter vector is distributed as a multivariate Gaussian with zero mean, that is $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma}$ is the covariance matrix (Section 6.6). Note that the conjugate prior of a Gaussian is also a Gaussian (Section 6.7.1) and therefore we expect the posterior distribution to also be a Gaussian. We will see the details of maximum a posteriori estimation in Chapter 9.

The idea of including prior knowledge about where good parameters lie is widespread in machine learning. An alternative view which we saw in Section 8.1 is the idea of regularization, which introduces an additional term that biases the resulting parameters to be close to the origin.

Remark. The maximum likelihood estimate $\boldsymbol{\theta}_{\text{ML}}$ possesses the following properties (Lehmann and Casella, 1998; Efron and Hastie, 2016):

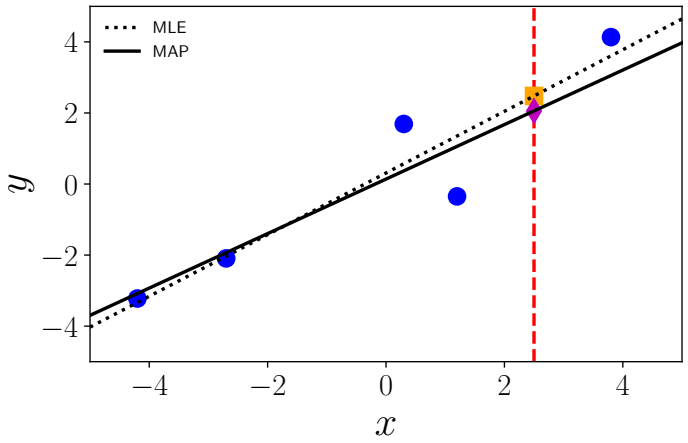


Figure 8.6 Comparing the Maximum Likelihood estimate and the Maximum A Posteriori estimate and their predictions at $x = 2.5$. The prior biases the slope to be less steep and the intercept to be closer to zero.

- Asymptotic consistency: The MLE converges to the true value in the limit of infinitely many observations, plus a random error that is approximately normal.
- The size of the samples necessary to achieve these properties can be quite large.
- The error's variance decays in $1/N$ where N is the number of data points.
- Especially, in the "small" data regime, maximum likelihood estimation can lead to *overfitting*.

♢

Further Reading

When considering probabilistic models the principle of maximum likelihood estimation generalizes the idea of least-squares regression for linear models (which we will discuss in detail in Chapter 9). When restricting the predictor to have linear form with an additional nonlinear function φ applied to the output,

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \varphi(\boldsymbol{\theta}^\top \boldsymbol{x}_n) \tag{8.27}$$

we can consider other models for other prediction tasks, such as binary classification or modelling count data (McCullagh and Nelder, 1989). An alternative view of this is to consider likelihoods that are from the exponential family (Section 6.7). The class of models, which have linear dependence between parameters and data, and have potentially nonlinear transformation φ (called a link function) is referred to as generalized linear models (Agresti, 2002, Chapter 4).

Maximum likelihood estimation has a rich history, and was originally proposed by Sir Ronald Fisher in the 1930s. We will expand upon the idea of a probabilistic model in Section 8.3. One debate among researchers

who use probabilistic models, is the discussion between Bayesian and frequentist statistics. As mentioned in Section 6.1.1 it boils down to the definition of probability. Recall that one can consider probability to be a generalization of logical reasoning to allow for uncertainty (Cheeseman, 1985; Jaynes, 2003). The method of maximum likelihood estimation is frequentist in nature, and the interested reader is pointed to Efron and Hastie (2016) for a balanced view of both Bayesian and frequentist statistics.

There are some probabilisitic models where maximum likelihood estimation may not be possible. The reader is referred to more advanced statistical textbooks, e.g., Casella and Berger (2002), for approaches such as method of moments, M-estimation and estimating equations.

8.3 Probabilistic Modeling

In machine learning, we are frequently concerned with the interpretation and analysis of data, e.g., for prediction of future events and decision making. To make this task more tractable, we often build models that describe the process that generates the data. For example, when we want to describe the outcome of a coin-flip experiment, we can describe this process using a Bernoulli distribution as described in Chapter 6. In this example, we can say that an outcome $x \in \{\text{head, tail}\}$ can be described as the conditional distribution $p(x \mid \mu)$ where x is the outcome of the experiment and μ is the probability of "heads".

In this section, we will focus on probabilisitic models. The benefit of using probabilistic models is that we have the set of tools from probability (Chapter 6) available to us for modeling, inference, parameter estimation and model selection.

Remark. Thinking about empirical risk minimization (Section 8.1) as "probability free" is incorrect. There is an underlying unknown probability distribution $p(\boldsymbol{x}, y)$ that governs the data generation, but the approach of empirical risk minimization is agnostic to that choice of distribution. This is in contrast to standard statistical approaches that require the knowledge of $p(\boldsymbol{x}, y)$. Furthermore, since the distribution is a joint distribution on both examples $\boldsymbol{x}$ and labels y, the labels can be non-deterministic. In contrast to standard statistics we do not need to specify the noise distribution for the labels y. ♢

8.3.1 MLE, MAP, and Bayesian Inference

Let us revisit the discussion about modeling with probability distributions we had at the beginning of this chapter. There are three levels where we can use a probability distribution. At the first level, we can use a probability distribution to model our uncertainty about the observation. For example, in (8.15) we make the assumption that there is Gaussian noise

(with mean 0 and variance σ^2) that corrupts the observation of a linear function. A way to express this is to write

$$y_n = \boldsymbol{x}_n^\top \boldsymbol{\theta} + \varepsilon \quad \text{where} \quad \varepsilon \sim \mathcal{N}\left(0, \sigma^2\right). \tag{8.28}$$

By making this assumption, we obtain the likelihood described in Section 8.2.1 where we can then maximize.

At the second level, we can use a probability distribution to describe our uncertainty about the parameters $\boldsymbol{\theta}$. This is detailed in Section 8.2.2, where we place a probability distribution to model the parameter vector $\boldsymbol{\theta}$ that encodes our beliefs about the unknown parameters. The probability distribution over parameters is known as the prior distribution, and by using Bayes' Theorem we obtain the posterior distribution over the parameters $\boldsymbol{\theta}$, which describes an "updated" prior belief, i.e., the belief about the unknown parameters *after* having seen some data. Instead of maximizing the likelihood, we can maximize the posterior with respect to the model parameters $\boldsymbol{\theta}$. This approach is called *maximum a posteriori estimation* (*MAP estimation*), and it will generally yield a different result than maximum likelihood estimation. Note that in both maximum likelihood and maximum a posteriori cases in the previous paragraphs, the estimated best solution is a single value of the parameter $\boldsymbol{\theta}$.

maximum a posteriori estimation

MAP estimation

With maximum likelihood or MAP estimates, we obtain a single best parameter setting $\boldsymbol{\theta}^*$, which we can use when making predictions. More specifically, when predicting an outcome $\boldsymbol{x}_*$ we can do this by using $\boldsymbol{\theta}^*$ directly in the likelihood function that connects parameters and data to obtain a prediction $p(\boldsymbol{x}_* \mid \boldsymbol{\theta}^*)$. At the third level, we can use a probability distribution when making predictions, instead of focusing on a single parameter setting $\boldsymbol{\theta}^*$. To do so, we maintain a full probability distribution on the parameters (MLE and MAP estimation pick a single parameter value) and make predictions by accounting for all plausible parameter settings $\boldsymbol{\theta}$ under this distribution. This is done by (weighted) averaging, i.e., integration so that the predictive distribution

$$p(\boldsymbol{x}_*) = \mathbb{E}_{\boldsymbol{\theta}}[p(\boldsymbol{x}_* \mid \boldsymbol{\theta})] = \int p(\boldsymbol{x}_* \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \tag{8.29}$$

no longer depends on the parameters $\boldsymbol{\theta}$ – they have been marginalized/ integrated out. This is referred to as *Bayesian inference*.

Bayesian inference

8.3.2 Latent Variables

Taking a probabilistic view of machine learning implies that we also want to treat predictors (models) as random variables. While data $x_1, \ldots, x_N$ can be observed, these models themselves possess quantities/parameters that are not directly observed. In the coin-flip experiment described in the introduction to this section, the probability μ of "heads" is typically not

known and depends on the coin we use. We describe these kind of unknown quantities with a random variable. Given that these random variables cannot be observed but only inferred we call them *hidden variables* or *latent variables*. While we can make general statements about plausible values of these latent variables prior to observing any data (e.g., by using a prior distribution), a key challenge in machine learning is to infer more about these unobserved variables given a dataset.

hidden variables

latent variables

Remark. We tend to use the notation $\boldsymbol{\theta}$ to represent the vector of unobserved random variables. We also refer to $\boldsymbol{\theta}$ as the *parameters* of the model. ♢

parameters

Recall from Section 8.2.2 that Bayes' Theorem gives us a principled tool to update our probability distribution over $\boldsymbol{\theta}$ given observed data. Here we go from a prior distribution on the latent variables to a posterior distribution after taking the likelihood into account. Since we can write the numerator of (8.25) as the joint distribution $p(\boldsymbol{\theta}, \boldsymbol{x}) = p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$ of the latent variables and the data, this joint distribution is of central importance and sufficient to compute the quantities of interest, either by conditioning or marginalization (Section 6.2.1). For example, we obtain the posterior distribution by conditioning so that we can make informed statements about the latent variables, and we obtain the marginal likelihood (evidence) $p(\boldsymbol{x})$ via marginalization where we integrate out the latent variables. The marginal likelihood is very useful for model selection as we will discuss in Section 8.5. Hence, we can define a probabilistic model in machine learning as the joint distribution $p(\boldsymbol{\theta}, \boldsymbol{x})$ of all latent and observed variables.

A probabilistic model in machine learning describes the joint distribution of all latent and observed variables.

As mentioned above, in machine learning, it is important to get some information about the latent variables given a dataset. The posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ gives us complete information about the parameters after observing the data. We can then use the full posterior distribution to make statements about future outcomes by averaging over all plausible settings of the latent variables, that is we can predict/generate/fantasize/hallucinate new data via

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}\,. \tag{8.30}$$

Unfortunately, in most cases (in particular, when we choose non-conjugate priors), we cannot compute the posterior distribution using Bayes' theorem because the computations quickly become intractable. A solution to this problem is to estimate a single parameter vector $\boldsymbol{\theta}^*$ that explains the available data "best", e.g., by maximum likelihood estimation as discussed in Section 8.2. Then, we can make a prediction of new data directly via the likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta}^*)$ – without needing integration.

Remark. In the machine learning literature, there can be a somewhat arbitrary separation between "variables" and "parameters". While parame-

ters are estimated (e.g., via maximum likelihood) variables are usually marginalized out as in (8.30). In this book, we are not so strict with this separation because, in principle, we can place a prior on any parameter and integrate it out, which would then turn the parameter into a variable according to the separation above. ♢

There are modelling situations in machine learning where we may wish to introduce new random variables $\boldsymbol{z}$ into the problem. In these scenarios, the direct model of the problem (involving only $\boldsymbol{x}, \boldsymbol{\theta}$) may be computationally difficult to solve, but introducing a set of latent variables $\boldsymbol{z}$ allows us to design an efficient algorithm. We will see an example of this in Chapter 11, where we introduce the Gaussian mixture model and use it for density estimation.

Further Reading

Probabilistic models in machine learning Bishop (2006); Barber (2012); Murphy (2012) provide a way for users to capture uncertainty about data and predictive models in a principled fashion. Ghahramani (2015) presents a short review of probabilistic models in machine learning. Given a probabilistic model, we may be lucky enough to be able to compute parameters of interest analytically. However, in general, analytic solutions are rare and computational methods such as sampling (Brooks et al., 2011) and variational inference (Blei et al., 2017) are used.

In recent years, there have been several proposed programming languages that aim to treat the variables defined in software as random variables corresponding to probability distributions. The long-term dream is to be able to write complex functions of probability distributions, while under the hood the compiler automatically takes care of the rules of Bayesian inference. This is a rapidly changing field, but several examples of promising languages at the present are:

Stan `http://mc-stan.org/`
Edward `http://edwardlib.org/`
PyMC `https://docs.pymc.io/`
Pyro `http://pyro.ai/`
Tensorflow Probability `https://github.com/tensorflow/probability`
Infer.NET `http://infernet.azurewebsites.net/`

8.4 Directed Graphical Models

In this section we introduce a graphical language for specifying a probabilistic models, called the *directed graphical model*. They provides a compact and succinct way to specify probabilistic models, and allows the reader to visually parse dependencies between random variables. A graphical model visually captures the way in which the joint distribution over

directed graphical model

Directed graphical models are also known as Bayesian networks.

all random variables can be decomposed into a product of factors depending only on a subset of these variables. In Section 8.3, we identified the joint distribution of a probabilistic model as the key quantity of interest because it comprises information about the prior, the likelihood and the posterior. However, the joint distribution by itself can be quite complicated, and it does not tell us anything about structural properties of the probabilistic model. For example, the joint distribution $p(a, b, c)$ does not tell us anything about independence relations. This is the point where graphical models come into play. This section relies on the concepts of independence and conditional independence, as described in Section 6.4.3.

In a *graphical model*, nodes are random variables; in Figure 8.7(a), the nodes of the random variables a, b, c represent their respective (marginal) probabilities $p(a)$, $p(b)$ and $p(c)$. Edges represent probabilistic relations between variables, e.g., conditional probabilities.

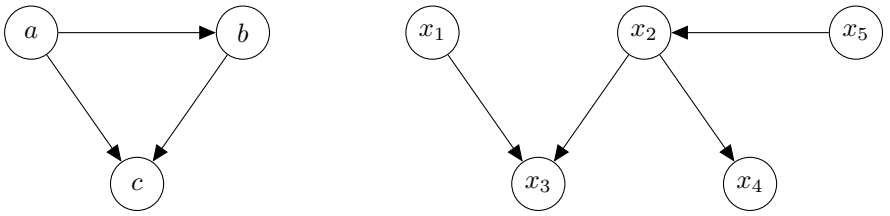
graphical model

Remark. Not every distribution can be represented in a particular choice of graphical model. A discussion of this can be found in Bishop (2006). ♢

Probabilistic graphical models have some convenient properties:

- They are a simple way to visualize the structure of a probabilistic model
- They can be used to design or motivate new kind of statistical models
- Inspection of the graph alone gives us insight into properties, e.g., conditional independence
- Complex computations for inference and learning in statistical models can be expressed in terms of graphical manipulations.

8.4.1 Graph Semantics



(a) Fully connected. (b) Not fully connected.

Figure 8.7 Examples of directed graphical models.

Directed graphical models/Bayesian networks are a method for representing conditional dependencies in a probabilistic model. They provide a visual description of the conditional probabilities, hence, providing a simple language for describing complex interdependence. The modular description also entails computational simplification. Directed links (arrows) between two nodes (random variables) are conditional probabilities. For example, the arrow between a and b in Figure 8.7(a) gives the conditional probability $p(b \mid a)$ of b given a.

Directed graphical models
Bayesian networks

With additional assumptions, the arrows can be used to indicate causal relationships (Pearl, 2009), but we do not make these assumptions here.

Directed graphical models can be derived from joint distributions if we know something about their factorization.

Example 8.7
Consider the joint distribution

$$p(a, b, c) = p(c \mid a, b)p(b \mid a)p(a) \tag{8.31}$$

of three random variables a, b, c. The factorization of the joint distribution in (8.31) tells us something about the relationship between the random variables:

- c depends directly on a and b
- b depends directly on a
- a depends neither on b nor on c

For the factorization in (8.31), we obtain the directed graphical model in Figure 8.7(a).

In general, we can construct the corresponding directed graphical model from a factorized joint distribution as follows:

1. Create a node for all random variables
2. For each conditional distribution, we add a directed link (arrow) to the graph from the nodes corresponding to the variables on which the distribution is conditioned on

The graph layout depends on the factorization of the joint distribution.

The graph layout depends on the choice of factorization of the joint distribution.

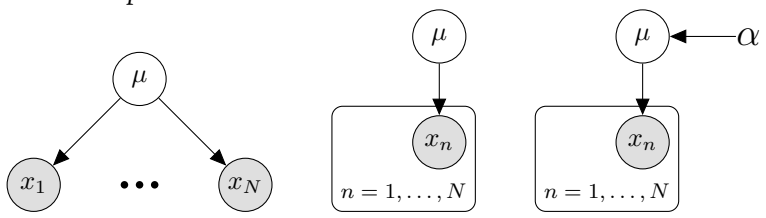
We discussed how to get from a known factorization of the joint distribution to the corresponding directed graphical model. Now, we will go exactly the opposite and describe how to extract the joint distribution of a set of random variables from a given graphical model.

Example 8.8
Let us look at the graphical model in Figure 8.7(b) and exploit two observations:

- The joint distribution $p(x_1, \ldots, x_5)$ we seek is the product of a set of conditionals, one for each node in the graph. In this particular example, we will need five conditionals.
- Each conditional depends only on the parents of the corresponding node in the graph. For example, x_4 will be conditioned on x_2.

With these two properties we arrive at the desired factorization of the joint distribution

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_5)p(x_2 \mid x_5)p(x_3 \mid x_1, x_2)p(x_4 \mid x_2)\,. \tag{8.32}$$



(a) Version with x_n explicit. (b) Version with plate notation. (c) Hyperparameter α on the latent μ.

Figure 8.8 Graphical models for a repeated Bernoulli experiment.

In general, the joint distribution $p(\boldsymbol{x}) = p(x_1, \ldots, x_K)$ is given as

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k \,|\, \text{Pa}_k) \tag{8.33}$$

where Pa_k means "the parent nodes of x_k".

We conclude this subsection with a concrete example of the coin flip experiment. Consider a Bernoulli experiment where the probability that the outcome x of this experiment is "heads" is

$$p(x \,|\, \mu) = \text{Ber}(\mu)\,. \tag{8.34}$$

We now repeat this experiment N times and observe outcomes $x_1, \ldots, x_N$ so that we obtain the joint distribution

$$p(x_1, \ldots, x_N|\mu) = \prod_{n=1}^{N} p(x_n|\mu)\,. \tag{8.35}$$

The expression on the right hand side is a product of Bernoulli distributions on each individual outcome because the experiments are independent. Recall from Section 6.4.3 that statistical independence means that the distribution factorizes. To write the graphical model down for this setting, we make the distinction between unobserved/latent variables and observed variables. Graphically, observed variables are denoted by shaded nodes so that we obtain the graphical model in Figure 8.8(a). We see that the single parameter μ is the same for all x_n, $n = 1, \ldots, N$. A more compact, but equivalent, graphical model for this setting is given in Figure 8.8(b), where we use the *plate* notation. The plate (box) repeats everything inside (in this case the observations x_n) N times. Therefore, both graphical models are equivalent, but the plate notation is more compact. Graphical models immediately allow us to place a hyper-prior on μ. Figure 8.8(c) places a Beta(α) prior on the latent variable μ. If we treat α as a constant (deterministic parameter), i.e., not a random variable, we omit the circle around it.

plate

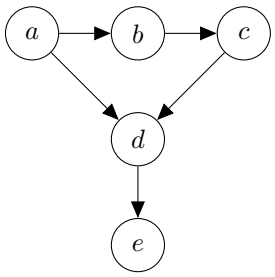


Figure 8.9 D-separation example.

8.4.2 Conditional Independence and D-Separation

Directed graphical models allow us to find conditional independence (Section 6.4.3) relationship properties of the joint distribution only by looking at the graph. A concept called *d-separation* (Pearl, 1988) is key to this.

d-separation

Consider a general directed graph in which $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are arbitrary non-intersecting sets of nodes (whose union may be smaller than the complete set of nodes in the graph). We wish to ascertain whether a particular conditional independence statement, $\mathcal{A}$ is conditionally independent of $\mathcal{B}$ given $\mathcal{C}$, denoted by

$$\mathcal{A} \perp\!\!\!\perp \mathcal{B} \,|\, \mathcal{C}\,, \tag{8.36}$$

is implied by a given directed acyclic graph. To do so, we consider all possible paths from any node in $\mathcal{A}$ to any nodes in $\mathcal{B}$. Any such path is said to be blocked if it includes any node such that either

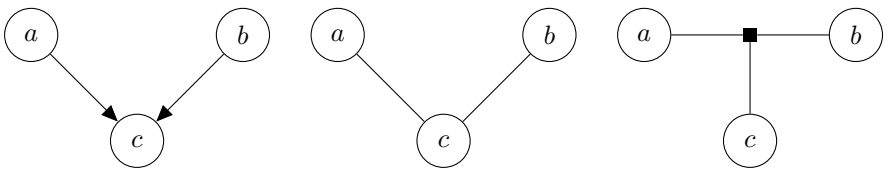
- the arrows on the path meet either head to tail or tail to tail at the node, and the node is in the set $\mathcal{C}$, or
- the arrows meet head to head at the node and neither the node nor any of its descendants is in the set $\mathcal{C}$.

If all paths are blocked, then $\mathcal{A}$ is said to be *d-separated* from $\mathcal{B}$ by $\mathcal{C}$, and the joint distribution over all of the variables in the graph will satisfy $\mathcal{A} \perp\!\!\!\perp \mathcal{B} \,|\, \mathcal{C}$.

Example 8.9 (Conditional Independence)
Consider the graphical model in Figure 8.9. By visual inspection we see that

$$b \perp\!\!\!\perp d \,|\, a, c\,, \tag{8.37a}$$
$$a \perp\!\!\!\perp c \,|\, b\,, \tag{8.37b}$$
$$b \not\perp\!\!\!\perp d \,|\, c\,, \tag{8.37c}$$
$$a \not\perp\!\!\!\perp c \,|\, b, e\,. \tag{8.37d}$$

Directed graphical models allow a compact representation of probabilisitic models, and we will see examples of directed graphical models in



(a) Directed graphical model (b) Undirected graphical model (c) Factor graph

Figure 8.10 Three types of graphical models: (a) Directed graphical models (Bayesian network); (b) Undirected graphical models (Markov random field); (c) Factor graphs.

Chapter 9, 10 and 11. The representation along with the concept of conditional independence, allows us to factorize the respective probabilisitic models into expressions that are easier to optimize.

Further Reading

An introduction to probabilistic graphical models can be found in Bishop (2006, Chapter 8), and an extensive description of the different applications and corresponding algorithmic implications can be found in Koller and Friedman (2009).

There are three main types of probabilistic graphical models:

Directed graphical models

Undirected graphical models

Factor graphs

- *Directed graphical models* (Bayesian networks), see Figure 8.11(a)
- *Undirected graphical models* (Markov random fields), see Figure 8.11(b)
- *Factor graphs*, see Figure 8.11(c)

Graphical models allow for graph-based algorithms for inference and learning, e.g., via local message passing. Applications range from ranking in online games (Herbrich et al., 2007) and computer vision (e.g., image segmentation, semantic labeling, image de-noising, image restoration (Sucar and Gillies, 1994; Shotton et al., 2006; Szeliski et al., 2008; Kittler and Föglein, 1984)) to coding theory (McEliece et al., 1998), solving linear equation systems (Shental et al., 2008) and iterative Bayesian state estimation in signal processing (Bickson et al., 2007; Deisenroth and Mohamed, 2012).

One topic which is particularly important in real applications that we do not discuss in this book is the idea of structured prediction (Bakir et al., 2007; Nowozin et al., 2014) which allow machine learning models to tackle predictions that are structured, for example sequences, trees and graphs. The popularity of neural network models has allowed more flexible probabilistic models to be used, resulting in many useful applications of structured models (Goodfellow et al., 2016, Chapter 16). In recent years, there has been a renewed interest in graphical models due to its applications to causal inference (Rosenbaum, 2017; Pearl, 2009; Imbens and Rubin, 2015; Peters et al., 2017).

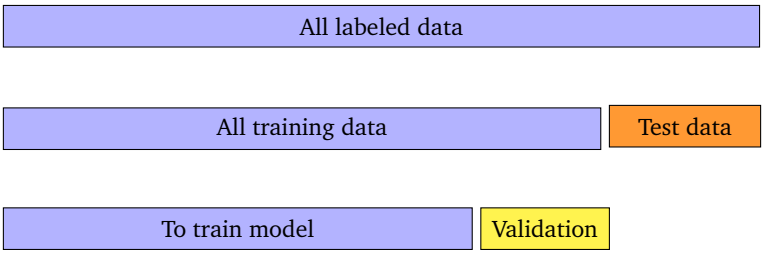


Figure 8.11 Nested cross validation. We perform two levels of K fold cross validation. The inner level is used to estimate the performance of a particular choice of model or hyperparameter on a internal validation set. The outer level is used to estimate generalization performance for *the best choice of model* chosen by the inner loop.

8.5 Model Selection

In machine learning, we often need to make high level modeling decisions that critically influence the performance of the model. The choices we make (e.g., the degree of the polynomial in a regression setting) influence the number and type of free parameters in the model and thereby also the flexibility and expressivity of the model. More complex models are more flexible in the sense that they can be used to describe more data sets. For instance, a polynomial of degree 1 (a line $y = a_0 + a_1 x$) can only be used to describe linear relations between inputs x and observations y. A polynomial of degree 2 can additionally describe quadratic relationships between inputs and observations.

Note that a polynomial $y = a_0 + a_1 x + a_2 x^2$ can also describe linear functions by setting $a_2 = 0$, i.e., it is strictly more expressive than a first-order polynomial.

One would now think that very flexible models are generally preferable to simple models because they are more expressive. A general problem is that at training time we can only use the training set to evaluate the performance of the model and learn its parameters. However, the performance on the training set is not really what we are interested in. In Section 8.2, we have seen that maximum likelihood estimation can lead to overfitting, especially when the training data set is small. Ideally, our model (also) works well on the test set (which is not available at training time). Therefore, we need some mechanisms for assessing how a model *generalizes* to unseen test data. *Model selection* is concerned with exactly this problem.

8.5.1 Nested Cross Validation

We have already seen an approach (cross validation in Section 8.1.4) that can be used for model selection. Recall that cross validation provides an estimate of the generalization error by repeatedly splitting the dataset into training and validation sets. We can apply this idea one more time, that is for each split, we can perform another round of cross validation. This is sometimes referred to as *nested cross validation*. We can test different model and hyperparameter choices in the inner loop. To distinguish the two levels, the set used to estimate the generalization performance is often

nested cross validation

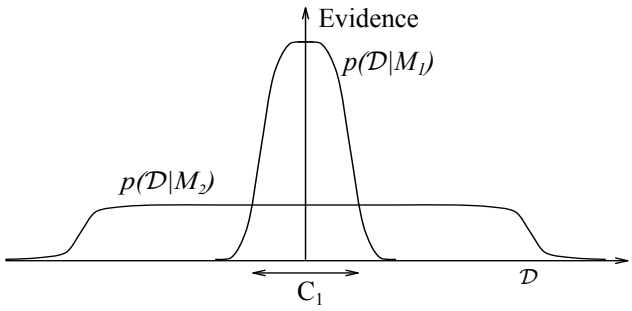


Figure 8.12 Bayesian inference embodies Occam's razor (MacKay, 2003), see text for description.

called the *test set* and the set used for choosing the best model is called the *validation set*.

test set
validation set

$$\mathbb{E}_{\mathcal{V}}[G(\mathcal{V}) \mid M] \approx \frac{1}{K} \sum_{k=1}^{K} G(\mathcal{V}^{(k)} \mid M)\,, \tag{8.38}$$

where $G(\mathcal{V})$ is the generalization error (e.g., RMSE) on the validation set $\mathcal{V}$ for model M. We repeat this procedure for all models and choose the model that performs best. Note that cross-validation not only gives us the expected generalization error, but we can also obtain high-order statistics, e.g., the standard error, an estimate of how uncertain the mean estimate is.

The standard error is defined as $\frac{\sigma}{\sqrt{K}}$, where K is the number of experiments and σ the standard deviation.

Once the model is chosen we can evaluate the final performance on the test set.

8.5.2 Bayesian Model Selection

There are many approaches to model selection, some of which are covered in this section. Generally, they all attempt to trade off model complexity and data fit: The objective is to find the simplest model that explains the data reasonably well. This concept is also known as *Occam's Razor*.

We assume that simpler models are less prone to overfitting than complex models.

Occam's Razor
We are looking for the simplest model that explains the data.

Remark. If we treat model selection as a hypothesis testing problem, we are looking for the simplest hypothesis that is consistent with the data (Murphy, 2012). ♢

One may consider placing a prior on models that favors simpler models. However, it is not necessary to do this: An "automatic Occam's Razor" is quantitatively embodied in the application of Bayesian probability (Spiegelhalter and Smith, 1980; MacKay, 1992; Jefferys and Berger, 1992). Figure 8.12 from MacKay (2003) gives us the basic intuition why complex and very expressive models may turn out to be a less probably choice for modeling a given dataset $\mathcal{D}$. Let us think of the horizontal axis representing the space of all possible datasets $\mathcal{D}$. If we are interested in the posterior probability $p(M_i \mid \mathcal{D})$ of model M_i given the data $\mathcal{D}$, we can

employ Bayes' theorem. Assuming a uniform prior $p(M)$ over all models, Bayes' theorem rewards models in proportion to how much they predicted the data that occurred. This probability of the data given model M_i, $p(\mathcal{D} \mid M_i)$, is called the *evidence* for M_i. A simple model M_1 can only predict a small number of datasets, which is shown by $p(\mathcal{D} \mid M_1)$; a more powerful model M_2 that has, e.g., more free parameters than M_1, is able to predict a greater variety of datasets. This means, however, that M_2 does not predict the datasets in region C_1 as strongly as M_1. Suppose that equal prior probabilities have been assigned to the two models. Then, if the data set falls in region C_1, the less powerful model M_1 is the more probable model.

Note that these predictions are quantified by a normalized probability distribution on $\mathcal{D}$, i.e., it needs to integrate/sum to 1.

evidence

Above, we argued that models need to be able to explain the data, i.e., there should be a way to generate data from a given model. Furthermore if the model has been appropriately learned from the data, then we expect that the generated data should be similar to the empirical data. For this, it is helpful to phrase model selection as a hierarchical inference problem, which allows us to compute the *posterior distribution over models*.

posterior distribution over models

Let us consider a finite number of models $M = \{M_1, \ldots, M_K\}$, where each model M_k possesses parameters $\boldsymbol{\theta}_k$. In *Bayesian model selection*, we place a prior $p(M)$ on the set of models. The corresponding *generative process* that allows us to generate data from this model is

$$M_k \sim p(M) \tag{8.39}$$

$$\boldsymbol{\theta}_k \mid M_k \sim p(\boldsymbol{\theta}_k) \tag{8.40}$$

$$\mathcal{D} \mid \boldsymbol{\theta}_k \sim p(\mathcal{D} \mid \boldsymbol{\theta}_k) \tag{8.41}$$

and illustrated in Figure 8.13.

Figure 8.13 Illustration of the hierarchical generative process in Bayesian model selection. We place a prior $p(M)$ on the set of models. For each model, there is a prior $p(\boldsymbol{\theta}_k \mid M_k)$ on the corresponding model parameters, which are then used to generate the data $\mathcal{D}$.

Bayesian model selection generative process

Given a training set $\mathcal{D}$, we apply Bayes' theorem and compute the posterior distribution over models as

$$p(M_k \mid \mathcal{D}) \propto p(M_k) p(\mathcal{D} \mid M_k) \,. \tag{8.42}$$

Note that this posterior no longer depends on the model parameters $\boldsymbol{\theta}_k$ because they have been integrated out in the Bayesian setting since

$$p(\mathcal{D} \mid M_k) = \int p(\mathcal{D} \mid \boldsymbol{\theta}_k) p(\boldsymbol{\theta}_k \mid M_k) d\boldsymbol{\theta}_k \,. \tag{8.43}$$

From the posterior in (8.42), we determine the MAP estimate as

$$M^* = \arg\max_{M_k} p(M_k \mid \mathcal{D}) \,. \tag{8.44}$$

With a uniform prior $p(M_k) = \frac{1}{K}$, which gives every model equal (prior) probability, determining the MAP estimate over models amounts to picking the model that maximizes the *model evidence/marginal likelihood*

model evidence marginal likelihood

$$p(\mathcal{D} \mid M_k) = \int p(\mathcal{D} \mid \boldsymbol{\theta}_k) p(\boldsymbol{\theta}_k \mid M_k) d\boldsymbol{\theta}_k \,, \tag{8.45}$$

where $p(\boldsymbol{\theta}_k \mid M_k)$ is the prior distribution of the model parameters $\boldsymbol{\theta}_k$ of model M_k.

Remark (Likelihood and Marginal Likelihood). There are some important differences between a likelihood and a marginal likelihood (evidence): While the likelihood is prone to overfitting, the marginal likelihood is typically not as the model parameters have been marginalized out (i.e., we no longer have to fit the parameters). Furthermore, the marginal likelihood automatically embodies a trade-off between model complexity and data fit (Occam's razor). ♢

8.5.3 Bayes Factors for Model Comparison

Consider the problem of comparing two probabilistic models M_1, M_2, given a data set $\mathcal{D}$. If we compute the posteriors $p(M_1 \mid \mathcal{D})$ and $p(M_2 \mid \mathcal{D})$, we can compute the ratio of the posteriors (*posterior odds*) posterior odds

$$\underbrace{\frac{p(M_1 \mid \mathcal{D})}{p(M_2 \mid \mathcal{D})}}_{\text{posterior odds}} = \frac{\frac{p(\mathcal{D} \mid M_1)p(M_1)}{p(\mathcal{D})}}{\frac{p(\mathcal{D} \mid M_2)p(M_2)}{p(\mathcal{D})}} = \underbrace{\frac{p(M_1)}{p(M_2)}}_{\text{prior odds}} \underbrace{\frac{p(\mathcal{D} \mid M_1)}{p(\mathcal{D} \mid M_2)}}_{\text{Bayes factor}} \tag{8.46}$$

The first fraction on the right-hand-side (prior odds) measures how much our prior (initial) beliefs favor M_1 over M_2. The ratio of the marginal likelihoods (second fraction on the right-hand-side) is called the *Bayes factor* and measures how well the data $\mathcal{D}$ is predicted by M_1 compared to M_2. Bayes factor

Remark. The *Jeffreys-Lindley paradox* states that the "Bayes factor always favors the simpler model since the probability of the data under a complex model with a diffuse prior will be very small" (Murphy, 2012). Here, a diffuse prior refers to a prior that does not favor specific models, i.e., many models are a priori plausible under this prior. ♢ Jeffreys-Lindley paradox

If we choose a uniform prior over models, the prior odds term in (8.46) is 1, i.e., the posterior odds is the ratio of the marginal likelihoods (Bayes factor)

$$\frac{p(\mathcal{D} \mid M_1)}{p(\mathcal{D} \mid M_2)}\,. \tag{8.47}$$

If the Bayes factor is greater than 1, we choose model M_1, otherwise model M_2.

Remark (Computing the Marginal Likelihood). The marginal likelihood plays an important role in model selection: We need to compute Bayes factors (8.46) and posterior distributions over models (8.42).

Unfortunately, computing the marginal likelihood requires us to solve an integral (8.45). This integration is generally analytically intractable, and we will have to resort to approximation techniques, e.g., numerical

integration (Stoer and Burlirsch, 2002), stochastic approximations using Monte Carlo (Murphy, 2012) or Bayesian Monte Carlo techniques (O'Hagan, 1991; Rasmussen and Ghahramani, 2003).

However, there are special cases in which we can solve it. In Section 6.7.1, we discussed conjugate models. If we choose a conjugate parameter prior $p(\boldsymbol{\theta})$, we can compute the marginal likelihood in closed form. In Chapter 9, we will do exactly this in the context of linear regression. ♢

Further Reading

We mentioned at the start of the section that there are high level modeling choices that influence the performance of the model. Examples include:

- The degree of a polynomial in a regression setting
- The number of components in a mixture model
- The network architecture of a (deep) neural network
- The type of kernel in a support vector machine
- The dimensionality of the latent space in PCA
- The learning rate (schedule) in an optimization algorithm

Rasmussen and Ghahramani (2001) showed that the automatic Occam's razor does not necessarily penalize the number of parameters in a model but it is active in terms of the complexity of functions. They also showed that the automatic Occam's razor also holds for Bayesian nonparametric models with many parameters, e.g., Gaussian processes.

In parametric models, the number of parameters is often related to the complexity of the model class.

If we focus on the maximum likelihood estimate, there exist a number of heuristics for model selection that discourage overfitting. These are called information criteria, and we choose the model with the largest value. The *Akaike Information Criterion (AIC)* (Akaike, 1974)

$$\log p(\boldsymbol{x} \mid \boldsymbol{\theta}) - M \tag{8.48}$$

corrects for the bias of the maximum likelihood estimator by addition of a penalty term to compensate for the overfitting of more complex models (with lots of parameters). Here, M is the number of model parameters.

The *Bayesian Information Criterion (BIC)* (Schwarz, 1978)

$$\ln p(\boldsymbol{x}) = \log \int p(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \approx \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) - \frac{1}{2} M \ln N \tag{8.49}$$

can be used for exponential family distributions. Here, N is the number of data points and M is the number of parameters. BIC penalizes model complexity more heavily than AIC.

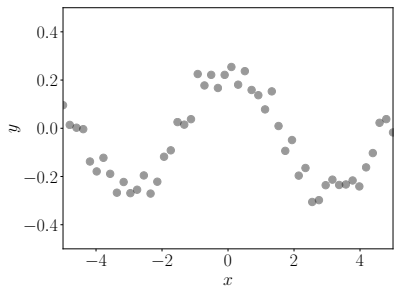
9

Linear Regression

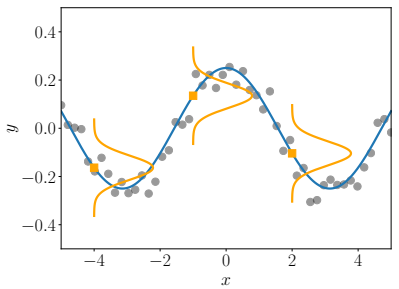
In the following, we will apply the mathematical concepts from Chapters 2, 5, 6 and 7 to solving linear regression (curve fitting) problems. In *regression*, we want to find a function f that maps inputs $\boldsymbol{x} \in \mathbb{R}^D$ to corresponding function values $f(\boldsymbol{x}) \in \mathbb{R}$ given a set of training inputs $\boldsymbol{x}_n$ and corresponding observations $y_n = f(\boldsymbol{x}_n) + \epsilon$, where ϵ is a random variable that comprises measurement noise and unmodeled processes. An illustration of such a regression problem is given in Figure 9.1. A typical regression problem is given in Figure 9.1(a): For some input values x we observe (noisy) function values $y = f(x) + \epsilon$. The task is to infer the function f that generated the data. A possible solution is given in Figure 9.1(b), where we also show three distributions centered at the function values $f(x)$ that represent the noise in the data.

regression

Regression is a fundamental problem in machine learning, and regression problems appear in a diverse range of research areas and applications, including time-series analysis (e.g., system identification), control and robotics (e.g., reinforcement learning, forward/inverse model learning), optimization (e.g., line searches, global optimization), and deep-learning applications (e.g., computer games, speech-to-text translation, image recognition, automatic video annotation). Regression is also a key ingredient of classification algorithms.



(a) Regression problem: Observed noisy function values from which we wish to infer the underlying function that generated the data.



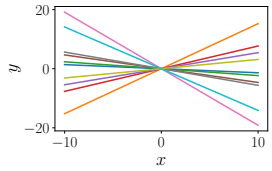
(b) Regression solution: Possible function that could have generated the data (blue) with indication of the measurement noise of the function value at the corresponding inputs (orange distributions).

Figure 9.1
(a) Dataset;
(b) Possible solution to the regression problem.

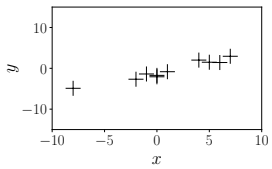
Finding a regression function requires solving a variety of problems, including

- **Choice of the model (type) and the parametrization** of the regression function. Given a data set, what function classes (e.g., polynomials) are good candidates for modeling the data, and what particular parametrization (e.g., degree of the polynomial) should we choose? Model selection, as discussed in Section 8.5, allows us to compare various models to find the simplest model that explains the training data reasonably well.
- **Finding good parameters.** Having chosen a model of the regression function, how do we find good model parameters? Here, we will need to look at different loss/objective functions (they determine what a "good" fit is) and optimization algorithms that allow us to minimize this loss.
- **Overfitting and model selection.** Overfitting is a problem when the regression function fits the training data "too well" but does not generalize to unseen test data. Overfitting typically occurs if the underlying model (or its parametrization) is overly flexible and expressive, see Section 8.5. We will look at the underlying reasons and discuss ways to mitigate the effect of overfitting in the context of linear regression.
- **Relationship between loss functions and parameter priors.** Loss functions (optimization objectives) are often motivated and induced by probabilistic models. We will look at the connection between loss functions and the underlying prior assumptions that induce these losses.
- **Uncertainty modeling.** In any practical setting, we have access to only a finite, potentially large, amount of (training) data for selecting the model class and the corresponding parameters. Given that this finite amount of training data does not cover all possible scenarios, we way want to describe the remaining parameter uncertainty to obtain a measure of confidence of the model's prediction at test time; the smaller the training set the more important uncertainty modeling. Consistent modeling of uncertainty equips model predictions with confidence bounds.

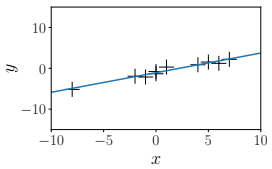
In the following, we will be using the mathematical tools from Chapters 3, 5, 6 and 7 to solve linear regression problems. We will discuss maximum likelihood and maximum a posteriori (MAP) estimation to find optimal model parameters. Using these parameter estimates, we will have a brief look at generalization errors and overfitting. Toward the end of this chapter, we will discuss Bayesian linear regression, which allows us to reason about model parameters at a higher level, thereby removing some of the problems encountered in maximum likelihood and MAP estimation.



(a) Example functions (straight lines) that can be described using the linear model in (9.2).



(b) Training set.



(c) Maximum likelihood estimate.

Figure 9.2 Linear regression without features. (a) Example functions that fall into this category. (b) Training set. (c) Maximum likelihood estimate.

9.1 Problem Formulation

We consider the regression problem

$$y = f(\boldsymbol{x}) + \epsilon\,, \tag{9.1}$$

where $\boldsymbol{x} \in \mathbb{R}^D$ are inputs and $y \in \mathbb{R}$ are noisy function values (targets). Furthermore, $\epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$ is independent, identically distributed (i.i.d.) measurement noise. In this particular case, ϵ is Gaussian distributed with mean 0 and variance σ^2. Our objective is to find a function that is close (similar) to the unknown function that generated the data.

In this chapter, we focus on parametric models, i.e., we choose a parametrized function f and find parameters that "work well" for modeling the data. In linear regression, we consider the special case that the parameters appear linearly in our model. An example of linear regression is

$$y = f(\boldsymbol{x}) + \epsilon = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon\,, \tag{9.2}$$

where $\boldsymbol{\theta} \in \mathbb{R}^D$ are the parameters we seek, and $\epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$ is i.i.d. Gaussian measurement/observation noise. The class of functions described by (9.2) are straight lines that pass through the origin. In (9.2), we chose a parametrization $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\theta}$. For the time being we assume that the noise variance σ^2 is known. The noise model induces the *likelihood*

likelihood

$$p(y \,|\, \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\big(y \,|\, \boldsymbol{x}^\top \boldsymbol{\theta},\, \sigma^2\big)\,, \tag{9.3}$$

which is the probability of observing a target value y given that we know the input location $\boldsymbol{x}$ and the parameters $\boldsymbol{\theta}$. Note that the only source of uncertainty originates from the observation noise (as $\boldsymbol{x}$ and $\boldsymbol{\theta}$ are assumed known in (9.3))—without any observation noise, the relationship between $\boldsymbol{x}$ and y would be deterministic and (9.3) would be a delta distribution.

For $x, \theta \in \mathbb{R}$ the linear regression model in (9.2) describes straight lines (linear functions), and the parameter θ would be the slope of the line. Figure 9.2(a) shows some examples. This model is not only linear in the parameters, but also linear in the inputs x. We will see later that $y = \phi(x)\theta$ for nonlinear transformations ϕ is also a linear regression model because "linear regression" refers to models that are "linear in the parameters", i.e., models that describe a function by a linear combination of input features.

Linear regression refers to models that are linear in the parameters.

In the following, we will discuss in more detail how to find good parameters $\boldsymbol{\theta}$ and how to evaluate whether a parameter set "works well".

9.2 Parameter Estimation

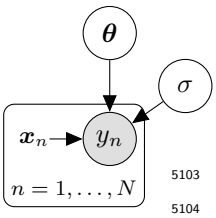
Consider the linear regression setting (9.2) and assume we are given a *training set* $\mathcal{D}$ consisting of N inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and corresponding observations/targets $y_n \in \mathbb{R}$, $n = 1, \dots, N$. The corresponding graphical model is given in Figure 9.3. Note that y_i and y_j are conditionally independent given their respective inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$, such that the likelihood function factorizes according to

$$p(y_1, \dots, y_N \mid \boldsymbol{x}_1, \dots, \boldsymbol{x}_N) = \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta},\, \sigma^2\right). \quad (9.4)$$

The likelihood and the factors $p(y_n \mid \boldsymbol{x}_n)$ are Gaussian due to the noise distribution.

training set

Figure 9.3 Probabilistic graphical model for linear regression. Observed random variables are shaded, deterministic/known values are without circles. The parameters $\boldsymbol{\theta}$ are treated as unknown/latent quantities.



In the following, we are interested in finding optimal parameters $\boldsymbol{\theta}^* \in \mathbb{R}^D$ for the linear regression model (9.2). Once the parameters $\boldsymbol{\theta}^*$ are found, we can predict function values by using this parameter estimate in (9.2) so that at an arbitrary test input $\boldsymbol{x}_*$ we predict the probability for an output y_* as

$$p(y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta}^*) = \mathcal{N}\left(y_* \mid \boldsymbol{x}_*^\top \boldsymbol{\theta}^*,\, \sigma^2\right). \quad (9.5)$$

In the following, we will have a look at parameter estimation by maximizing the likelihood, a topic that we already covered to some degree in Section 8.2.

9.2.1 Maximum Likelihood Estimation

maximum likelihood estimation

A widely used approach to finding the desired parameters $\boldsymbol{\theta}_{\text{ML}}$ is *maximum likelihood estimation* where we find parameters $\boldsymbol{\theta}_{\text{ML}}$ that maximize the likelihood (9.4).

Maximizing the likelihood means maximizing the probability of the (training) data given the parameters.

We obtain the maximum likelihood parameters as

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}), \quad (9.6)$$

design matrix

where we define the *design matrix* $\boldsymbol{X} := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} := [y_1, \dots, y_N]^\top \in \mathbb{R}^N$ as the collections of training inputs and targets, respectively. Note that the nth row in the design matrix $\boldsymbol{X}$ corresponds to the data point $\boldsymbol{x}_n$.

The likelihood is not a probability distribution in the parameters.

Remark. Note that the likelihood is not a probability distribution in $\boldsymbol{\theta}$: It is simply a function of the parameters $\boldsymbol{\theta}$ but does not integrate to 1 (i.e., it is unnormalized), and may not even be integrable with respect to $\boldsymbol{\theta}$. However, the likelihood in (9.6) is a normalized probability distribution in the data $\boldsymbol{y}$. ♢

ce the logarithm (strictly) notonically reasing function, optimum of a ction f is ntical to the imum of $\log f$.

To find the desired parameters $\boldsymbol{\theta}_{\text{ML}}$ that maximize the likelihood, we typically perform gradient ascent (or gradient descent on the negative likelihood). In the case of linear regression we consider here, however, a closed-form solution exists, which makes iterative gradient descent unnecessary. In practice, instead of maximizing the likelihood directly, we apply the log-transformation to the likelihood function and minimize the negative log-likelihood.

Remark (Log Transformation). Since the likelihood function is a product of N Gaussian distributions, the log-transformation is useful since a) it does not suffer from numerical underflow, b) the differentiation rules will turn out simpler. Numerical underflow will be a problem when we multiply N probabilities, where N is the number of data points, since we cannot represent very small numbers, such as 10^{-256}. Furthermore, the log-transform will turn the product into a sum of log-probabilities such that the corresponding gradient is a sum of individual gradients, instead of a repeated application of the product rule (5.54) to compute the gradient of a product of N terms. ♢

To find the optimal parameters $\boldsymbol{\theta}_{\text{ML}}$ of our linear regression problem, we minimize the negative log-likelihood

$$-\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = -\log \prod_{n=1}^{N} p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) \,, \quad (9.7)$$

where we exploited that the likelihood (9.4) factorizes over the number of data points due to our independence assumption on the training set.

In the linear regression model (9.2) the likelihood is Gaussian (due to the Gaussian additive noise term), such that we arrive at

$$\log p(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = -\frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + \text{const} \quad (9.8)$$

where the constant includes all terms independent of $\boldsymbol{\theta}$. Using (9.8) in the negative log-likelihood (9.7) we obtain (ignoring the constant terms)

The negative log-likelihood function is also called *error function*.

$$\mathcal{L}(\boldsymbol{\theta}) := -\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 \quad (9.9a)$$

$$= \frac{1}{2\sigma^2} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) = \frac{1}{2\sigma^2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2 \,, \quad (9.9b)$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$.

Remark. There is some notation overloading: We often summarize the set of training inputs in $\boldsymbol{X}$, whereas in the design matrix we additionally assume a specific "shape". ♢

In (9.9b) we used the fact that the sum of squared errors between the observations y_n and the corresponding model prediction $\boldsymbol{x}_n^\top \boldsymbol{\theta}$ equals the

squared distance between $\boldsymbol{y}$ and $\boldsymbol{X}\boldsymbol{\theta}$. Remember from Section 3.1 that $\|\boldsymbol{x}\|^2 = \boldsymbol{x}^\top \boldsymbol{x}$ if we choose the dot product as the inner product.

With (9.9b) we have now a concrete form of the negative log-likelihood function we need to optimize. We immediately see that (9.9b) is quadratic in $\boldsymbol{\theta}$. This means that we can find a unique global solution $\boldsymbol{\theta}_{\text{ML}}$ for minimizing the negative log-likelihood $\mathcal{L}$. We can find the global optimum by computing the gradient of $\mathcal{L}$, setting it to $\mathbf{0}$ and solving for $\boldsymbol{\theta}$.

Using the results from Chapter 5, we compute the gradient of $\mathcal{L}$ with respect to the parameters as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})\right) \tag{9.10a}$$

$$= \frac{1}{2\sigma^2}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}\right) \tag{9.10b}$$

$$= \frac{1}{\sigma^2}(-\boldsymbol{y}^\top\boldsymbol{X} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}) \in \mathbb{R}^{1\times D}\,. \tag{9.10c}$$

As a necessary optimality condition we set this gradient to $\mathbf{0}$ and obtain

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \mathbf{0} \overset{(9.10c)}{\iff} \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{y}^\top\boldsymbol{X} \tag{9.11a}$$

$$\iff \boldsymbol{\theta}^\top = \boldsymbol{y}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1} \tag{9.11b}$$

$$\iff \boldsymbol{\theta}_{\text{ML}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}\,. \tag{9.11c}$$

We could right-multiply the first equation by $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ because $\boldsymbol{X}^\top\boldsymbol{X}$ is positive definite (if we do not have two identical inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$ for $i \neq j$).

Remark. In this case, setting the gradient to $\mathbf{0}$ is a necessary and sufficient condition and we obtain a global minimum since the Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{D\times D}$ is positive definite. ♢

Example 9.1 (Fitting Lines)
Let us have a look at Figure 9.2, where we aim to fit a straight line $f(x) = \theta x$, where θ is an unknown slope, to a data set using maximum likelihood estimation. Examples of functions in this model class (straight lines) are shown in Figure 9.2(a). For the data set shown in Figure 9.2(b) we find the maximum likelihood estimate of the slope parameter θ using (9.11c) and obtain the maximum likelihood linear function in Figure 9.2(c).

Maximum Likelihood Estimation with Features

Linear regression refers to "linear-in-the-parameters" regression models, but the inputs can undergo any nonlinear transformation.

So far, we considered the linear regression setting described in (9.2), which allowed us to fit straight lines to data using maximum likelihood estimation. However, straight lines are not particularly expressive when it comes to fitting more interesting data. Fortunately, linear regression offers us a way to fit nonlinear functions within the linear regression framework: Since "linear regression" only refers to "linear in the parameters", we can

perform an arbitrary nonlinear transformation $\phi(\boldsymbol{x})$ of the inputs $\boldsymbol{x}$ and then linearly combine the components of the result. The model parameters $\boldsymbol{\theta}$ still appear only linearly. The corresponding linear regression model is

$$y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon = \sum_{k=0}^{K-1} \theta_k \phi_k(\boldsymbol{x}) + \epsilon\,, \tag{9.12}$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^K$ is a (nonlinear) transformation of the inputs $\boldsymbol{x}$ and $\phi_k : \mathbb{R}^D \to \mathbb{R}$ is the kth component of the *feature vector* $\boldsymbol{\phi}$. feature vector

Example 9.2 (Polynomial Regression)
We are concerned with a regression problem $y = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta} + \epsilon$, where $x \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathbb{R}^K$. A transformation that is often used in this context is

$$\boldsymbol{\phi}(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^{K-1} \end{bmatrix} \in \mathbb{R}^K\,. \tag{9.13}$$

This means, we "lift" the original one-dimensional input space into a K-dimensional feature space consisting of all monomials x^k for $k = 0, \ldots, K-1$. With these features, we can model polynomials of degree $\leqslant K-1$ within the framework of linear regression: A polynomial of degree $K-1$ is

$$f(x) = \sum_{k=0}^{K-1} \theta_k x^k = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta} \tag{9.14}$$

where $\boldsymbol{\phi}$ is defined in (9.13) and $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{K-1}]^\top \in \mathbb{R}^K$ contains the (linear) parameters θ_k.

Let us now have a look at maximum likelihood estimation of the parameters $\boldsymbol{\theta}$ in the linear regression model (9.12). We consider training inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and targets $y_n \in \mathbb{R}$, $n = 1, \ldots, N$, and define the *feature matrix* (*design matrix*) as feature matrix design matrix

$$\boldsymbol{\Phi} := \begin{bmatrix} \boldsymbol{\phi}^\top(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\phi}^\top(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \cdots & \phi_{K-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \cdots & \phi_{K-1}(\boldsymbol{x}_2) \\ \vdots & & \vdots \\ \phi_0(\boldsymbol{x}_N) & \cdots & \phi_{K-1}(\boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times K}\,, \tag{9.15}$$

where $\Phi_{ij} = \phi_j(\boldsymbol{x}_i)$ and $\phi_j : \mathbb{R}^D \to \mathbb{R}$.

Example 9.3 (Feature Matrix for Second-order Polynomials)
For a second-order polynomial and N training points $x_n \in \mathbb{R}, n = 1, \ldots, N$, the feature matrix is

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix} . \tag{9.16}$$

With the feature matrix $\boldsymbol{\Phi}$ defined in (9.15) the negative log-likelihood for the linear regression model (9.12) can be written as

$$-\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \text{const} . \tag{9.17}$$

Comparing (9.17) with the negative log-likelihood in (9.9b) for the "feature-free" model, we immediately see we just need to replace $\boldsymbol{X}$ with $\boldsymbol{\Phi}$. Since both $\boldsymbol{X}$ and $\boldsymbol{\Phi}$ are independent of the parameters $\boldsymbol{\theta}$ that we wish to optimize, we arrive immediately at the *maximum likelihood estimate*

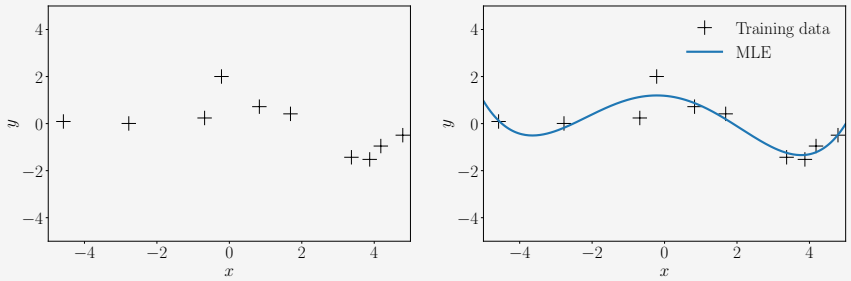
maximum likelihood estimate

$$\boldsymbol{\theta}_{\text{ML}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y} \tag{9.18}$$

for the linear regression problem with nonlinear features defined in (9.12).

Remark. When we were working without features, we required $\boldsymbol{X}^\top \boldsymbol{X}$ to be invertible, which is the case when the rows of $\boldsymbol{X}$ are linearly independent. In (9.18), we therefore require $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ to be invertible. This is the case if and only if the rows of the feature matrix are linearly independent. Nonlinear feature transformations can make previously linearly dependent inputs $\boldsymbol{X}$ linearly independent (and vice versa). ♢

Example 9.4 (Maximum Likelihood Polynomial Fit)



(a) Regression data set.

(b) Polynomial of degree 4 determined by maximum likelihood estimation.

Figure 9.4 Polynomial regression. (a) Data set consisting of (x_n, y_n) pairs, $n = 1, \ldots, 10$; (b) Maximum likelihood polynomial of degree 4.

Consider the data set in Figure 9.5(a). The data set consists of $N = 20$

pairs (x_n, y_n), where $x_n \sim \mathcal{U}[-5, 5]$ and $y_n = -\sin(x_n/5) + \cos(x_n) + \epsilon$, where $\epsilon \sim \mathcal{N}(0,\, 0.2^2)$.

We fit a polynomial of degree $K = 4$ using maximum likelihood estimation, i.e., parameters $\boldsymbol{\theta}_{\text{ML}}$ are given in (9.18). The maximum likelihood estimate yields function values $\boldsymbol{\phi}^\top(x_*)\boldsymbol{\theta}_{\text{ML}}$ at any test location x_*. The result is shown in Figure 9.5(b).

Estimating the Noise Variance

Thus far, we assumed that the noise variance σ^2 is known. However, we can also use the principle of maximum likelihood estimation to obtain σ^2_{ML} for the noise variance. To do this, we follow the standard procedure: we write down the log-likelihood, compute its derivative with respect to $\sigma^2 > 0$, set it to 0 and solve:

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \sigma^2) = \sum_{n=1}^{N} \log \mathcal{N}\big(y_n \mid \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n),\, \sigma^2\big) \tag{9.19a}$$

$$= \sum_{n=1}^{N} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(y_n - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n))^2 \right) \tag{9.19b}$$

$$= -\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2} \underbrace{\sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n))^2}_{=:s} + \text{const}\,. \tag{9.19c}$$

The partial derivative of the log-likelihood with respect to σ^2 is then

$$\frac{\partial \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}, \sigma^2)}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4} s = 0 \tag{9.20a}$$

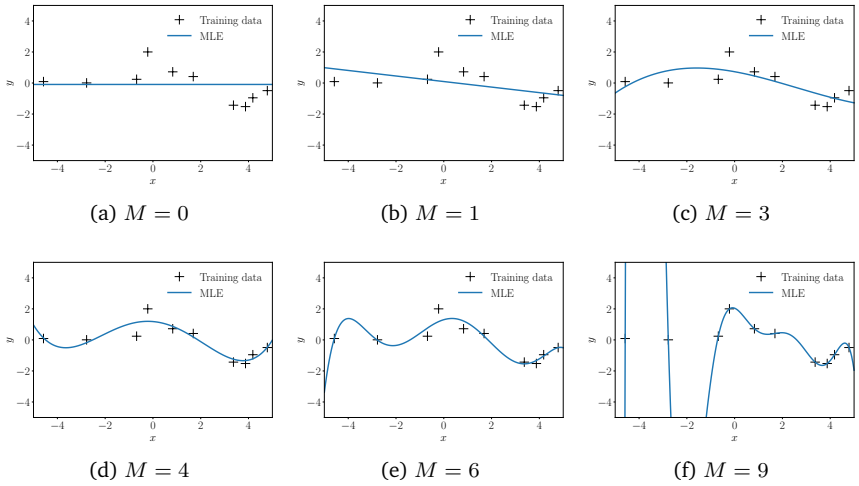
$$\iff \frac{N}{2\sigma^2} = \frac{s}{2\sigma^4} \tag{9.20b}$$

$$\iff \sigma^2_{\text{ML}} = \frac{s}{N} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n))^2\,. \tag{9.20c}$$

Therefore, the maximum likelihood estimate for the noise variance is the mean squared distance between the noise-free function values $\boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$ and the corresponding noisy observations y_n at $\boldsymbol{x}_n$, for $n = 1, \ldots, N$.

9.2.2 Overfitting in Linear Regression

We just discussed how to use maximum likelihood estimation to fit linear models (e.g., polynomials) to data. We can evaluate the quality of the model by computing the error/loss incurred. One way of doing this is to compute the negative log-likelihood (9.9b), which we minimized to determine the MLE. Alternatively, given that the noise parameter σ^2 is not



(a) $M = 0$ (b) $M = 1$ (c) $M = 3$

(d) $M = 4$ (e) $M = 6$ (f) $M = 9$

Figure 9.5 Maximum likelihood fits for different polynomial degrees M.

a free model parameter, we can ignore the scaling by $1/\sigma^2$, so that we end up with a squared-error-loss function $\|\boldsymbol{y} - \boldsymbol{\Phi\theta}\|^2$. Instead of using this squared loss, we often use the *root mean squared error (RMSE)*

root mean squared error (RMSE)

$$\sqrt{\|\boldsymbol{y} - \boldsymbol{\Phi\theta}\|^2 / N} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2}\,, \tag{9.21}$$

The RMSE is normalized.

which (a) allows us to compare errors of data sets with different sizes and (b) has the same scale and the same units as the observed function values y_n. For example, assume we fit a model that maps post-codes ($\boldsymbol{x}$ is given in latitude,longitude) to house prices (y-values are EUR). Then, the RMSE is also measured in EUR, whereas the squared error is given in EUR^2. If we choose to include the factor σ^2 from the original negative log-likelihood (9.9b) then we end up with a "unit-free" objective.

For model selection (see Section 8.5) we can use the RMSE (or the negative log-likelihood) to determine the best degree of the polynomial by finding the polynomial degree M that minimizes the objective. Given that the polynomial degree is a natural number, we can perform a brute-force search and enumerate all (reasonable) values of M. For a training set of size N it is sufficient to test $0 \leqslant M \leqslant N-1$. For $M \geqslant N$ we would need to solve an underdetermined system of linear equations so that we would end up with infinitely many solutions.

Figure 9.5 shows a number of polynomial fits determined by maximum likelihood for the dataset from Figure 9.5(a) with $N = 10$ observations. We notice that polynomials of low degree (e.g., constants ($M = 0$) or linear ($M = 1$) fit the data poorly and, hence, are poor representations of the true underlying function. For degrees $M = 3, \ldots, 5$ the fits look plausible and smoothly interpolate the data. When we go to higher-degree polynomials, we notice that they fit the data better and better. In the extreme

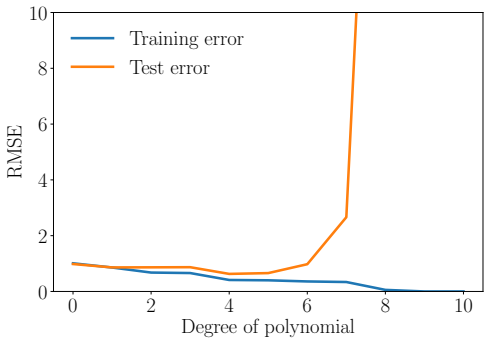


Figure 9.6 Training and test error.

case of $M = N - 1 = 9$, the function will pass through every single data point. However, these high-degree polynomials oscillate wildly and are a poor representation of the underlying function that generated the data, such that we suffer from *overfitting*.

overfitting

Remember that the goal is to achieve good generalization by making accurate predictions for new (unseen) data. We obtain some quantitative insight into the dependence of the generalization performance on the polynomial of degree M by considering a separate test set comprising 200 data points generated using exactly the same procedure used to generate the training set. As test inputs, we chose a linear grid of 200 points in the interval of $[-5, 5]$. For each choice of M, we evaluate the RMSE (9.21) for both the training data and the test data.

Note that the noise variance $\sigma^2 > 0$.

Looking now at the test error, which is a qualitive measure of the generalization properties of the corresponding polynomial, we notice that initially the test error decreases, see Figure 9.6 (orange). For fourth-order polynomials the test error is relatively low and stays relatively constant up to degree 5. However, from degree 6 onward the test error increases significantly, and high-order polynomials have very bad generalization properties. In this particular example, this also is evident from the corresponding maximum likelihood fits in Figure 9.5. Note that the *training error* (blue curve in Figure 9.6) never increases when the degree of the polynomial increases. In our example, the best generalization (the point of the smallest *test error*) is obtained for a polynomial of degree $M = 4$.

training error

test error

9.2.3 Regularization and Maximum A Posteriori Estimation

We just saw that maximum likelihood estimation is prone to overfitting. It often happens that the magnitude of the parameter values becomes relatively big if we run into overfitting (Bishop, 2006). One way to mitigate the effect of overfitting is to penalize big parameter values by a technique called *regularization*. In regularization, we add a term to the log-likelihood that penalizes the magnitude of the parameters $\boldsymbol{\theta}$. A typical example is a

regularization

regularized "loss function" of the form

$$-\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 \,, \tag{9.22}$$

regularizer 5218 where the second term is the *regularizer*, and $\lambda \geqslant 0$ controls the "strict-
5219 ness" of the regularization.

5220 *Remark.* Instead of the Euclidean norm $\|\cdot\|_2$, we can choose any p-norm
5221 $\|\cdot\|_p$. In practice, smaller values for p lead to sparser solutions. Here,
5222 "sparse" means that many parameter values $\theta_n = 0$, which is also use-
LASSO 5223 ful for variable selection. For $p = 1$, the regularizer is called *LASSO* (least
5224 absolute shrinkage and selection operator) and was proposed by Tibshi-
5225 rani (1996). ♢

From a probabilistic perspective, adding a regularizer is identical to placing a prior distribution $p(\boldsymbol{\theta})$ on the parameters and then selecting the parameters that maximize the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$, i.e., we choose the parameters $\boldsymbol{\theta}$ that are "most probable" given the training data. The posterior over the parameters $\boldsymbol{\theta}$, given the training data $\boldsymbol{X}, \boldsymbol{y}$, is obtained by applying Bayes' theorem as

$$p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{X})} \,. \tag{9.23}$$

5226 The parameter vector $\boldsymbol{\theta}_{\text{MAP}}$ that maximizes the posterior (9.23) is called
maximum 5227 the *maximum a-posteriori* (*MAP*) estimate.
a-posteriori
MAP To find the MAP estimate, we follow steps that are similar in flavor to maximum likelihood estimation. We start with the log-transform and compute the log-posterior as

$$\log p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + \text{const} \,, \tag{9.24}$$

5228 where the constant comprises the terms that are independent of $\boldsymbol{\theta}$. We see
5229 that the log-posterior in (9.24) is the sum of the log-likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$
5230 and the log-prior $\log p(\boldsymbol{\theta})$.

Remark (Relation to Regularization). Choosing a Gaussian parameter prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2 \boldsymbol{I})$, $b^2 = \frac{1}{2\lambda}$, the (negative) log-prior term will be

$$-\log p(\boldsymbol{\theta}) = \underbrace{\lambda \boldsymbol{\theta}^\top \boldsymbol{\theta}}_{=\lambda \|\boldsymbol{\theta}\|_2^2} + \text{const} \,, \tag{9.25}$$

5231 and we recover exactly the regularization term in (9.22). This means that
5232 for a quadratic regularization, the regularization parameter λ in (9.22)
5233 corresponds to twice the precision (inverse variance) of the Gaussian (iso-
5234 tropic) prior $p(\boldsymbol{\theta})$. Therefore, the log-prior in (9.24) reflects the impact
5235 of the regularizer that penalizes implausible values, i.e., values that are
5236 unlikely under the prior. ♢

To find the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$, we minimize the negative log-posterior

distribution with respect to $\boldsymbol{\theta}$, i.e., we solve

$$\boldsymbol{\theta}_{\text{MAP}} \in \arg\min_{\boldsymbol{\theta}} \{-\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\} . \tag{9.26}$$

We determine the gradient of the negative log-posterior with respect to $\boldsymbol{\theta}$ as

$$-\frac{\mathrm{d} \log p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})}{\mathrm{d}\boldsymbol{\theta}} = -\frac{\mathrm{d} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} - \frac{\mathrm{d} \log p(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} , \tag{9.27}$$

where we identify the first term on the right-hand-side as the gradient of the negative log-likelihood given in (9.10c).

More concretely, with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2\boldsymbol{I})$ on the parameters $\boldsymbol{\theta}$, the negative log-posterior for the linear regression setting (9.12), we obtain the negative log posterior

$$-\log p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{1}{2b^2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{const} . \tag{9.28}$$

Here, the first term corresponds to the contribution from the log-likelihood, and the second term originates from the log-prior. The gradient of the log-posterior with respect to the parameters $\boldsymbol{\theta}$ is then

$$-\frac{\mathrm{d} \log p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})}{\mathrm{d}\boldsymbol{\theta}} = \frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top . \tag{9.29}$$

We will find the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$ by setting this gradient to $\mathbf{0}$:

$$\frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top = \mathbf{0} \tag{9.30a}$$

$$\iff \boldsymbol{\theta}^\top\left(\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{1}{b^2}\boldsymbol{I}\right) - \frac{1}{\sigma^2}\boldsymbol{y}^\top\boldsymbol{\Phi} = \mathbf{0} \tag{9.30b}$$

$$\iff \boldsymbol{\theta}^\top\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right) = \boldsymbol{y}^\top\boldsymbol{\Phi} \tag{9.30c}$$

$$\iff \boldsymbol{\theta}^\top = \boldsymbol{y}^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1} \tag{9.30d}$$

so that we obtain the MAP estimate (by transposing both sides of the last equality)

$$\boldsymbol{\theta}_{\text{MAP}} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} . \tag{9.31}$$

$\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is symmetric and positive semidefinite and the additional term is strictly positive definite, such that all eigenvalues of the matrix to be inverted are positive.

Comparing the MAP estimate in (9.31) with the maximum likelihood estimate in (9.18) we see that the only difference between both solutions is the additional term $\frac{\sigma^2}{b^2}\boldsymbol{I}$ in the inverse matrix. This term ensures that $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}$ is symmetric and strictly positive definite (i.e., its inverse exists) and plays the role of the *regularizer*.

regularizer

Example 9.5 (MAP Estimation for Polynomial Regression)

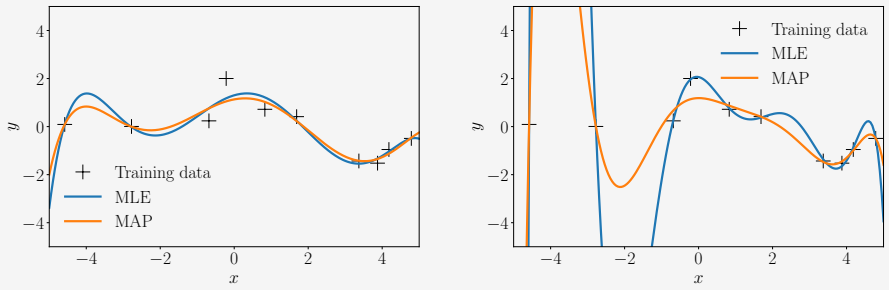


Figure 9.7 Polynomial regression: Maximum likelihood and MAP estimates.

In the polynomial regression example from Section 9.2.1, we place a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on the parameters $\boldsymbol{\theta}$ and determine the MAP estimates according to (9.31). In Figure 9.7, we show both the maximum likelihood and the MAP estimates for polynomials of degree 6 (left) and degree 8 (right). The prior (regularizer) does not play a significant role for the low-degree polynomial, but keeps the function relatively smooth for higher-degree polynomials. However, the MAP estimate can only push the boundaries of overfitting – it is not a general solution to this problem.

In the following, we will discuss Bayesian linear regression where we average over all plausible sets of parameters instead of focusing on a point estimate.

9.3 Bayesian Linear Regression

Previously, we looked at linear regression models where we estimated the model parameters $\boldsymbol{\theta}$, e.g., by means of maximum likelihood or MAP estimation. We discovered that MLE can lead to severe overfitting, in particular, in the small-data regime. MAP addresses this issue by placing a prior on the parameters that plays the role of a regularizer.

Bayesian linear regression

Bayesian linear regression pushes the idea of the parameter prior a step further and does not even attempt to compute a point estimate of the parameters, but instead the full posterior over the parameters is taken into account when making predictions. This means we do not fit any parameters, but we compute an average over all plausible parameters settings (according to the posterior).

9.3.1 Model

In Bayesian linear regression, we consider the model

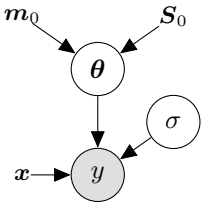
$$\begin{aligned} &\text{prior} && p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{m}_0,\, \boldsymbol{S}_0\right), \\ &\text{likelihood} && p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\left(y \mid \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta},\, \sigma^2\right), \end{aligned} \tag{9.32}$$

where we now explicitly place a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{m}_0,\, \boldsymbol{S}_0\right)$ on $\boldsymbol{\theta}$, which turns the parameter vector into a latent variable. The full probabilistic model, i.e., the joint distribution of observed and latent variables, y and $\boldsymbol{\theta}$, respectively, is

$$p(y, \boldsymbol{\theta} \mid \boldsymbol{x}) = p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}), \tag{9.33}$$

which allows us to write down the corresponding graphical model in Figure 9.8, where we made the parameters of the Gaussian prior on $\boldsymbol{\theta}$ explicit.

Figure 9.8 Graphical model for Bayesian linear regression.



9.3.2 Prior Predictions

In practice, we are usually not so much interested in the parameter values $\boldsymbol{\theta}$. Instead, our focus often lies in the predictions we make with those parameter values. In a Bayesian setting, we take the parameter distribution and average over all plausible parameter settings when we make predictions. More specifically, to make predictions at an input location $\boldsymbol{x}_*$, we integrate out $\boldsymbol{\theta}$ and obtain

$$p(y_* \mid \boldsymbol{x}_*) = \int p(y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta}}[p(y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta})], \tag{9.34}$$

which we can interpret as the average prediction of $y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta}$ for all plausible parameters $\boldsymbol{\theta}$ according to the prior distribution $p(\boldsymbol{\theta})$. Note that predictions using the prior distribution only require to specify the input locations $\boldsymbol{x}_*$, but no training data.

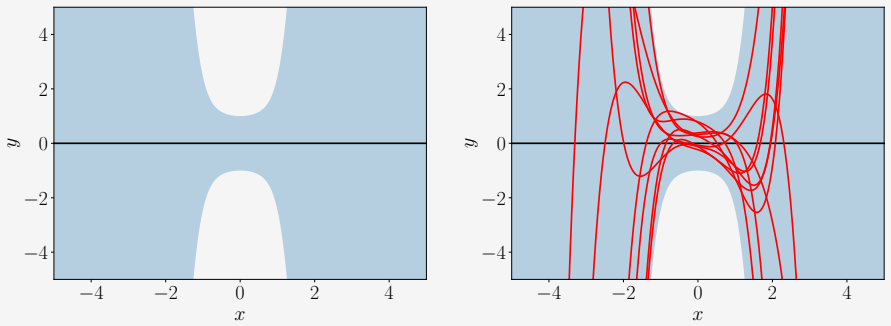
In our model, we chose a conjugate (Gaussian) prior on $\boldsymbol{\theta}$ so that the predictive distribution is Gaussian as well (and can be computed in closed form): With the prior distribution $p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{m}_0,\, \boldsymbol{S}_0\right)$, we obtain the predictive distribution as

$$p(y_* \mid \boldsymbol{x}_*) = \mathcal{N}\left(\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{m}_0,\, \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_0\boldsymbol{\phi}(\boldsymbol{x}_*) + \sigma^2\right), \tag{9.35}$$

where we used that (i) the prediction is Gaussian due to conjugacy and the marginalization property of Gaussians, (ii), the Gaussian noise is independent so that $\mathbb{V}[y_*] = \mathbb{V}[\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}] + \mathbb{V}[\epsilon]$, (iii) y_* is a linear transformation of $\boldsymbol{\theta}$ so that we can apply the rules for computing the mean and covariance of the prediction analytically by using (6.50) and (6.51), respectively.

In (9.35), the term $\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_0\boldsymbol{\phi}(\boldsymbol{x}_*)$ in the predictive variance explicitly accounts for the uncertainty associated with the parameters $\boldsymbol{\theta}$, whereas σ^2 is the uncertainty contribution due to the measurement noise.

Example 9.6 (Prior over Functions)
Let us consider a Bayesian linear regression problem with polynomials of degree 5. We choose a parameter prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \frac{1}{4}\boldsymbol{I})$. Figure 9.9 visualizes the distribution over functions induced by this parameter prior, including some function samples from this prior.



(a) Prior distribution over functions.

(b) Samples from the prior distribution over functions.

Figure 9.9 Prior over functions. (a) Distribution over functions represented by the mean function (black line) and the marginal uncertainties (shaded), representing the 95% confidence bounds; (b) Samples from the prior over functions, which are induced by the samples from the parameter prior.

So far, we looked at computing predictions using the parameter prior $p(\boldsymbol{\theta})$. However, when we have a parameter posterior (given some training data $\boldsymbol{X}, \boldsymbol{y}$), the same principles for prediction and inference hold as in (9.34) – we just need to replace the prior $p(\boldsymbol{\theta})$ with the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$. In the following, we will derive the posterior distribution in detail before using it to make predictions.

9.3.3 Posterior Distribution

Given a training set of inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and corresponding observations $y_n \in \mathbb{R}$, $n = 1, \ldots, N$, we compute the posterior over the parameters using Bayes' theorem as

$$p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{X})}, \tag{9.36}$$

where $\boldsymbol{X}$ is the collection of training inputs and $\boldsymbol{y}$ the collection of training targets. Furthermore, $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ the parameter prior and

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \tag{9.37}$$

marginal likelihood
evidence

the *marginal likelihood/evidence*, which is independent of the parameters $\boldsymbol{\theta}$ and ensures that the posterior is normalized, i.e., it integrates to 1. We can think of the marginal likelihood as the likelihood averaged over all possible parameter settings (with respect to the prior distribution $p(\boldsymbol{\theta})$).

In our specific model (9.32), the posterior (9.36) can be computed in closed form as

$$p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_N,\, \boldsymbol{S}_N\big)\,, \tag{9.38a}$$

$$\boldsymbol{S}_N = (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\,, \tag{9.38b}$$

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y})\,, \tag{9.38c}$$

where the subscript N indicates the size of the training set. In the following, we will detail how we arrive at this posterior.

Bayes' theorem tells us that the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$ is proportional to the product of the likelihood $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ and the prior $p(\boldsymbol{\theta})$:

$$\text{posterior} \qquad p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{X})} \tag{9.39a}$$

$$\text{likelihood} \qquad p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{\theta},\, \sigma^2\boldsymbol{I}\big) \tag{9.39b}$$

$$\text{prior} \qquad p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_0,\, \boldsymbol{S}_0\big) \tag{9.39c}$$

We will discuss two approaches to derive the desired posterior.

Approach 1: Linear Transformation of Gaussian Random Variables

Looking at the numerator of the posterior in (9.39a), we know that the Gaussian prior times the Gaussian likelihood (where the parameters on which we place the Gaussian appears linearly in the mean) is an (unnormalized) Gaussian (see Section 6.6.2). If necessary, we can find the normalizing constant using (6.114). If we want to compute that product by using the results from (6.112)–(6.113) in Section 6.6.2, we need to ensure the product has the "right" form, i.e.,

$$\mathcal{N}\big(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{\theta},\, \sigma^2\boldsymbol{I}\big)\mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_0,\, \boldsymbol{S}_0\big) = \mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{\mu},\, \boldsymbol{\Sigma}\big)\mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_0,\, \boldsymbol{S}_0\big) \tag{9.40}$$

for some $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. With this form we determine the desired product immediately as

$$\mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{\mu},\, \boldsymbol{\Sigma}\big)\mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_0,\, \boldsymbol{S}_0\big) \propto \mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \tag{9.41a}$$

$$\boldsymbol{S}_N = (\boldsymbol{S}_0^{-1} + \boldsymbol{\Sigma}^{-1})^{-1} \tag{9.41b}$$

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})\,. \tag{9.41c}$$

In order to get the "right" form, we need to turn $\mathcal{N}\big(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{\theta},\, \sigma^2\boldsymbol{I}\big)$ into $\mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{\mu},\, \boldsymbol{\Sigma}\big)$ for appropriate choices of $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. We will do this by using a linear transformation of Gaussian random variables (see Section 6.6), which allows us to exploit the property that linearly transformed Gaussian random variables are Gaussian distributed. More specifically, we will find $\boldsymbol{\mu} = \boldsymbol{B}\boldsymbol{y}$ and $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{B}\boldsymbol{B}^\top$ by linearly transforming the relationship $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\theta}$ in the likelihood into $\boldsymbol{B}\boldsymbol{y} = \boldsymbol{\theta}$ for a suitable $\boldsymbol{B}$. We obtain

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\theta} \overset{\times\boldsymbol{\Phi}^\top}{\iff} \boldsymbol{\Phi}^\top\boldsymbol{y} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\theta} \overset{\times(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}}{\iff} \underbrace{(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top}_{=:\boldsymbol{B}}\boldsymbol{y} = \boldsymbol{\theta} \tag{9.42}$$

Therefore, we can write $\boldsymbol{\theta} = \boldsymbol{B}\boldsymbol{y}$, and by using the rules for linear transformations of the mean and covariance from (6.50)–(6.51) we obtain

$$\mathcal{N}\big(\boldsymbol{\theta} \,|\, \boldsymbol{B}\boldsymbol{y},\, \sigma^2 \boldsymbol{B}\boldsymbol{B}^\top\big) = \mathcal{N}\big(\boldsymbol{\theta} \,|\, (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y},\, \sigma^2(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\big) \tag{9.43}$$

after some re-arranging of the terms for the covariance matrix.

If we now look at (9.43) and define its mean as $\boldsymbol{\mu}$ and covariance matrix as $\boldsymbol{\Sigma}$ in (9.41c) and (9.41b), respectively, we obtain the covariance $\boldsymbol{S}_N$ and the mean $\boldsymbol{m}_N$ of the parameter posterior $\mathcal{N}\big(\boldsymbol{\theta} \,|\, \boldsymbol{m}_N,\, \boldsymbol{S}_N\big)$ as

$$\boldsymbol{S}_N = (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\,, \tag{9.44a}$$

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \underbrace{\sigma^{-2}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})}_{\boldsymbol{\Sigma}^{-1}}\underbrace{(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y}}_{\boldsymbol{\mu}}) \tag{9.44b}$$

$$= \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y})\,, \tag{9.44c}$$

The posterior mean equals the MAP estimate.

respectively. Note that the posterior mean $\boldsymbol{m}_N$ equals the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$ from (9.31). This also makes sense since the posterior distribution is unimodal (Gaussian) with its maximum at the mean.

Remark. The posterior precision (inverse covariance)

$$\boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} \tag{9.45}$$

of the parameters $\boldsymbol{\theta}$ (see (9.44a)) contains two terms: $\boldsymbol{S}_0^{-1}$ is the prior precision and $\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is a data-dependent (precision) term. Both terms (matrices) are symmetric and positive definite. The data-dependent term $\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ grows as more data is taken into account. This means (at least) two things:

$\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ accumulates contributions from the data.

- The posterior precision grows as more and more data is taken into account; therefore, the covariance, and with it the uncertainty about the parameters, shrinks.
- The relative influence of the parameter prior vanishes for large N.

Therefore, for $N \to \infty$ the prior plays no role, and the parameter posterior tends to a point estimate, the MAP estimate. ♢

Approach 2: Completing the Squares

Instead of looking at the product of the prior and the likelihood, we can transform the problem into log-space and solve for the mean and covariance of the posterior by completing the squares.

The sum of the log-prior and the log-likelihood is

$$\log\mathcal{N}\big(\boldsymbol{y} \,|\, \boldsymbol{\Phi}\boldsymbol{\theta},\, \sigma^2\boldsymbol{I}\big) + \log\mathcal{N}\big(\boldsymbol{\theta} \,|\, \boldsymbol{m}_0,\, \boldsymbol{S}_0\big) \tag{9.46a}$$

$$= -\frac{1}{2}\big(\sigma^{-2}(\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\theta}) + (\boldsymbol{\theta}-\boldsymbol{m}_0)^\top\boldsymbol{S}_0^{-1}(\boldsymbol{\theta}-\boldsymbol{m}_0)\big) + \text{const} \tag{9.46b}$$

where the constant contains terms independent of $\boldsymbol{\theta}$. We will ignore the constant in the following. We now factorize (9.46b), which yields

$$\begin{aligned} &-\frac{1}{2}(\sigma^{-2}\boldsymbol{y}^\top\boldsymbol{y} - 2\sigma^{-2}\boldsymbol{y}^\top\boldsymbol{\Phi}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\boldsymbol{S}_0^{-1}\boldsymbol{\theta} \\ &- 2\boldsymbol{m}_0^\top\boldsymbol{S}_0^{-1}\boldsymbol{\theta} + \boldsymbol{m}_0^\top\boldsymbol{S}_0^{-1}\boldsymbol{m}_0) \end{aligned} \tag{9.47a}$$

$$= -\frac{1}{2}(\boldsymbol{\theta}^\top(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1})\boldsymbol{\theta} - 2(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)^\top\boldsymbol{\theta}) + \text{const}\,, \tag{9.47b}$$

where the constant contains the black terms in (9.47a), which are independent of $\boldsymbol{\theta}$. The orange terms are terms that are linear in $\boldsymbol{\theta}$, and the blue terms are the ones that are quadratic in $\boldsymbol{\theta}$. By inspecting (9.47b), we find that this equation is quadratic in $\boldsymbol{\theta}$. The fact that the unnormalized log-posterior distribution is a (negative) quadratic form implies that the posterior is Gaussian, i.e.,

$$p(\boldsymbol{\theta}\,|\,\boldsymbol{X},\boldsymbol{y}) = \exp(\log p(\boldsymbol{\theta}\,|\,\boldsymbol{X},\boldsymbol{y})) \propto \exp(\log p(\boldsymbol{y}\,|\,\boldsymbol{X},\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})) \tag{9.48a}$$

$$\propto \exp\Big(-\frac{1}{2}(\boldsymbol{\theta}^\top(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1})\boldsymbol{\theta} - 2(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)^\top\boldsymbol{\theta})\Big)\,, \tag{9.48b}$$

where we used (9.47b) in the last expression.

The remaining task is it to bring this (unnormalized) Gaussian into the form that is proportional to $\mathcal{N}(\boldsymbol{\theta}\,|\,\boldsymbol{m}_N,\,\boldsymbol{S}_N)$, i.e., we need to identify the mean $\boldsymbol{m}_N$ and the covariance matrix $\boldsymbol{S}_N$. To do this, we use the concept of *completing the squares*. The desired log-posterior is

completing the squares

$$\log\mathcal{N}(\boldsymbol{\theta}\,|\,\boldsymbol{m}_N,\,\boldsymbol{S}_N) = -\frac{1}{2}((\boldsymbol{\theta}-\boldsymbol{m}_N)^\top\boldsymbol{S}_N^{-1}(\boldsymbol{\theta}-\boldsymbol{m}_N)) + \text{const} \tag{9.49a}$$

$$= -\frac{1}{2}(\boldsymbol{\theta}^\top\boldsymbol{S}_N^{-1}\boldsymbol{\theta} - 2\boldsymbol{m}_N^\top\boldsymbol{S}_N^{-1}\boldsymbol{\theta} + \boldsymbol{m}_N^\top\boldsymbol{S}_N^{-1}\boldsymbol{m}_N)\,. \tag{9.49b}$$

Here, we factorized the quadratic form $(\boldsymbol{\theta}-\boldsymbol{m}_N)^\top\boldsymbol{S}_N^{-1}(\boldsymbol{\theta}-\boldsymbol{m}_N)$ into a term that is quadratic in $\boldsymbol{\theta}$ alone (blue), a term that is linear in $\boldsymbol{\theta}$ (orange), and a constant term (black). This allows us now to find $\boldsymbol{S}_N$ and $\boldsymbol{m}_N$ by matching the colored expressions in (9.47b) and (9.49b), which yields

$$\boldsymbol{S}_N^{-1} = \boldsymbol{\Phi}^\top\sigma^{-2}\boldsymbol{I}\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1} \iff \boldsymbol{S}_N = (\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1})^{-1}\,, \tag{9.50}$$

$$\boldsymbol{m}_N^\top\boldsymbol{S}_N^{-1} = (\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)^\top \iff \boldsymbol{m}_N = \boldsymbol{S}_N(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)\,. \tag{9.51}$$

This is identical to the solution in (9.44a)–(9.44c), which we obtained by linear transformations of Gaussian random variables.

Remark (Completing the Squares—General Approach). If we are given an equation

$$\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{a}^\top \boldsymbol{x} + \text{const}_1\,, \tag{9.52}$$

where $\boldsymbol{A}$ is symmetric and positive definite, which we wish to bring into the form

$$(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}(\boldsymbol{x}-\boldsymbol{\mu}) + \text{const}_2\,, \tag{9.53}$$

we can do this by setting

$$\boldsymbol{\Sigma} := \boldsymbol{A}\,, \tag{9.54}$$

$$\boldsymbol{\mu} := \boldsymbol{\Sigma}^{-1}\boldsymbol{a} \tag{9.55}$$

and $\text{const}_2 = \text{const}_1 - \boldsymbol{\mu}^\top \boldsymbol{\Sigma}\boldsymbol{\mu}$. ♢

We can see that the terms inside the exponential in (9.48b) are of the form (9.52) with

$$\boldsymbol{A} := \sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{S}_0^{-1}\,, \tag{9.56}$$

$$\boldsymbol{a} := \sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0\,. \tag{9.57}$$

Since $\boldsymbol{A}, \boldsymbol{a}$ can be difficult to identify in equations like (9.47a), it is often helpful to bring these equations into the form (9.52) that decouples quadratic term, linear terms and constants, which simplifies finding the desired solution.

9.3.4 Posterior Predictions

In (9.34), we computed the predictive distribution of y_* at a test input $\boldsymbol{x}_*$ using the parameter prior $p(\boldsymbol{\theta})$. In principle, predicting with the parameter posterior $p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$ is not fundamentally different given that in our conjugate model the prior and posterior are both Gaussian (with different parameters). Therefore, by following the same reasoning as in Section 9.3.2 we obtain the (posterior) predictive distribution

$$p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*) = \int p(y_* \mid \boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{\theta} \tag{9.58a}$$

$$= \int \mathcal{N}\big(y_* \mid \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta},\, \sigma^2\big)\mathcal{N}\big(\boldsymbol{\theta} \mid \boldsymbol{m}_N,\, \boldsymbol{S}_N\big)\mathrm{d}\boldsymbol{\theta} \tag{9.58b}$$

$$= \mathcal{N}\big(y_* \mid \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{m}_N,\, \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*) + \sigma^2\big) \tag{9.58c}$$

The term $\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*)$ reflects the posterior uncertainty associated with the parameters $\boldsymbol{\theta}$. Note that $\boldsymbol{S}_N$ depends on the training inputs $\boldsymbol{X}$, see (9.44a). The predictive mean coincides with the MAP estimate.

Remark (Mean and Variance of Noise-Free Function Values). In many cases, we are not interested in the predictive distribution $p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*)$ of a (noisy) observation. Instead, we would like to obtain the distribution

of the (noise-free) latent function values $f(\boldsymbol{x}_*) = \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}$. We determine the corresponding moments by exploiting the properties of means and variances, which yields

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{x}_*) \mid \boldsymbol{X}, \boldsymbol{y}] &= \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}] = \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}] \\ &= \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{m}_N = \boldsymbol{m}_N^\top\boldsymbol{\phi}(\boldsymbol{x}_*)\,, \end{aligned} \tag{9.59}$$

$$\begin{aligned}\mathbb{V}_{\boldsymbol{\theta}}[f(\boldsymbol{x}_*) \mid \boldsymbol{X}, \boldsymbol{y}] &= \mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}] \\ &= \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}]\boldsymbol{\phi}(\boldsymbol{x}_*) \\ &= \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*) \end{aligned} \tag{9.60}$$

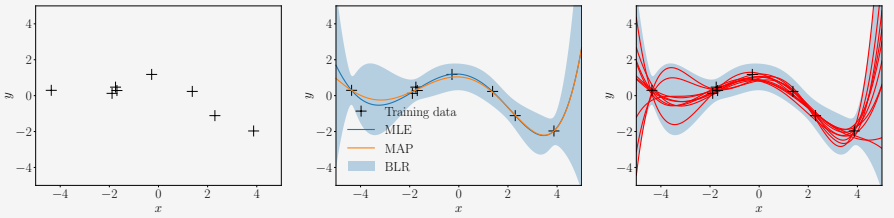
We see that the predictive mean is the same as the predictive mean for noisy observations as the noise has mean 0, and the predictive variance only differs by σ^2, which is the variance of the measurement noise: When we predict noisy function values, we need to include σ^2 as a source of uncertainty, but this term is not needed for noise-free predictions. Here, the only remaining uncertainty stems from the parameter posterior. ♢

Integrating out parameters induces a distribution over functions.

Remark (Distribution over Functions). The fact that we integrate out the parameters $\boldsymbol{\theta}$ induces a distribution over functions: If we sample $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y})$ from the parameter posterior, we obtain a single function realization $\boldsymbol{\theta}_i^\top\boldsymbol{\phi}(\cdot)$. The *mean function*, i.e., the set of all expected function values $\mathbb{E}_{\boldsymbol{\theta}}[f(\cdot) \mid \boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}]$, of this distribution over functions is $\boldsymbol{m}_N^\top\boldsymbol{\phi}(\cdot)$. The (marginal) variance, i.e., the variance of the function $f(\cdot)$, are given by $\boldsymbol{\phi}^\top(\cdot)\boldsymbol{S}_N\boldsymbol{\phi}(\cdot)$. ♢

mean function

Example 9.7 (Posterior over Functions)



(a) Training data.

(b) Posterior over functions represented by the marginal uncertainties (shaded) showing the 95% predictive confidence bounds, the maximum likelihood estimate (MLE) and the MAP estimate (MAP), which is identical to the posterior mean function.

(c) Samples from the posterior over functions, which are induced by the samples from the parameter posterior.

Figure 9.10 Bayesian linear regression and posterior over functions. (a) Training data; (b) posterior distribution over functions; (c) Samples from the posterior over functions.

Let us revisit the Bayesian linear regression problem with polynomials of degree 5. We choose a parameter prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \frac{1}{4}\boldsymbol{I})$. Figure 9.9

visualizes the prior over functions induced by the parameter prior and sample functions from this prior.

Figure 9.10 shows the posterior over functions that we obtain via Bayesian linear regression. The training dataset is shown in Figure 9.11(a); Figure 9.11(b) shows the posterior distribution over functions, including the functions we would obtain via maximum likelihood and MAP estimation. The function we obtain using the MAP estimate also corresponds to the posterior mean function in the Bayesian linear regression setting. Figure 9.11(c) shows some plausible realizations (samples) of functions under that posterior over functions.

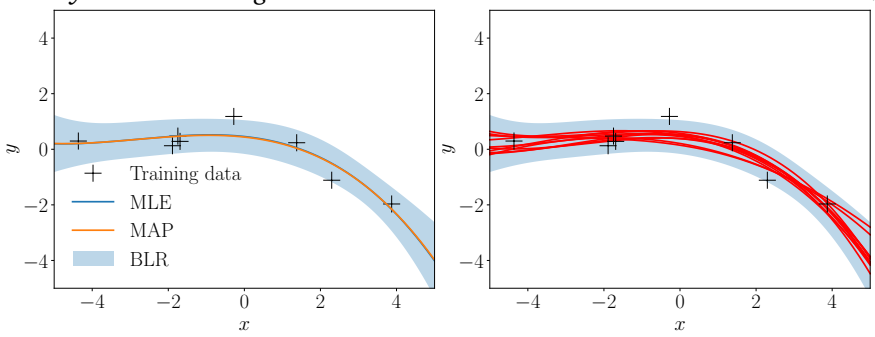
Figure 9.11 shows some examples of the posterior distribution over functions induced by the parameter posterior. For different polynomial degrees M the left panels show the maximum likelihood estimate, the MAP estimate (which is identical to the posterior mean function) and the 95% predictive confidence bounds, represented by the shaded area. The right panels show samples from the posterior over functions: Here, we sampled parameters $\boldsymbol{\theta}_i$ from the parameter posterior and computed the function $\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}_i$, which is a single realization of a function under the posterior distribution over functions. For low-order polynomials, the parameter posterior does not allow the parameters to vary much: The sampled functions are nearly identical. When we make the model more flexible by adding more parameters (i.e., we end up with a higher-order polynomial), these parameters are not sufficiently constrained by the posterior, and the sampled functions can be easily visually separated. We also see in the corresponding panels on the left how the uncertainty increases, especially at the boundaries. Although for a 7th-order polynomial the MAP estimate yields a reasonable fit, the Bayesian linear regression model additionally tells us that the posterior uncertainty is huge. This information can be critical when we use these predictions in a decision-making system, where bad decisions can have significant consequences (e.g., in reinforcement learning or robotics).

9.3.5 Computing the Marginal Likelihood

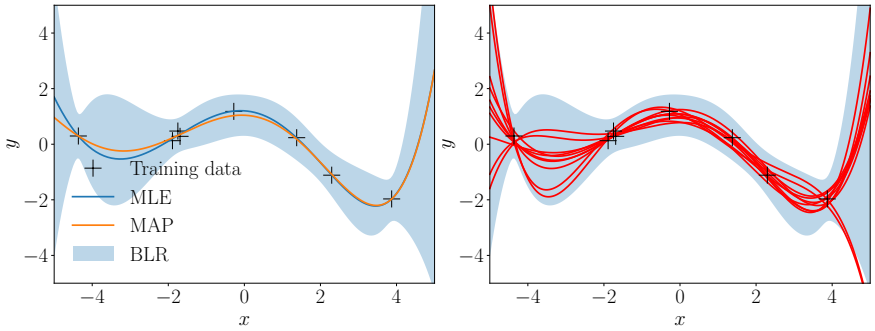
In Section 8.5.2, we highlighted the importance of the marginal likelihood for Bayesian model selection. In the following, we compute the marginal likelihood for Bayesian linear regression with a conjugate Gaussian prior on the parameters, i.e., exactly the setting we have been discussing in this chapter. Just to re-cap, we consider the following generative process:

$$\boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{m}_0, \boldsymbol{S}_0\right) \tag{9.61a}$$

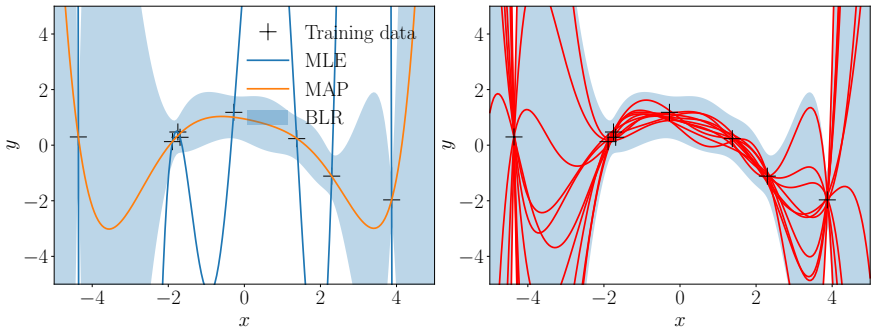
$$y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right), \tag{9.61b}$$



(a) Posterior distribution for polynomials of degree $M = 3$ (left) and samples from the posterior over functions (right).



(b) Posterior distribution for polynomials of degree $M = 5$ (left) and samples from the posterior over functions (right).



(c) Posterior distribution for polynomials of degree $M = 7$ (left) and samples from the posterior over functions (right).

Figure 9.11 Bayesian linear regression. Left panels: Shaded areas indicate the 95% predictive confidence bounds. The mean of the Bayesian linear regression model coincides with the MAP estimate. The predictive uncertainty is the sum of the noise term and the posterior parameter uncertainty, which depends on the location of the test input. Right panels: Sampled functions from the posterior distribution.

$n = 1, \ldots, N$. The marginal likelihood is given by

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \tag{9.62a}$$

$$= \int \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{\theta},\, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{\theta} \mid \boldsymbol{m}_0,\, \boldsymbol{S}_0\right) \mathrm{d}\boldsymbol{\theta}\,, \tag{9.62b}$$

The marginal likelihood can be interpreted as the expected likelihood under the prior, i.e., $\mathbb{E}_{\boldsymbol{\theta}}[p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})]$.

where we integrate out the model parameters $\boldsymbol{\theta}$. We compute the marginal likelihood in two steps: First, we show that the marginal likelihood is

Gaussian (as a distribution in $\boldsymbol{y}$); Second, we compute the mean and covariance of this Gaussian.

1. The marginal likelihood is Gaussian: From Section 6.6.2 we know that (i) the product of two Gaussian random variables is an (unnormalized) Gaussian distribution, (ii) a linear transformation of a Gaussian random variable is Gaussian distributed. In (9.62b), we require a linear transformation to bring $\mathcal{N}\big(\boldsymbol{y} \,|\, \boldsymbol{X}\boldsymbol{\theta},\, \sigma^2\boldsymbol{I}\big)$ into the form $\mathcal{N}\big(\boldsymbol{\theta} \,|\, \boldsymbol{\mu},\, \boldsymbol{\Sigma}\big)$ for some $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. Once this is done, the integral can be solved in closed form. The result is the normalizing constant of the product of the two Gaussians. The normalizing constant itself has Gaussian shape, see (6.114).
2. Mean and covariance. We compute the mean and covariance matrix of the marginal likelihood by exploiting the standard results for means and covariances of affine transformations of random variables, see Section 6.4.4. The mean of the marginal likelihood is computed as

$$\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{y} \,|\, \boldsymbol{X}] = \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}] = \boldsymbol{X}\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta}] = \boldsymbol{X}\boldsymbol{m}_0\,. \tag{9.63}$$

Note that $\boldsymbol{\epsilon} \sim \mathcal{N}\big(\boldsymbol{0},\, \sigma^2\boldsymbol{I}\big)$ is a vector of i.i.d. random variables. The covariance matrix is given as

$$\begin{aligned}\mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{y}] &= \mathrm{Cov}[\boldsymbol{X}\boldsymbol{\theta}] + \sigma^2\boldsymbol{I} = \boldsymbol{X}\,\mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{\theta}]\boldsymbol{X}^\top + \sigma^2\boldsymbol{I} && \text{(9.64a)}\\ &= \boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2\boldsymbol{I} && \text{(9.64b)}\end{aligned}$$

Hence, the marginal likelihood is

$$\begin{aligned}p(\boldsymbol{y} \,|\, \boldsymbol{X}) = (2\pi)^{-\frac{N}{2}}\det(\boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2\boldsymbol{I})^{-\frac{1}{2}}\\ \times \exp\big(-\tfrac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{m}_0)^\top(\boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{m}_0)\big)\,.\end{aligned} \tag{9.65}$$

The marginal likelihood can now be used for Bayesian model selection as discussed in Section 8.5.2.

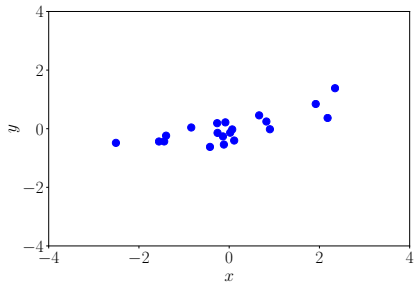
9.4 Maximum Likelihood as Orthogonal Projection

Having crunched through much algebra to derive maximum likelihood and MAP estimates, we will now provide a geometric interpretation of maximum likelihood estimation. Let us consider a simple linear regression setting

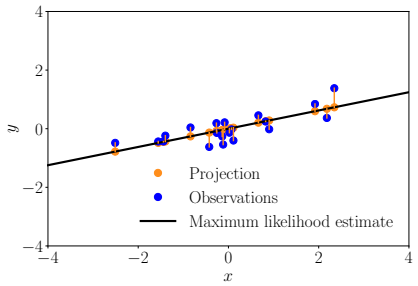
$$y = x\theta + \epsilon, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)\,, \tag{9.66}$$

in which we consider linear functions $f : \mathbb{R} \to \mathbb{R}$ that go through the origin (we omit features here for clarity). The parameter θ determines the slope of the line. Figure 9.12(a) shows a one-dimensional dataset.

With a training data set $\boldsymbol{X} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$, $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in$



(a) Regression dataset consisting of noisy observations y_n (blue) of function values $f(x_n)$ at input locations x_n.



(b) The orange dots are the projections of the noisy observations (blue dots) onto the line $\theta_{\text{ML}}x$. The maximum likelihood solution to a linear regression problem finds a subspace (line) onto which the overall projection error (orange lines) of the observations is minimized.

Figure 9.12 Geometric interpretation of least squares. (a) Dataset; (b) Maximum likelihood solution interpreted as a projection.

$\mathbb{R}^N$, we recall the results from Section 9.2.1 and obtain the maximum likelihood estimator for the slope parameter as

$$\theta_{\text{ML}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y} = \frac{\boldsymbol{X}^\top \boldsymbol{y}}{\boldsymbol{X}^\top \boldsymbol{X}} \in \mathbb{R}\,. \tag{9.67}$$

This means for the training inputs $\boldsymbol{X}$ we obtain the optimal (maximum likelihood) reconstruction of the training data, i.e., the approximation with the minimum least-squares error

$$\boldsymbol{X}\theta_{\text{ML}} = \boldsymbol{X}\frac{\boldsymbol{X}^\top \boldsymbol{y}}{\boldsymbol{X}^\top \boldsymbol{X}} = \frac{\boldsymbol{X}\boldsymbol{X}^\top}{\boldsymbol{X}^\top \boldsymbol{X}}\boldsymbol{y}\,. \tag{9.68}$$

Linear regression can be thought of as a method for solving systems of linear equations.

Maximum likelihood linear regression performs an orthogonal projection.

As we are basically looking for a solution of $\boldsymbol{y} = \boldsymbol{X}\theta$, we can think of linear regression as a problem for solving systems of linear equations. Therefore, we can relate to concepts from linear algebra and analytic geometry that we discussed in Chapters 2 and 3. In particular, looking carefully at (9.68) we see that the maximum likelihood estimator θ_{ML} in our example from (9.66) effectively does an orthogonal projection of $\boldsymbol{y}$ onto the one-dimensional subspace spanned by $\boldsymbol{X}$. Recalling the results on orthogonal projections from Section 3.7, we identify $\frac{\boldsymbol{X}\boldsymbol{X}^\top}{\boldsymbol{X}^\top \boldsymbol{X}}$ as the projection matrix, θ_{ML} as the coordinates of the projection onto the one-dimensional subspace of $\mathbb{R}^N$ spanned by $\boldsymbol{X}$ and $\boldsymbol{X}\theta_{\text{ML}}$ as the orthogonal projection of $\boldsymbol{y}$ onto this subspace.

Therefore, the maximum likelihood solution provides also a geometrically optimal solution by finding the vectors in the subspace spanned by $\boldsymbol{X}$ that are "closest" to the corresponding observations $\boldsymbol{y}$, where "closest" means the smallest (squared) distance of the function values y_n to $x_n\theta$. This is achieved by orthogonal projections. Figure 9.12(b) shows the orthogonal projection of the noisy observations onto the subspace that

minimizes the squared distance between the original dataset and its projection, which corresponds to the maximum likelihood solution.

In the general linear regression case where

$$y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}\left(0, \sigma^2\right) \tag{9.69}$$

with vector-valued features $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^K$, we again can interpret the maximum likelihood result

$$\boldsymbol{y} \approx \boldsymbol{\Phi}\boldsymbol{\theta}_{\text{ML}}\,, \tag{9.70}$$

$$\boldsymbol{\theta}_{\text{ML}} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} \tag{9.71}$$

as a projection onto a K-dimensional subspace of $\mathbb{R}^N$, which is spanned by the columns of the feature matrix $\boldsymbol{\Phi}$, see Section 3.7.2.

If the feature functions ϕ_k that we use to construct the feature matrix $\boldsymbol{\Phi}$ are orthonormal (see Section 3.6), we obtain a special case where the columns of $\boldsymbol{\Phi}$ form an orthonormal basis (see Section 3.5), such that $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} = \boldsymbol{I}$. This will then lead to the projection

$$\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{y} = \left(\sum_{k=1}^{K}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^\top\right)\boldsymbol{y} \tag{9.72}$$

so that the coupling between different features has disappeared and the maximum likelihood projection is simply the sum of projections of $\boldsymbol{y}$ onto the individual basis vectors $\boldsymbol{\phi}_k$, i.e., the columns of $\boldsymbol{\Phi}$. Many popular basis functions in signal processing, such as wavelets and Fourier bases, are orthogonal basis functions. When the basis is not orthogonal, one can convert a set of linearly independent basis functions to an orthogonal basis by using the Gram-Schmidt process (Strang, 2003).

9.5 Further Reading

In this chapter, we discussed linear regression for Gaussian likelihoods and conjugate Gaussian priors on the parameters of the model. This allowed for closed-form Bayesian inference. However, in some applications we may want to choose a different likelihood function. For example, in a binary *classification* setting, we observe only two possible (categorical) outcomes, and a Gaussian likelihood is inappropriate in this setting. Instead, we can choose a Bernoulli likelihood that will return a probability of the predicted label to be 1 (or 0). We refer to the books by Bishop (2006); Murphy (2012); Barber (2012) for an in-depth introduction to classification problems. A different example where non-Gaussian likelihoods are important is count data. Counts are non-negative integers, and in this case a Binomial or Poisson likelihood would be a better choice than a Gaussian. All these examples fall into the category of *generalized linear models*, a flexible generalization of linear regression that allows for response variables that have error distribution models other than a Gaussian

classification

generalized linear models

distribution. The GLM generalizes linear regression by allowing the linear model to be related to the observed values via a smooth and invertible function $\sigma(\cdot)$ that may be nonlinear so that $y = \sigma(f)$, where $f = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x})$ is the linear regression model from (9.12). We can therefore think of a generalized linear model in terms of function composition $y = \sigma \circ f$ where f is a linear regression model and σ the activation function. Note, that although we are talking about "generalized linear models" the outputs y are no longer linear in the parameters $\boldsymbol{\theta}$. In *logistic regression*, we choose the *logistic sigmoid* $\sigma(f) = \frac{1}{1+\exp(-f)} \in [0, 1]$, which can be interpreted as the probability of observing a binary output $y = 1$ of a Bernoulli random variable. The function $\sigma(\cdot)$ is called *transfer function* or *activation function*, its inverse is called the *canonical link function*. From this perspective, it is also clear that generalized linear models are the building blocks of (deep) feedforward neural networks: If we consider a generalized linear model $\boldsymbol{y} = \sigma(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{A}$ is a weight matrix and $\boldsymbol{b}$ a bias vector, we identify this generalized linear model as a single-layer neural network with activation function $\sigma(\cdot)$. We can now recursively compose these functions via

logistic regression

logistic sigmoid

transfer function

activation function

canonical link function

For ordinary linear regression the activation function would simply be the identity.

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}_k) \\ \boldsymbol{f}_k(\boldsymbol{x}_k) &= \sigma_k(\boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{b}_k) \end{aligned} \tag{9.73}$$

Generalized linear models are the building blocks of deep neural networks.

for $k = 0, \ldots, K-1$ where $\boldsymbol{x}_0$ are the input features and $\boldsymbol{x}_K = \boldsymbol{y}$ are the observed outputs, such that $\boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_0$ is a K-layer deep neural network. Therefore, the building blocks of this deep neural network are the generalized linear models defined in (9.73). A great post on the relation between GLMs and deep networks is available at `https://tinyurl.com/glm-dnn`. Neural networks (Bishop, 1995; Goodfellow et al., 2016) are significantly more expressive and flexible than linear regression models. However, maximum likelihood parameter estimation is a non-convex optimization problem, and marginalization of the parameters in a fully Bayesian setting is analytically intractable.

We briefly hinted at the fact that a distribution over parameters induces a distribution over regression functions. *Gaussian processes* (Rasmussen and Williams, 2006) are regression models where the concept of a distribution over function is central. Instead of placing a distribution over parameters a Gaussian process places a distribution directly on the space of functions without the "detour" via the parameters. To do so, the Gaussian process exploits the *kernel trick* (Schölkopf and Smola, 2002), which allows us to compute inner products between two function values $f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)$ only by looking at the corresponding input $\boldsymbol{x}_i, \boldsymbol{x}_j$. A Gaussian process is closely related to both Bayesian linear regression and support vector regression but can also be interpreted as a Bayesian neural network with a single hidden layer where the number of units tends to infinity (Neal, 1996; Williams, 1997). An excellent introduction to Gaus-

Gaussian processes

kernel trick

sian processes can be found in (MacKay, 1998; Rasmussen and Williams, 2006).

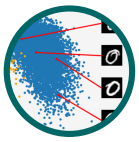
We focused on Gaussian parameter priors in the discussions in this chapters because they allow for closed-form inference in linear regression models. However, even in a regression setting with Gaussian likelihoods we may choose a non-Gaussian prior. Consider a setting where the inputs are $\boldsymbol{x} \in \mathbb{R}^D$ and our training set is small and of size $N \ll D$. This means that the regression problem is under-determined. In this case, we can choose a parameter prior that enforces sparsity, i.e., a prior that tries to set as many parameters to 0 as possible (*variable selection*). This prior provides a stronger regularizer than the Gaussian prior, which often leads to an increased prediction accuracy and interpretability of the model. The Laplace prior is one example that is frequently used for this purpose. A linear regression model with the Laplace prior on the parameters is equivalent to linear regression with L1 regularization (*LASSO*) (Tibshirani, 1996). The Laplace distribution is sharply peaked at zero (its first derivative is discontinuous) and it concentrates its probability mass closer to zero than the Gaussian distribution, which encourages parameters to be 0. Therefore, the non-zero parameters are relevant for the regression problem, which is the reason why we also speak of "variable selection".

variable selection

LASSO

10

Dimensionality Reduction with Principal Component Analysis



A 640×480 pixels color image is a data point in a million-dimensional space, where every pixel responds to three dimensions, one for each color channel (red, green, blue).

Working directly with high-dimensional data, such as images, comes with some difficulties: it is hard to analyze, interpretation is difficult, visualization is nearly impossible, and (from a practical point of view) storage of the data vectors can be expensive. However, high-dimensional data often has properties that we can exploit. For example, high-dimensional data is often overcomplete, i.e., many dimensions are redundant and can be explained by a combination of other dimensions. Furthermore, dimensions in high-dimensional data are often correlated so that the data possesses an intrinsic lower-dimensional structure. Dimensionality reduction exploits structure and correlation and allows us to work with a more compact representation of the data, ideally without losing information. We can think of dimensionality reduction as a compression technique, similar to jpeg or mp3, which are compression algorithms for images and music.

In this chapter, we will discuss *principal component analysis* (PCA), an algorithm for linear *dimensionality reduction*. PCA, proposed by Pearson (1901b) and Hotelling (1933), has been around for more than 100 years and is still one of the most commonly used techniques for data compression and data visualization. It is also used for the identification of simple patterns, latent factors and structures of high-dimensional data. In the signal processing community, PCA is also known as the *Karhunen-Loève transform*. In this chapter, we derive PCA from first principles, drawing on our understanding of basis and basis change (see Sections 2.6.1 and 2.7.2), projections (see Section 3.7), eigenvalues (see Section 4.2), Gaussian distributions (see Section 6.5) and constrained optimization (see Section 7.2).

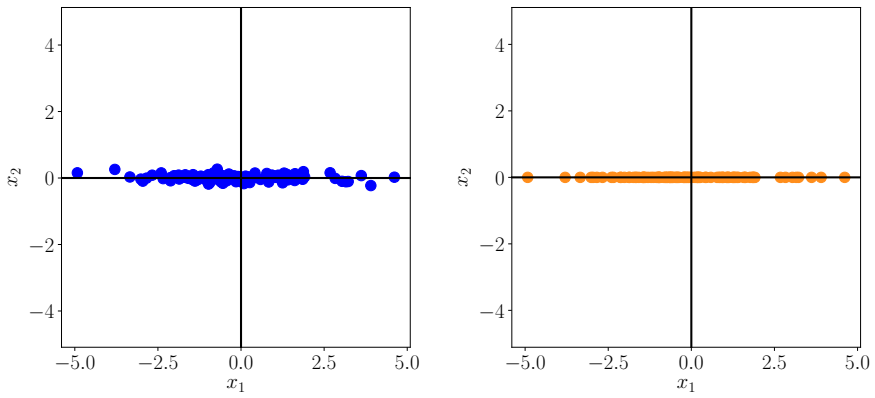
principal component analysis

dimensionality reduction

Karhunen-Loève transform

Dimensionality reduction generally exploits a property of high-dimensional data (e.g., images) that it often lies on a low-dimensional subspace, and that many dimensions are highly correlated, redundant or contain little information. Figure 10.1 gives an illustrative example in two dimensions. Although the data in Figure 10.1(a) does not quite lie on a line, the data does not vary much in the x_2-direction, so that we can express it as if it was on a line – with nearly no loss, see Figure 10.1(b). To describe the data in Figure 10.1(b), only the x_1-coordinate is required, and the data lies in a one-dimensional subspace of $\mathbb{R}^2$.

In the context of Table 1.1, the problem of dimensionality reduction falls



(a) Dataset with x_1 and x_2 coordinates.

(b) Compressed dataset where only the x_1 coordinate is relevant.

Figure 10.1 Illustration: Dimensionality reduction. (a) The original dataset does not vary much along the x_2 direction. (b) The data from (a) can be represented using the x_1-coordinate alone with nearly no loss.

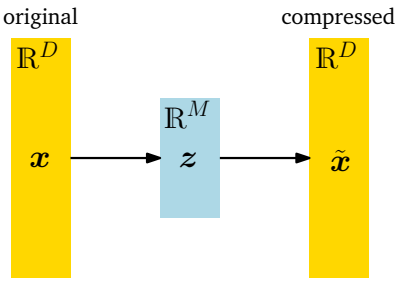


Figure 10.2 Graphical illustration of PCA. In PCA, we find a compressed version $\tilde{\boldsymbol{x}}$ of original data $\boldsymbol{x}$ that has an intrinsic lower-dimensional representation $\boldsymbol{z}$.

into the category of an unsupervised learning problem with continuous latent variables.

10.1 Problem Setting

In PCA, we are interested in finding projections $\tilde{\boldsymbol{x}}_n$ of data points $\boldsymbol{x}_n$ that are as similar to the original data points as possible, but which have a significantly lower intrinsic dimensionality. Figure 10.1 gives an illustration what this could look like.

More concretely, we consider an i.i.d. dataset $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^D$, with mean $\mathbf{0}$ that possesses the *data covariance matrix*

data covariance matrix

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top . \tag{10.1}$$

Furthermore, we assume there exists a low-dimensional compressed representation (code)

$$\boldsymbol{z}_n = \boldsymbol{B}^\top \boldsymbol{x}_n \in \mathbb{R}^M \tag{10.2}$$

of $\boldsymbol{x}_n$, where we define the projection matrix

$$\boldsymbol{B} := [\boldsymbol{b}_1, \dots, \boldsymbol{b}_M] \in \mathbb{R}^{D \times M} . \tag{10.3}$$

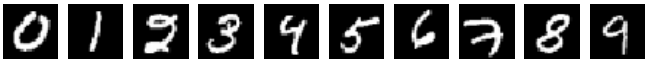


Figure 10.3 Examples of handwritten digits from the MNIST dataset. `http://yann.lecun.com/exdb/mnist/`

The columns $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ of $\boldsymbol{B}$ form a basis of the M-dimensional subspace in which the projected data $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x} \in \mathbb{R}^D$ live.

We assume that the columns of $\boldsymbol{B}$ are orthonormal so that $\boldsymbol{b}_i^\top \boldsymbol{b}_j = 0$ if and only if $i \neq j$. We seek an M-dimensional subspace $U \subseteq \mathbb{R}^D$, $\dim(U) = M < D$ onto which we project the data. We denote the projected data by $\tilde{\boldsymbol{x}}_n \in U$, and their coordinates (with respect to the basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ of U) by $\boldsymbol{z}_n$. Our aim is to find projections $\tilde{\boldsymbol{x}}_n \in \mathbb{R}^D$ (or equivalently the codes $\boldsymbol{z}_n$ and the basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$) so that they are as similar to the original data $\boldsymbol{x}_n$ and minimize the loss due to compression.

In Section 10.2, we will find low-dimensional representations that retain as much information as possible and minimize the compression loss. An alternative derivation of PCA is given in Section 10.3, we will be looking at minimizing the squared reconstruction error $\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2$ between the original data $\boldsymbol{x}_n$ and its projection $\tilde{\boldsymbol{x}}_n$.

Figure 10.2 illustrates the setting we consider in PCA, where $\boldsymbol{z}$ represents the intrinsic lower dimension of the compressed data $\tilde{\boldsymbol{x}}$ and plays the role of a bottleneck, which controls how much information can flow between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. In PCA, we consider a linear relationship between the original data $\boldsymbol{x}$ and its low-dimensional code $\boldsymbol{z}$ so that $\boldsymbol{z} = \boldsymbol{B}^\top \boldsymbol{x}$ and $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z}$ for a suitable matrix $\boldsymbol{B}$.

Example 10.1 (Coordinate Representation/Code)

Consider $\mathbb{R}^2$ with the canonical basis $\boldsymbol{e}_1 = [1, 0]^\top$, $\boldsymbol{e}_2 = [0, 1]^\top$. From Chapter 2 we know that $\boldsymbol{x} \in \mathbb{R}^2$ can be represented as a linear combination of these basis vectors, e.g.,

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} = 5\boldsymbol{e}_1 + 3\boldsymbol{e}_2 \,. \tag{10.4}$$

However, when we consider vectors of the form

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} 0 \\ z \end{bmatrix} \in \mathbb{R}^2 \,, \quad z \in \mathbb{R} \,, \tag{10.5}$$

they can always be written as $0\boldsymbol{e}_1 + z\boldsymbol{e}_2$. To represent these vectors it is sufficient to remember/store the *coordinate/code* z of $\tilde{\boldsymbol{x}}$ with respect to the $\boldsymbol{e}_2$ vector.

More precisely, the set of $\tilde{\boldsymbol{x}}$ vectors (with the standard vector addition and scalar multiplication) forms a vector subspace U (see Section 2.4) with $\dim(U) = 1$ because $U = \text{span}[\boldsymbol{e}_2]$.

The dimension of a vector space corresponds to the number of its basis vectors (see Section 2.6.1).

Throughout this chapter, we will use the MNIST digits dataset as a re-

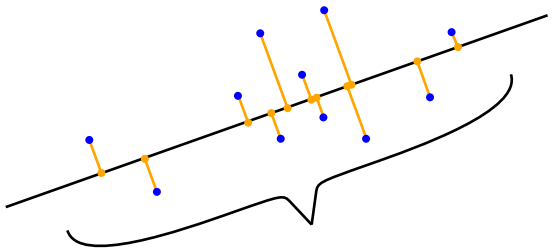


Figure 10.4 PCA finds a lower-dimensional subspace (line) that maintains as much variance (spread of the data) as possible when the data (blue) is projected onto this subspace (orange).

occurring example, which contains $60,000$ examples of handwritten digits 0–9. Each digit is a grayscale image of size 28×28, i.e., it contains 784 pixels so that we can interpret every image in this dataset as a vector $\boldsymbol{x} \in \mathbb{R}^{784}$. Examples of these digits are shown in Figure 10.3.

10.2 Maximum Variance Perspective

Figure 10.1 gave an example of how a two-dimensional dataset can be represented using a single coordinate. In Figure 10.1(b), we chose to ignore the x_2-coordinate of the data because it did not add too much information so that the compressed data is similar to the original data in Figure 10.1(a). We could have chosen to ignore the x_1-coordinate, but then the compressed data had been very dissimilar from the original data, and much information in the data would have been lost.

If we interpret information content in the data as how "space filling" the data set is, then we can describe the information contained in the data by looking at the spread of the data. From Section 6.4.1 we know that the variance is an indicator of the spread of the data, and we can derive PCA as a dimensionality reduction algorithm that maximizes the variance in the low-dimensional representation of the data to retain as much information as possible. Figure 10.4 illustrates this.

Considering the setting discussed in Section 10.1, our aim is to find a matrix $\boldsymbol{B}$ (see (10.3)) that retains as much information as possible when compressing data by projecting it onto the subspace spanned by the columns $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ of $\boldsymbol{B}$. Retaining most information after data compression is equivalent to capturing the largest amount of variance in the low-dimensional code (Hotelling, 1933).

Remark. (Centered Data) For the data covariance matrix in (10.1) we assumed centered data. We can make this assumption without loss of generality: Let us assume that $\boldsymbol{\mu}$ is the mean of the data. Using the properties of the variance, which we discussed in Section 6.4.3 we obtain

$$\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{z}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{B}^\top(\boldsymbol{x} - \boldsymbol{\mu})] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{B}^\top\boldsymbol{x} - \boldsymbol{B}^\top\boldsymbol{\mu}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{B}^\top\boldsymbol{x}]\,, \qquad (10.6)$$

i.e., the variance of the low-dimensional code does not depend on the mean of the data. Therefore, we assume without loss of generality that the

data has mean $\mathbf{0}$ for the remainder of this section. With this assumption the mean of the low-dimensional code is also $\mathbf{0}$ since $\mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}] = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{B}^\top \boldsymbol{x}] = \boldsymbol{B}^\top \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \mathbf{0}$. ♢

10.2.1 Direction with Maximal Variance

We maximize the variance of the low-dimensional code using a sequential approach. We start by seeking a single vector $\boldsymbol{b}_1 \in \mathbb{R}^D$ that maximizes the variance of the projected data, i.e., we aim to maximize the variance of the first coordinate z_1 of $\boldsymbol{z} \in \mathbb{R}^M$ so that

The vector $\boldsymbol{b}_1$ will be the first column of the matrix $\boldsymbol{B}$ and therefore the first of M orthonormal basis vectors that span the lower-dimensional subspace.

$$V_1 := \mathbb{V}[z_1] = \frac{1}{N}\sum_{n=1}^{N} z_{1n}^2 \tag{10.7}$$

is maximized, where we exploited the i.i.d. assumption of the data and defined z_{1n} as the first coordinate of the low-dimensional representation $\boldsymbol{z}_n \in \mathbb{R}^M$ of $\boldsymbol{x}_n \in \mathbb{R}^D$. Note that first component of $\boldsymbol{z}_n$ is given by

$$z_{1n} = \boldsymbol{b}_1^\top \boldsymbol{x}_n \,, \tag{10.8}$$

i.e., it is the coordinate of the orthogonal projection of $\boldsymbol{x}_n$ onto the one-dimensional subspace spanned by $\boldsymbol{b}_1$, see Section 3.7. We substitute (10.8) into (10.7), which yields

$$V_1 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{b}_1^\top \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{b}_1^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_1 \tag{10.9a}$$

$$= \boldsymbol{b}_1^\top \left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top\right) \boldsymbol{b}_1 = \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 \,, \tag{10.9b}$$

where $\boldsymbol{S}$ is the data covariance matrix defined in (10.1). In (10.9a) we have used the fact that the dot product of two vectors is symmetric with respect to its arguments, that is $\boldsymbol{b}_1^\top \boldsymbol{x}_n = \boldsymbol{x}_n^\top \boldsymbol{b}_1$.

Notice that arbitrarily increasing the magnitude of the vector $\boldsymbol{b}_1$ increases V_1, that is, a vector $\boldsymbol{b}_1$ that is two times longer can result in V_1 that is potentially four times larger. Therefore, we restrict all solutions to $\|\boldsymbol{b}_1\|^2 = 1$, which results in a constrained optimization problem in which we seek the direction along which the data varies most.

$\|\boldsymbol{b}_1\|^2 = 1 \iff \|\boldsymbol{b}_1\| = 1$.

With the restriction of the solution space to unit vectors the vector $\boldsymbol{b}_1$ that points in the direction of maximum variance can be found by the constrained optimization problem

$$\begin{aligned} &\max_{\boldsymbol{b}_1} \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 \\ &\text{subject to } \|\boldsymbol{b}_1\|^2 = 1 \,. \end{aligned} \tag{10.10}$$

Following Section 7.2, we obtain the Lagrangian

$$\mathfrak{L}(\boldsymbol{b}_1, \lambda) = \boldsymbol{b}_1^\top \boldsymbol{S} \boldsymbol{b}_1 + \lambda_1(1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1) \tag{10.11}$$

to solve this constrained optimization problem. The partial derivatives of $\mathfrak{L}$ with respect to $\boldsymbol{b}_1$ and λ_1 are

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{b}_1} = 2\boldsymbol{b}_1^\top \boldsymbol{S} - 2\lambda_1 \boldsymbol{b}_1^\top \tag{10.12}$$

$$\frac{\partial \mathfrak{L}}{\partial \lambda_1} = 1 - \boldsymbol{b}_1^\top \boldsymbol{b}_1 \,, \tag{10.13}$$

respectively. Setting these partial derivatives to $\mathbf{0}$ gives us the relations

$$\boldsymbol{S}\boldsymbol{b}_1 = \lambda_1 \boldsymbol{b}_1 \,, \tag{10.14}$$

$$\boldsymbol{b}_1^\top \boldsymbol{b}_1 = 1 \,. \tag{10.15}$$

By comparing with the definition of an eigenvalue decomposition (Section 4.4), we see that $\boldsymbol{b}_1$ is an eigenvector of the data covariance matrix $\boldsymbol{S}$, and the Lagrange multiplier λ_1 plays the role of the corresponding eigenvalue. This eigenvector property (10.14) allows us to rewrite our variance objective (10.10) as

The quantity $\sqrt{\lambda_1}$ is also called the *loading* of the unit vector $\boldsymbol{b}_1$ and represents the standard deviation of the data accounted for by the principal subspace $\mathrm{span}[\boldsymbol{b}_1]$.

$$V_1 = \boldsymbol{b}_1^\top \boldsymbol{S}\boldsymbol{b}_1 = \lambda_1 \boldsymbol{b}_1^\top \boldsymbol{b}_1 = \lambda_1 \,, \tag{10.16}$$

i.e., the variance of the data projected onto a one-dimensional subspace equals the eigenvalue that is associated with the basis vector $\boldsymbol{b}_1$ that spans this subspace. Therefore, to maximize the variance of the low-dimensional code we choose the basis vector associated with the largest eigenvalue of the data covariance matrix. This eigenvector is called the first *principal component*. We can determine the effect/contribution of the principal component $\boldsymbol{b}_1$ in the original data space by mapping the coordinate z_{1n} back into data space, which gives us the projected data point

principal component

$$\tilde{\boldsymbol{x}}_n = \boldsymbol{b}_1 z_{1n} = \boldsymbol{b}_1 \boldsymbol{b}_1^\top \boldsymbol{x}_n \in \mathbb{R}^D \tag{10.17}$$

in the original data space.

Remark. Although $\tilde{\boldsymbol{x}}_n$ is a D-dimensional vector it only requires a single coordinate z_{1n} to represent it with respect to the basis vector $\boldsymbol{b}_1 \in \mathbb{R}^D$. $\diamondsuit$

10.2.2 M-dimensional Subspace with Maximal Variance

Assume we have found the first $m-1$ principal components as the $m-1$ eigenvectors of $\boldsymbol{S}$ that are associated with the largest $m-1$ eigenvalues. Since $\boldsymbol{S}$ is symmetric, these eigenvectors form an ONB of an $(m-1)$-dimensional subspace of $\mathbb{R}^D$. Generally, the mth principal component can be found by subtracting the effect of the first $m-1$ principal components $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{m-1}$ from the data, thereby trying to find principal components that compress the remaining information. We achieve this by first subtracting the contribution of the $m-1$ principal components from the data,

similar to (10.17), so that we arrive at the new data matrix

$$\hat{\boldsymbol{X}} := \boldsymbol{X} - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{X} \,, \tag{10.18}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ contains the data points as column vectors. The matrix $\hat{\boldsymbol{X}} := [\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_N] \in \mathbb{R}^{D \times N}$ in (10.18) contains the data that only contains the information that has not yet been compressed.

Remark (Notation). Throughout this chapter, we do not follow the convention of collecting data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ as rows of the data matrix, but we define them to be the columns of $\boldsymbol{X}$. This means that our data matrix $\boldsymbol{X}$ is a $D \times N$ matrix instead of the conventional $N \times D$ matrix. The reason for our choice is that the algebra operations work out smoothly without the need to either transpose the matrix or to redefine vectors as row vectors that are left-multiplied onto matrices. ♢

To find the mth principal component, we maximize the variance

$$V_m = \mathbb{V}[z_m] = \frac{1}{N} \sum_{n=1}^{N} z_{mn}^2 = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{b}_m^\top \boldsymbol{x}_n)^2 = \boldsymbol{b}_m^\top \hat{\boldsymbol{S}} \boldsymbol{b}_m \,, \tag{10.19}$$

subject to $\|\boldsymbol{b}_m\|^2 = 1$, where we followed the same steps as in (10.9b) and defined $\hat{\boldsymbol{S}}$ as the data covariance matrix of $\hat{\boldsymbol{X}}$. As previously, when we looked at the first principal component alone, we solve a constrained optimization problem and discover that the optimal solution $\boldsymbol{b}_m$ is the eigenvector of $\hat{\boldsymbol{S}}$ that is associated with the largest eigenvalue of $\hat{\boldsymbol{S}}$.

However, it also turns out that $\boldsymbol{b}_m$ is an eigenvector of $\boldsymbol{S}$. It holds that

$$\hat{\boldsymbol{S}} = \frac{1}{N} \sum_{n=1}^{N} \hat{\boldsymbol{x}}_n \hat{\boldsymbol{x}}_n^\top \overset{(10.18)}{=} \frac{1}{N} \sum_{n=1}^{N} \Big(\boldsymbol{x}_n - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{x}_n \Big) \Big(\boldsymbol{x}_n - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{x}_n \Big)^\top \tag{10.20a}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top - 2 \boldsymbol{x}_n \boldsymbol{x}_n^\top \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top + \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \,, \tag{10.20b}$$

where we exploited the symmetries $\boldsymbol{x}_n^\top \boldsymbol{b}_i = \boldsymbol{b}_i^\top \boldsymbol{x}_n$ and $\boldsymbol{b}_i \boldsymbol{x}_n^\top = \boldsymbol{x}_n \boldsymbol{b}_i^\top$ to summarize

$$-\boldsymbol{x}_n \boldsymbol{x}_n^\top \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top = -2 \boldsymbol{x}_n \boldsymbol{x}_n^\top \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top \,. \tag{10.21}$$

If we take a vector $\boldsymbol{b}_m$ with $\|\boldsymbol{b}_m\| = 1$ that is orthogonal to all $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{m-1}$ and right-multiply $\boldsymbol{b}_m$ to $\hat{\boldsymbol{S}}$ in (10.20b) we obtain

$$\hat{\boldsymbol{S}} \boldsymbol{b}_m = \frac{1}{N} \sum_{n=1}^{N} \hat{\boldsymbol{x}}_n \hat{\boldsymbol{x}}_n^\top \boldsymbol{b}_m = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_m = \boldsymbol{S} \boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m \,. \tag{10.22}$$

Here we applied the orthogonality property $\boldsymbol{b}_i^\top \boldsymbol{b}_m = 0$ for $i = 1, \ldots, m-1$ (all terms involving sums up to $m-1$ vanish). Equation (10.22) reveals that $\boldsymbol{b}_m$ is an eigenvector of both $\hat{\boldsymbol{S}}$ and the original data covariance matrix $\boldsymbol{S}$. In the former case, λ_m is the largest eigenvalue, in latter case, λ_m is the mth largest eigenvalue. With this the variance of the data projected onto the mth principal component is

$$V_m = \boldsymbol{b}_m^\top \boldsymbol{S} \boldsymbol{b}_m \overset{(10.22)}{=} \lambda_m \boldsymbol{b}_m^\top \boldsymbol{b}_m = \lambda_m \tag{10.23}$$

since $\boldsymbol{b}_m^\top \boldsymbol{b}_m = 1$. This means that the variance of the data, when projected onto an M-dimensional subspace, equals the sum of the eigenvalues that is associated with the corresponding eigenvectors of the data covariance matrix.

Overall, to find an M-dimensional subspace of $\mathbb{R}^D$ that retains as much information as possible, PCA tells us to choose the columns of the matrix $\boldsymbol{B}$ in (10.3) as the M eigenvectors of the data covariance matrix $\boldsymbol{S}$ that are associated with the M largest eigenvalues. The maximum amount of variance PCA can capture with the first M principal components is

$$V_M = \sum_{m=1}^{M} \lambda_m \,, \tag{10.24}$$

where the λ_m are the M largest eigenvalues of the data covariance matrix $\boldsymbol{S}$. Consequently, the variance lost by data compression via PCA is

$$J_M := \sum_{j=M+1}^{D} \lambda_j = V_D - V_M \,. \tag{10.25}$$

Instead of these absolute quantities, we can also define the relative amount of variance captured as $\frac{V_M}{V_D}$, and the relative amount of variance lost by compression as $1 - \frac{V_M}{V_D}$.

10.3 Projection Perspective

In the following, we will derive PCA as an algorithm for linear dimensionality reduction that directly minimizes the average reconstruction error. This perspective allows us to interpret PCA as an algorithm that implements an optimal linear auto-encoder. We will draw heavily from Chapters 2 and 3.

In the previous section, we derived PCA by maximizing the variance in the projected space to retain as much information as possible. In the following, we will look at the difference vectors between the original data $\boldsymbol{x}_n$ and their reconstruction $\tilde{\boldsymbol{x}}_n$ and minimize this distance so that $\boldsymbol{x}_n$ and $\tilde{\boldsymbol{x}}_n$ are as close as possible. Figure 10.5 illustrates this setting.

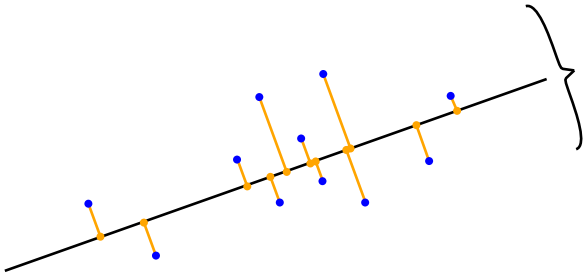


Figure 10.5 Illustration of the projection approach to PCA. We aim to find a lower-dimensional subspace (line) so that the difference vector between projected (orange) and original (blue) data is as short as possible.

10.3.1 Setting and Objective

Assume an (ordered) orthonormal basis (ONB) $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ of $\mathbb{R}^D$, i.e., $\boldsymbol{b}_i^\top \boldsymbol{b}_j = 1$ if and only if $i = j$ and 0 otherwise.

Remark. (Orthogonal Complement) Consider a D-dimensional vector space V and an M-dimensional subspace $U \subseteq V$. Then its *orthogonal complement* $U^\perp$ is a $(D-M)$-dimensional subspace of V and contains all vectors in V that are orthogonal to every vector in U. Furthermore, $U \cap U^\perp = \{\mathbf{0}\}$ so that any vector $\boldsymbol{x} \in V$ can be (uniquely) decomposed into

orthogonal complement

$$\boldsymbol{x} = \sum_{m=1}^{M} \lambda_m \boldsymbol{b}_m + \sum_{j=1}^{D-M} \psi_j \boldsymbol{b}_j^\perp, \quad \lambda_m,\ \psi_j \in \mathbb{R}\,, \tag{10.26}$$

where $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_M)$ is a basis of U and $(\boldsymbol{b}_1^\perp, \dots, \boldsymbol{b}_{D-M}^\perp)$ is a basis of $U^\perp$. ♢

From Section 2.5 we know that for a basis $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ of $\mathbb{R}^D$ any $\boldsymbol{x} \in \mathbb{R}^D$ can be written as a linear combination of the basis vectors of $\mathbb{R}^D$, i.e.,

$$\boldsymbol{x} = \sum_{d=1}^{D} \zeta_d \boldsymbol{b}_d = \sum_{m=1}^{M} \zeta_m \boldsymbol{b}_m + \sum_{j=M+1}^{D} \zeta_j \boldsymbol{b}_j \tag{10.27}$$

for suitable coordinates $\zeta_d \in \mathbb{R}$.

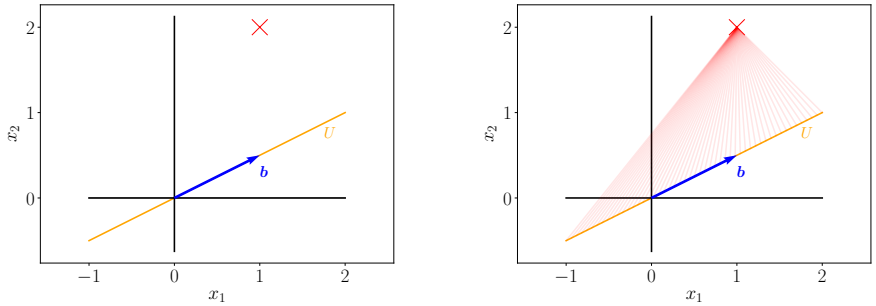
We are interested in finding vectors $\tilde{\boldsymbol{x}} \in \mathbb{R}^D$, which live in lower-dimensional subspace $U \subseteq \mathbb{R}^D$, $\dim(U) = M$, so that

$$\tilde{\boldsymbol{x}} = \sum_{m=1}^{M} z_m \boldsymbol{b}_m \in U \subseteq \mathbb{R}^D \tag{10.28}$$

is as similar to $\boldsymbol{x}$ as possible. Note that at this point we need to assume that the coordinates z_m of $\tilde{\boldsymbol{x}}$ and ζ_m of $\boldsymbol{x}$ are not identical.

Vectors $\tilde{\boldsymbol{x}} \in U$ could be vectors on a plane in $\mathbb{R}^3$. The dimensionality of the plane is 2, but the vectors still have three coordinates with respect to the standard basis of $\mathbb{R}^3$.

In the following, we use exactly this kind of representation of $\tilde{\boldsymbol{x}}$ to find optimal coordinates $\boldsymbol{z}$ and basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ such that $\tilde{\boldsymbol{x}}$ is as similar to the original data point $\boldsymbol{x}$, i.e., we aim to minimize the (Euclidean) distance $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$. Figure 10.6 illustrates this setting. Without loss of generality, we assume that the dataset $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^D$, is centered at $\mathbf{0}$, i.e., $\mathbb{E}[\mathcal{X}] = \mathbf{0}$. Without the zero-mean assumption, we would



(a) Setting.

(b) Differences $\boldsymbol{x} - \tilde{\boldsymbol{x}}$ for 50 candidates $\tilde{\boldsymbol{x}}$ are shown by the red lines.

Figure 10.6 Simplified projection setting. (a) A vector $\boldsymbol{x} \in \mathbb{R}^2$ (red cross) shall be projected onto a one-dimensional subspace $U \subseteq \mathbb{R}^2$ spanned by $\boldsymbol{b}$. (b) shows the difference vectors between $\boldsymbol{x}$ and some candidates $\tilde{\boldsymbol{x}}$.

arrive at exactly the same solution but the notation would be substantially more cluttered.

We are interested in finding the best linear projection of $\mathcal{X}$ onto a lower-dimensional subspace U of $\mathbb{R}^D$ with $\dim(U) = M$ and orthonormal basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$. We will call this subspace U the *principal subspace*. The projections of the data points are denoted by

principal subspace

$$\tilde{\boldsymbol{x}}_n := \sum_{m=1}^{M} z_{mn} \boldsymbol{b}_m = \boldsymbol{B}\boldsymbol{z}_n \in \mathbb{R}^D \,, \tag{10.29}$$

where $\boldsymbol{z}_n := [z_{1n}, \ldots, z_{Mn}]^\top \in \mathbb{R}^M$ is the coordinate vector of $\tilde{\boldsymbol{x}}_n$ with respect to the basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M)$. More specifically, we are interested in having the $\tilde{\boldsymbol{x}}_n$ as similar to $\boldsymbol{x}_n$ as possible.

The similarity measure we use in the following is the squared Euclidean norm $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2$ between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. We therefore define our objective as the minimizing the average squared Euclidean distance (*reconstruction error*) (Pearson, 1901b)

reconstruction error

$$J_M := \frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 \,, \tag{10.30}$$

where we make it explicit that the dimension of the subspace onto which we project the data is M. In order to find this optimal linear projection, we need to find the orthonormal basis of the principal subspace and the coordinates $\boldsymbol{z}_n \in \mathbb{R}^M$ of the projections with respect to this basis.

To find the coordinates $\boldsymbol{z}_n$ and the ONB of the principal subspace we follow a two-step approach. First, we optimize the coordinates $\boldsymbol{z}_n$ for a given ONB $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M)$; second, we find the optimal ONB.

10.3.2 Finding Optimal Coordinates

Let us start by finding the optimal coordinates $z_{1n}, \ldots, z_{Mn}$ of the projections $\tilde{\boldsymbol{x}}_n$ for $n = 1, \ldots, N$. Consider Figure 10.6(b) where the principal

subspace is spanned by a single vector $\boldsymbol{b}$. Geometrically speaking, finding the optimal coordinates z corresponds to finding the representation of the linear projection $\tilde{\boldsymbol{x}}$ with respect to b that minimizes the distance between $\tilde{\boldsymbol{x}} - \boldsymbol{x}$. From Figure 10.6(b) it is clear that this will be the orthogonal projection, and in the following we will show exactly this.

We assume an ONB $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_M)$ of $U \subseteq \mathbb{R}^D$. To find the optimal coordinates $\boldsymbol{z}_m$ with respect to this basis, we require the partial derivatives

$$\frac{\partial J_M}{\partial z_{in}} = \frac{\partial J_M}{\partial \tilde{\boldsymbol{x}}_n} \frac{\partial \tilde{\boldsymbol{x}}_n}{\partial z_{in}}, \tag{10.31a}$$

$$\frac{\partial J_M}{\partial \tilde{\boldsymbol{x}}_n} = -\frac{2}{N}(\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n)^\top \in \mathbb{R}^{1\times D}, \tag{10.31b}$$

$$\frac{\partial \tilde{\boldsymbol{x}}_n}{\partial z_{in}} \overset{(10.29)}{=} \frac{\partial}{\partial z_{in}} \left(\sum_{m=1}^{M} z_{mn} \boldsymbol{b}_m \right) = \boldsymbol{b}_i \tag{10.31c}$$

for $i = 1, \dots, M$, such that we obtain

$$\frac{\partial J_M}{\partial z_{in}} \overset{\substack{(10.31b)\\(10.31c)}}{=} -\frac{2}{N}(\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n)^\top \boldsymbol{b}_i \overset{(10.29)}{=} -\frac{2}{N}\left(\boldsymbol{x}_n - \sum_{m=1}^{M} z_{mn} \boldsymbol{b}_m \right)^\top \boldsymbol{b}_i \tag{10.32a}$$

$$\overset{\text{ONB}}{=} -\frac{2}{N}(\boldsymbol{x}_n^\top \boldsymbol{b}_i - z_{in} \underbrace{\boldsymbol{b}_i^\top \boldsymbol{b}_i}_{=1}) = -\frac{2}{N}(\boldsymbol{x}_n^\top \boldsymbol{b}_i - z_{in}). \tag{10.32b}$$

Setting this partial derivative to 0 yields immediately the optimal coordinates

$$z_{in} = \boldsymbol{x}_n^\top \boldsymbol{b}_i = \boldsymbol{b}_i^\top \boldsymbol{x}_n \tag{10.33}$$

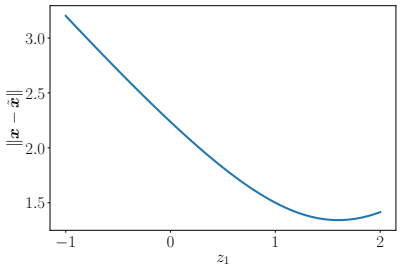
for $i = 1, \dots, M$ and $n = 1, \dots, N$. This means, the optimal coordinates z_{in} of the projection $\tilde{\boldsymbol{x}}_n$ are the coordinates of the orthogonal projection (see Section 3.7) of the original data point $\boldsymbol{x}_n$ onto the one-dimensional subspace that is spanned by $\boldsymbol{b}_i$. Consequently:

The coordinates of the optimal projection of $\boldsymbol{x}_n$ with respect to the basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ are the coordinates of the orthogonal projection of $\boldsymbol{x}_n$ onto the principal subspace.

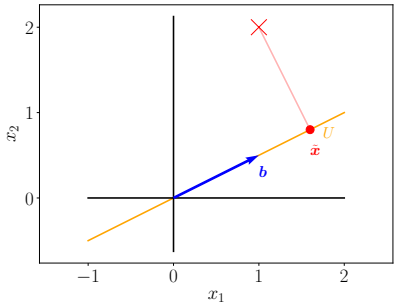
- The optimal linear projection $\tilde{\boldsymbol{x}}_n$ of $\boldsymbol{x}_n$ is an orthogonal projection.
- The coordinates of $\tilde{\boldsymbol{x}}_n$ with respect to the basis $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ are the coordinates of the orthogonal projection of $\boldsymbol{x}_n$ onto the principal subspace.
- An orthogonal projection is the best linear mapping we can find given the objective (10.30).
- The coordinates ζ_m of $\boldsymbol{x}$ in (10.27) and the coordinates z_m of $\tilde{\boldsymbol{x}}$ in (10.28) must be identical for $m = 1, \dots, M$ in PCA since $U^\perp = \operatorname{span}[\boldsymbol{b}_{M+1}, \dots, \boldsymbol{b}_D]$ is the orthogonal complement of $U = \operatorname{span}[\boldsymbol{b}_1, \dots, \boldsymbol{b}_M]$.

Remark (Orthogonal Projections with Orthonormal Basis Vectors). Let us briefly recap orthogonal projections from Section 3.7. If $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_D)$ is an

Figure 10.7 Optimal projection of a vector $\boldsymbol{x} \in \mathbb{R}^2$ onto a one-dimensional subspace (continuation from Figure 10.6). (a) Distances $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$ for some $\tilde{\boldsymbol{x}} \in U$. (b) Orthogonal projection and optimal coordinates.



(a) Distances $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$ for some $\tilde{\boldsymbol{x}} = z_1\boldsymbol{b} \in U = \text{span}[\boldsymbol{b}]$, see panel (b) for the setting.



(b) The vector $\tilde{\boldsymbol{x}}$ that minimizes the distance in panel (a) is its orthogonal projection onto U. The coordinate of the projection $\tilde{\boldsymbol{x}}$ with respect to the basis vector $\boldsymbol{b}$ that spans U is the factor we need to scale $\boldsymbol{b}$ in order to "reach" $\tilde{\boldsymbol{x}}$.

orthonormal basis of $\mathbb{R}^D$ then

$$\tilde{\boldsymbol{x}} = \boldsymbol{b}_j \underbrace{(\boldsymbol{b}_j^\top \boldsymbol{b}_j)}_{=1}{}^{-1} \boldsymbol{b}_j^\top \boldsymbol{x} = \boldsymbol{b}_j \boldsymbol{b}_j^\top \boldsymbol{x} \in \mathbb{R}^D \tag{10.34}$$

$\boldsymbol{x}^\top \boldsymbol{b}_j$ is the coordinate of the orthogonal projection of $\boldsymbol{x}$ onto the one-dimensional subspace spanned by $\boldsymbol{b}_j$.

is the orthogonal projection of $\boldsymbol{x}$ onto the subspace spanned by the jth basis vector, and $z_j = \boldsymbol{b}_j^\top \boldsymbol{x}$ is the coordinate of this projection with respect to the basis vector $\boldsymbol{b}_j$ that spans that subspace since $z_j \boldsymbol{b}_j = \tilde{\boldsymbol{x}}$. Figure 10.7 illustrates this setting.

More generally, if we aim to project onto an M-dimensional subspace of $\mathbb{R}^D$, we obtain the orthogonal projection of $\boldsymbol{x}$ onto the M-dimensional subspace with orthonormal basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ as

$$\tilde{\boldsymbol{x}} = \boldsymbol{B} \underbrace{(\boldsymbol{B}^\top \boldsymbol{B})}_{=\boldsymbol{I}}{}^{-1} \boldsymbol{B}^\top \boldsymbol{x} = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x}\,, \tag{10.35}$$

where we defined $\boldsymbol{B} := [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M] \in \mathbb{R}^{D \times M}$. The coordinates of this projection with respect to the ordered basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M)$ are $\boldsymbol{z} := \boldsymbol{B}^\top \boldsymbol{x}$ as discussed in Section 3.7.

We can think of the coordinates as a representation of the projected vector in a new coordinate system defined by $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M)$. Note that although $\tilde{\boldsymbol{x}} \in \mathbb{R}^D$ we only need M coordinates $z_1, \ldots, z_M$ to represent this vector; the other $D - M$ coordinates with respect to the basis vectors $(\boldsymbol{b}_{M+1}, \ldots, \boldsymbol{b}_D)$ are always 0. ♢

So far, we showed that for a given ONB we can find the optimal coordinates of $\tilde{\boldsymbol{x}}$ by an orthogonal projection onto the principal subspace. In the following, we will determine what the best basis is.

10.3.3 Finding the Basis of the Principal Subspace

To determine the basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ of the principal subspace, we rephrase the loss function (10.30) using the results we have so far. This will make it easier to find the basis vectors. To reformulate the loss function, we exploit our results from before and obtain

$$\tilde{\boldsymbol{x}}_n = \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m \overset{(10.33)}{=} \sum_{m=1}^{M} (\boldsymbol{x}_n^\top \boldsymbol{b}_m)\boldsymbol{b}_m \,. \tag{10.36}$$

We now exploit the symmetry of the dot product, which yields

$$\tilde{\boldsymbol{x}}_n = \left(\sum_{m=1}^{M} \boldsymbol{b}_m \boldsymbol{b}_m^\top\right) \boldsymbol{x}_n \,. \tag{10.37}$$

Since we can generally write the original data point $\boldsymbol{x}_n$ as a linear combination of all basis vectors, we can also write

$$\boldsymbol{x}_n = \sum_{d=1}^{D} z_{dn}\boldsymbol{b}_d \overset{(10.33)}{=} \sum_{d=1}^{D} (\boldsymbol{x}_n^\top \boldsymbol{b}_d)\boldsymbol{b}_d = \left(\sum_{d=1}^{D} \boldsymbol{b}_d \boldsymbol{b}_d^\top\right) \boldsymbol{x}_n \tag{10.38a}$$

$$= \left(\sum_{m=1}^{M} \boldsymbol{b}_m \boldsymbol{b}_m^\top\right) \boldsymbol{x}_n + \left(\sum_{j=M+1}^{D} \boldsymbol{b}_j \boldsymbol{b}_j^\top\right) \boldsymbol{x}_n \,, \tag{10.38b}$$

where we split the sum with D terms into a sum over M and a sum over $D - M$ terms. With this result, we find that the displacement vector $\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n$, i.e., the difference vector between the original data point and its projection, is

$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \left(\sum_{j=M+1}^{D} \boldsymbol{b}_j \boldsymbol{b}_j^\top\right) \boldsymbol{x}_n \tag{10.39a}$$

$$= \sum_{j=M+1}^{D} (\boldsymbol{x}_n^\top \boldsymbol{b}_j)\boldsymbol{b}_j \,. \tag{10.39b}$$

This means the difference is exactly the projection of the data point onto the orthogonal complement of the principal subspace: We identify the matrix $\sum_{j=M+1}^{D} \boldsymbol{b}_j \boldsymbol{b}_j^\top$ in (10.39a) as the projection matrix that performs this projection. This also means the displacement vector $\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n$ lies in the subspace that is orthogonal to the principal subspace as illustrated in Figure 10.8.

PCA finds the best rank-M approximation of the identity matrix.

Remark (Low-Rank Approximation). In (10.39a), we saw that the projection matrix, which projects $\boldsymbol{x}$ onto $\tilde{\boldsymbol{x}}$, is given by

$$\sum_{m=1}^{M} \boldsymbol{b}_m \boldsymbol{b}_m^\top = \boldsymbol{B}\boldsymbol{B}^\top \,. \tag{10.40}$$

By construction as a sum of rank-one matrices $\boldsymbol{b}_m \boldsymbol{b}_m^\top$ we see that $\boldsymbol{B}\boldsymbol{B}^\top$

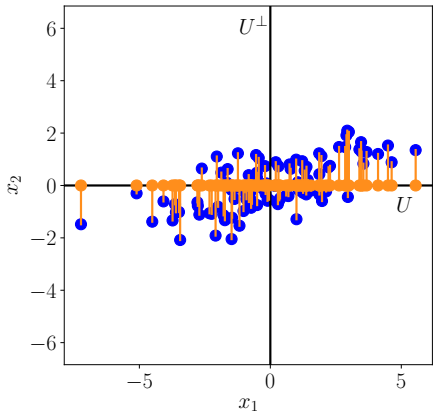


Figure 10.8 Orthogonal projection and displacement vectors. When projecting data points $\boldsymbol{x}_n$ (blue) onto subspace U_1 we obtain $\tilde{\boldsymbol{x}}_n$ (orange). The displacement vector $\tilde{\boldsymbol{x}}_n - \boldsymbol{x}_n$ lies completely in the orthogonal complement U_2 of U_1.

is symmetric and has rank M. Therefore, the average squared reconstruction error can also be written as

$$\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{x}_n - \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x}_n\right\|^2 \tag{10.41a}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left\|(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^\top)\boldsymbol{x}_n\right\|^2 . \tag{10.41b}$$

Finding orthonormal basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$ so that the difference between the original data $\boldsymbol{x}_n$ and their projections $\tilde{\boldsymbol{x}}_n$, $n = 1, \dots, N$, is minimized is equivalent to finding the best rank-M approximation $\boldsymbol{B}\boldsymbol{B}^\top$ of the identity matrix $\boldsymbol{I}$, see Section 4.6. ♢

Now, we have all the tools to reformulate the loss function (10.30).

$$J_M = \frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 \overset{(10.39b)}{=} \frac{1}{N}\sum_{n=1}^{N}\left\|\sum_{j=M+1}^{D}(\boldsymbol{b}_j^\top \boldsymbol{x}_n)\boldsymbol{b}_j\right\|^2 . \tag{10.42}$$

We now explicitly compute the squared norm and exploit the fact that the $\boldsymbol{b}_j$ form an ONB, which yields

$$J_M = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D}(\boldsymbol{b}_j^\top \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D}\boldsymbol{b}_j^\top \boldsymbol{x}_n \boldsymbol{b}_j^\top \boldsymbol{x}_n \tag{10.43a}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D}\boldsymbol{b}_j^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_j , \tag{10.43b}$$

where we exploited the symmetry of the dot product in the last step to

write $\boldsymbol{b}_j^\top \boldsymbol{x}_n = \boldsymbol{x}_n^\top \boldsymbol{b}_j$. We can now swap the sums and obtain

$$J_M = \sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top\right)}_{=:\boldsymbol{S}} \boldsymbol{b}_j = \sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \boldsymbol{S} \boldsymbol{b}_j \tag{10.44a}$$

$$= \sum_{j=M+1}^{D} \operatorname{tr}(\boldsymbol{b}_j^\top \boldsymbol{S} \boldsymbol{b}_j) \sum_{j=M+1}^{D} \operatorname{tr}(\boldsymbol{S} \boldsymbol{b}_j \boldsymbol{b}_j^\top) = \operatorname{tr}\Big(\underbrace{\Big(\sum_{j=M+1}^{D} \boldsymbol{b}_j \boldsymbol{b}_j^\top \Big)}_{\text{projection matrix}} \boldsymbol{S} \Big), \tag{10.44b}$$

where we exploited the property that the trace operator $\operatorname{tr}(\cdot)$, see (4.18), is linear and invariant to cyclic permutations of its arguments. Since we assumed that our dataset is centered, i.e., $\mathbb{E}[\mathcal{X}] = \mathbf{0}$, we identify $\boldsymbol{S}$ as the data covariance matrix. We see that the projection matrix in (10.44b) is constructed as a sum of rank-one matrices $\boldsymbol{b}_j \boldsymbol{b}_j^\top$ so that it itself is of rank $D - M$.

Minimizing the average squared reconstruction error is equivalent to minimizing the projection of the data covariance matrix onto the orthogonal complement of the principal subspace.

Equation (10.44a) implies that we can formulate the average squared reconstruction error equivalently as the covariance matrix of the data, projected onto the orthogonal complement of the principal subspace. Minimizing the average squared reconstruction error is therefore equivalent to minimizing the variance of the data when projected onto the subspace we ignore, i.e., the orthogonal complement of the principal subspace. Equivalently, we maximize the variance of the projection that we retain in the principal subspace, which links the projection loss immediately to the maximum-variance formulation of PCA discussed in Section 10.2. But this then also means that we will obtain the same solution that we obtained for the maximum-variance perspective. Therefore, we omit a derivation that is identical to the one Section 10.2 and summarize the results from earlier in the light of the projection perspective.

Minimizing the average squared reconstruction error is equivalent to maximizing the variance of the projected data.

The average squared reconstruction error, when projecting onto the M-dimensional principal subspace, is

$$J_M = \sum_{j=M+1}^{D} \lambda_j \,, \tag{10.45}$$

where λ_j are the eigenvalues of the data covariance matrix. Therefore, to minimize (10.45) we need to select the smallest $D - M$ eigenvalues, which then implies that their corresponding eigenvectors are the basis of the orthogonal complement of the principal subspace. Consequently, this means that the basis of the principal subspace are the eigenvectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ that are associated with the largest M eigenvalues of the data covariance matrix.

Example 10.2 (MNIST Digits Embedding)

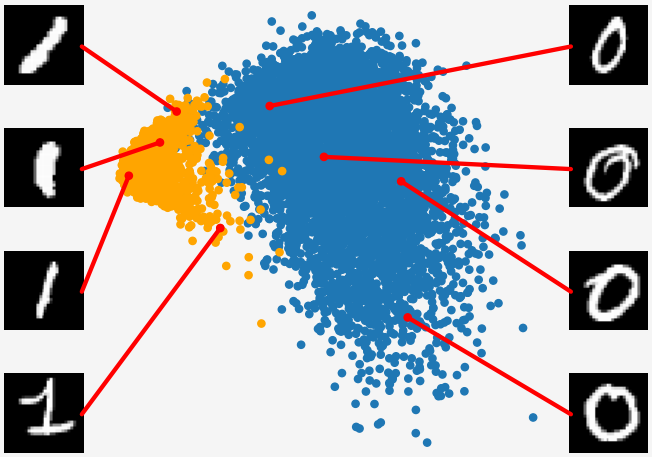


Figure 10.9 Embedding of MNIST digits 0 (blue) and 1 (orange) in a two-dimensional principal subspace using PCA. Four examples embeddings of the digits '0' and '1' in the principal subspace are highlighted in red with their corresponding original digit.

Figure 10.9 visualizes the training data of the MMIST digits '0' and '1' embedded in the vector subspace spanned by the first two principal components. We can see a relatively clear separation between '0's (blue dots) and '1's (orange dots), and we can see the variation within each individual cluster.

10.4 Eigenvector Computation and Low-Rank Approximations

obtained the basis of the principal subspace as the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top, \tag{10.46}$$

$$\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}. \tag{10.47}$$

To get the eigenvalues (and the corresponding eigenvectors) of $\boldsymbol{S}$, we can follow two approaches:

Eigendecomposition or SVD to compute eigenvectors.

- We perform an eigendecomposition (see Section 4.2) and compute the eigenvalues and eigenvectors of $\boldsymbol{S}$ directly.
- We use a singular value decomposition (see Section 4.5). Since $\boldsymbol{S}$ is symmetric and factorizes into $\boldsymbol{X}\boldsymbol{X}^\top$ (ignoring the factor $\frac{1}{N}$), the eigenvalues of $\boldsymbol{S}$ are the squared singular values of $\boldsymbol{X}$. More specifically, if

the SVD of $\boldsymbol{X}$ is given by

$$\underbrace{\boldsymbol{X}}_{D\times N} = \underbrace{\boldsymbol{U}}_{D\times D}\underbrace{\boldsymbol{\Sigma}}_{D\times N}\underbrace{\boldsymbol{V}^\top}_{N\times N}, \tag{10.48}$$

where $\boldsymbol{U} \in \mathbb{R}^{D\times D}$ and $\boldsymbol{V}^\top \in \mathbb{R}^{N\times N}$ are orthogonal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times N}$ is a matrix whose only non-zero entries are the singular values $\sigma_{ii} \geqslant 0$. Then it follows that

$$\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top = \frac{1}{N}\boldsymbol{U}\boldsymbol{\Sigma}\underbrace{\boldsymbol{V}^\top\boldsymbol{V}}_{=\boldsymbol{I}_N}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top = \frac{1}{N}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top. \tag{10.49}$$

With the results from Section 4.5 we get that the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$ (and therefore $\boldsymbol{S}$). Furthermore, the eigenvalues λ_d of $\boldsymbol{S}$ are related to the singular values of $\boldsymbol{X}$ via

The columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{S}$.

$$\lambda_d = \frac{\sigma_d^2}{N}. \tag{10.50}$$

10.4.1 PCA using Low-rank Matrix Approximations

To maximize the variance of the projected data (or minimize the average squared reconstruction error), PCA chooses the columns of $\boldsymbol{U}$ in (10.49) to be the eigenvectors that are associated with the M largest eigenvalues of the data covariance matrix $\boldsymbol{S}$ so that we identify $\boldsymbol{U}$ as the projection matrix $\boldsymbol{B}$ in (10.3), which projects the original data onto a lower-dimensional subspace of dimension M. The *Eckart-Young Theorem* (Section 4.6) offers a direct way to estimate the low-dimensional representation. Consider the best rank-M approximation

Eckart-Young Theorem

$$\tilde{\boldsymbol{X}}_M := \operatorname{argmin}_{\operatorname{rk}(\boldsymbol{A})\leqslant M} \|\boldsymbol{X} - \boldsymbol{A}\|_2 \in \mathbb{R}^{D\times N} \tag{10.51}$$

of $\boldsymbol{X}$, where $\|\cdot\|_2$ is the spectral norm defined in (4.110). The Eckart-Young Theorem states that $\tilde{\boldsymbol{X}}_M$ is given by truncating the SVD at the top-M singular value. In other words, we obtain

$$\tilde{\boldsymbol{X}}_M = \underbrace{\boldsymbol{U}_M}_{D\times M}\underbrace{\boldsymbol{\Sigma}_M}_{M\times M}\underbrace{\boldsymbol{V}_M^\top}_{M\times N} \in \mathbb{R}^{D\times N} \tag{10.52}$$

with orthogonal matrices $\boldsymbol{U}_M := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M] \in \mathbb{R}^{D\times M}$ and $\boldsymbol{V}_M := [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_M] \in \mathbb{R}^{N\times M}$ and a diagonal matrix $\boldsymbol{\Sigma}_M \in \mathbb{R}^{M\times M}$ whose diagonal entries are the M largest singular values of $\boldsymbol{X}$.

10.4.2 Practical Aspects

Finding eigenvalues and eigenvectors is also important in other fundamental machine learning methods that require matrix decompositions. In theory, as we discussed in Section 4.2, we can solve for the eigenvalues as roots of the characteristic polynomial. However, for matrices larger than

4×4 this is not possible because we would need to find the roots of a polynomial of degree 5 or higher. However, the Abel-Ruffini theorem (Ruffini, 1799; Abel, 1826) states that there exists no algebraic solution to this problem for polynomials of degree 5 or more. Therefore, in practice, we solve for eigenvalues or singular values using iterative methods, which are implemented in all modern packages for linear algebra.

`np.linalg.eigh` or `np.linalg.svd`

In many applications (such as PCA presented in this chapter), we only require a few eigenvectors. It would be wasteful to compute the full decomposition, and then discard all eigenvectors with eigenvalues that are beyond the first few. It turns out that if we are interested in only the first few eigenvectors (with the largest eigenvalues) iterative processes, which directly optimize these eigenvectors, are computationally more efficient than a full eigendecomposition (or SVD). In the extreme case of only needing the first eigenvector, a simple method called the *power iteration* is very efficient. Power iteration chooses a random vector $\boldsymbol{x}_0$ that is not in the null space of $\boldsymbol{S}$ and follows the iteration

power iteration

If $\boldsymbol{S}$ is invertible, it is sufficient to ensure that $\boldsymbol{x}_0 \neq \boldsymbol{0}$.

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{S}\boldsymbol{x}_k}{\|\boldsymbol{S}\boldsymbol{x}_k\|}, \quad k = 0, 1, \dots . \tag{10.53}$$

This means the vector $\boldsymbol{x}_k$ is multiplied by $\boldsymbol{S}$ in every iteration and then normalized, i.e., we always have $\|\boldsymbol{x}_k\| = 1$. This sequence of vectors converges to the eigenvector associated with the largest eigenvalue of $\boldsymbol{S}$. The original Google PageRank algorithm (Page et al., 1999) uses such an algorithm for ranking web pages based on their hyperlinks.

10.5 PCA in High Dimensions

In order to do PCA, we need to compute the data covariance matrix. In D dimensions, the data covariance matrix is a $D \times D$ matrix. Computing the eigenvalues and eigenvectors of this matrix is computationally expensive as it scales cubically in D. Therefore, PCA, as we discussed earlier, will be infeasible in very high dimensions. For example, if our $\boldsymbol{x}_n$ are images with $10,000$ pixels (e.g., 100×100 pixel images), we would need to compute the eigendecomposition of a $10,000 \times 10,000$ covariance matrix. In the following, we provide a solution to this problem for the case that we have substantially fewer data points than dimensions, i.e., $N \ll D$.

Assume we have a data set $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$, $\boldsymbol{x}_n \in \mathbb{R}^D$. Assuming the data is centered, the data covariance matrix is given as

$$\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{D \times D}, \tag{10.54}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]$ is a $D \times N$ matrix whose columns are the data points.

We now assume that $N \ll D$, i.e., the number of data points is smaller than the dimensionality of the data. If there are no duplicate data points

the rank of the covariance matrix $\boldsymbol{S}$ is N, so it has $D - N + 1$ many eigenvalues that are 0. Intuitively, this means that there are some redundancies.

In the following, we will exploit this and turn the $D \times D$ covariance matrix into an $N \times N$ covariance matrix whose eigenvalues are all greater than 0.

In PCA, we ended up with the eigenvector equation

$$\boldsymbol{S}\boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m\,, \quad m = 1, \dots, M\,, \tag{10.55}$$

where $\boldsymbol{b}_m$ is a basis vector of the principal subspace. Let us re-write this equation a bit: With $\boldsymbol{S}$ defined in (10.54), we obtain

$$\boldsymbol{S}\boldsymbol{b}_m = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m\,. \tag{10.56}$$

We now multiply $\boldsymbol{X}^\top \in \mathbb{R}^{N\times D}$ from the left-hand side, which yields

$$\frac{1}{N}\underbrace{\boldsymbol{X}^\top \boldsymbol{X}}_{N\times N}\underbrace{\boldsymbol{X}^\top \boldsymbol{b}_m}_{=:\boldsymbol{c}_m} = \lambda_m \boldsymbol{X}^\top \boldsymbol{b}_m \iff \frac{1}{N}\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{c}_m = \lambda_m \boldsymbol{c}_m\,, \tag{10.57}$$

and we get a new eigenvector/eigenvalue equation: λ_m remains eigenvalue, which confirms our results from Section 4.5.3 that the non-zero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^\top$ equal the non-zero eigenvalues of $\boldsymbol{X}^\top\boldsymbol{X}$. We obtain the eigenvector of the matrix $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{N\times N}$ associated with λ_m as $\boldsymbol{c}_m := \boldsymbol{X}^\top \boldsymbol{b}_m$. Assuming we have no duplicate data points, this matrix has rank N and is invertible. This also implies that $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}$ has the same (non-zero) eigenvalues as the data covariance matrix $\boldsymbol{S}$. But this is now an $N \times N$ matrix, so that we can compute the eigenvalues and eigenvectors much more efficiently than for the original $D \times D$ data covariance matrix.

Now, that we have the eigenvectors of $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}$, we are going to recover the original eigenvectors, which we still need for PCA. Currently, we know the eigenvectors of $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}$. If we left-multiply our eigenvalue/eigenvector equation with $\boldsymbol{X}$, we get

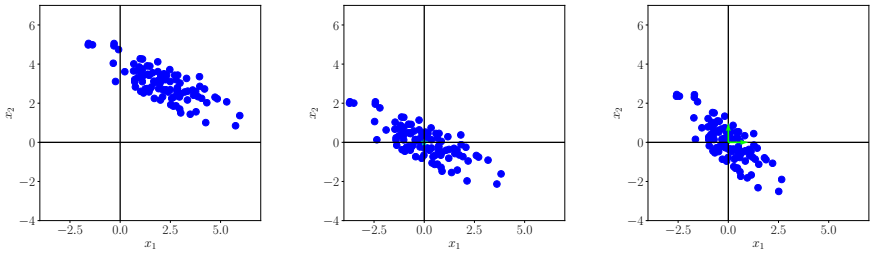
$$\underbrace{\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top}_{\boldsymbol{S}}\boldsymbol{X}\boldsymbol{c}_m = \lambda_m \boldsymbol{X}\boldsymbol{c}_m \tag{10.58}$$

and we recover the data covariance matrix again. This now also means that we recover $\boldsymbol{X}\boldsymbol{c}_m$ as an eigenvector of $\boldsymbol{S}$.

Remark. If we want to apply the PCA algorithm that we discussed in Section 10.6 we need to normalize the eigenvectors $\boldsymbol{X}\boldsymbol{c}_m$ of $\boldsymbol{S}$ so that they have norm 1. ♢

10.6 Key Steps of PCA in Practice

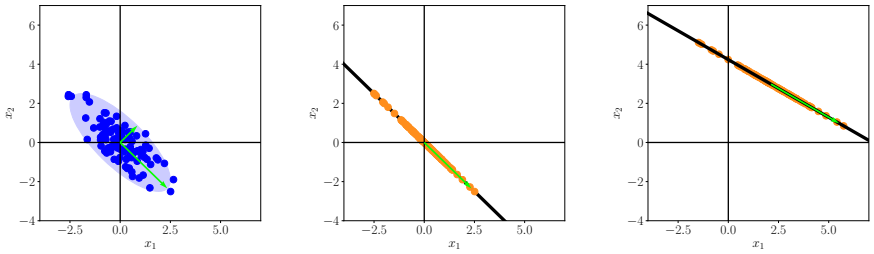
In the following, we will go through the individual steps of PCA using a running example, which is summarized in Figure 10.10. We are given a



(a) Original dataset.

(b) Step 1: Centering by subtracting the mean from each data point.

(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).

(e) Step 4: Project data onto the subspace spanned by the eigenvectors associated with the largest eigenvalues (principal subspace).

(f) Step 5: Undo the standardization and move projected data back into the original data space from (a).

Figure 10.10 Steps of PCA.

two-dimensional data set (Figure 10.10(a)), and we want to use PCA to project it onto a one-dimensional subspace.

1. **Mean subtraction** We start by centering the data by computing the mean $\boldsymbol{\mu}$ of the dataset and subtracting it from every single data point. This ensures that the data set has mean $\mathbf{0}$ (Figure 10.10(b)). Mean subtraction is not strictly necessary but reduces the risk of numerical problems.
2. **Standardization** Divide the data points by the standard deviation σ_d of the dataset for every dimension $d = 1, \ldots, D$. Now the data is unit free, and it has variance 1 along each axis, which is indicated by the two arrows in Figure 10.10(c). This step completes the *standardization* of the data.

standardization

3. **Eigendecomposition of the covariance matrix** Compute the data covariance matrix and its eigenvalues and corresponding eigenvectors. Since the covariance matrix is symmetric, the eigenvectors form an orthogonal basis. In Figure 10.10(d), the eigenvectors are scaled by the magnitude of the corresponding eigenvalue. The longer vector spans the principal subspace, which we denote by U. The data covariance matrix is represented by the ellipse.

4. **Projection** We can project any data point $\boldsymbol{x}_* \in \mathbb{R}^D$ onto the principal subspace: To get this right, we need to standardize $\boldsymbol{x}_*$ using the mean μ_d and standard deviation σ_d of the training data in the dth dimension, respectively, so that

$$x_*^{(d)} \leftarrow \frac{x_*^{(d)} - \mu_d}{\sigma_d}\,, \quad d = 1, \ldots, D\,, \tag{10.59}$$

where $x_*^{(d)}$ is the dth component of $\boldsymbol{x}_*$. We obtain the projection as

$$\tilde{\boldsymbol{x}}_* = \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x}_* \tag{10.60}$$

with coordinates $\boldsymbol{z}_* = \boldsymbol{B}^\top \boldsymbol{x}_*$ with respect to the basis of the principal subspace. Here, $\boldsymbol{B}$ is the matrix that contains the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix as columns.

5. **Moving back to data space** To see our projection in the original data format (i.e., before standardization), we need to undo the standardization (10.59) and multiply by the standard deviation before adding the mean so that we obtain

$$\tilde{x}_*^{(d)} \leftarrow \tilde{x}_*^{(d)}\sigma_d + \mu_d\,, \quad d = 1, \ldots, D\,. \tag{10.61}$$

Figure 10.10(f) illustrates the projection in the original data format.

Example 10.3 (MNIST Digits: Reconstruction)

In the following, we will apply PCA to the MNIST digits dataset, which contains $60,000$ examples of handwritten digits 0–9. Each digit is an image of size 28×28, i.e., it contains 784 pixels so that we can interpret every image in this dataset as a vector $\boldsymbol{x} \in \mathbb{R}^{784}$. Examples of these digits are shown in Figure 10.3. For illustration purposes, we apply PCA to a subset of the MNIST digits, and we focus on the digit '8'. We used 5,389 training images of the digit '8' and determined the principal subspace as detailed in this chapter. We then used the learned projection matrix to reconstruct a set of test images, which is illustrated in Figure 10.11. The first row of Figure 10.11 shows a set of four original digits from the test set. The following rows show reconstructions of exactly these digits when using a principal subspace of dimensions 1, 10, 100, 500, respectively. We can see that even with a single-dimensional principal subspace we get a halfway decent reconstruction of the original digits, which, however, is blurry and generic. With an increasing number of principal components (PCs) the reconstructions become sharper and more details can be accounted for. With 500 principal components, we effectively obtain a near-perfect reconstruction. If we were to choose 784 PCs we would recover the exact digit without any compression loss.

http://yann.lecun.com/exdb/mnist/

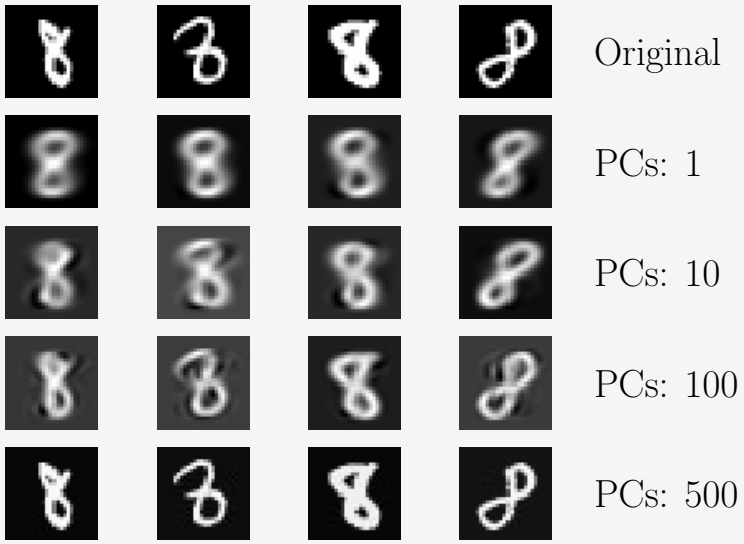


Figure 10.11 Effect of increasing number of principal components on reconstruction.

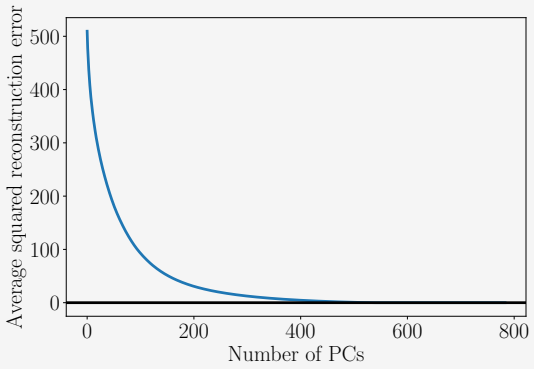


Figure 10.12 Average squared reconstruction error as a function of the number of principal components.

Figure 10.12 shows the average squared reconstruction error, which is

$$\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = \sum_{i=M+1}^{D}\lambda_i\,, \tag{10.62}$$

as a function of the number M of principal components. We can see that the importance of the principal components drops off rapidly, and only marginal gains can be achieved by adding more PCs. With about 550 PCs, we can essentially fully reconstruct the training data that contains the digit '8' (some pixels around the boundaries show no variation across the dataset as they are always black).

10.7 Latent Variable Perspective

In the previous sections, we derived PCA without any notion of a probabilistic model using the maximum-variance and the projection perspectives. On the one hand this approach may be appealing as it allows us to sidestep all the mathematical difficulties that come with probability theory, on the other hand a probabilistic model would offer us more flexibility and useful insights. More specifically, a probabilistic model would

- come with a likelihood function, and we can explicitly deal with noisy observations (which we did not even discuss earlier),
- allow us to do Bayesian model comparison via the marginal likelihood as discussed in Section 8.5,
- view PCA as a generative model, which allows us to simulate new data,
- allow us to make straightforward connections to related algorithms
- deal with data dimensions that are missing at random by applying Bayes' theorem,
- give us a notion of the novelty of a new data point,
- give us a principled way to extend the model, e.g., to a mixture of PCA models,
- have the PCA we derived in earlier sections as a special case,
- allow for a fully Bayesian treatment by marginalizing out the model parameters.

By introducing a continuous-valued latent variable $\boldsymbol{z} \in \mathbb{R}^M$ it is possible to phrase PCA as a probabilistic latent-variable model. Tipping and Bishop (1999) proposed this latent-variable model as *Probabilistic PCA* (PPCA). PPCA addresses most of the issues above, and the PCA solution that we obtained by maximizing the variance in the projected space or by minimizing the reconstruction error is obtained as the special case of maximum likelihood estimation in a noise-free setting.

Probabilistic PCA

10.7.1 Generative Process and Probabilistic Model

In PPCA, we explicitly write down the probabilistic model for linear dimensionality reduction. For this we assume a continuous latent variable $\boldsymbol{z} \in \mathbb{R}^M$ with a standard-Normal prior $p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right)$ and a linear relationship between the latent variables and the observed $\boldsymbol{x}$ data where

$$\boldsymbol{x} = \boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \in \mathbb{R}^D\,, \tag{10.63}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}\right)$ is Gaussian observation noise, $\boldsymbol{B} \in \mathbb{R}^{D\times M}$ and $\boldsymbol{\mu} \in \mathbb{R}^D$ describe the linear/affine mapping from latent to observed variables. Therefore, PPCA links latent and observed variables via

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu},\, \sigma^2\boldsymbol{I}\right)\,. \tag{10.64}$$

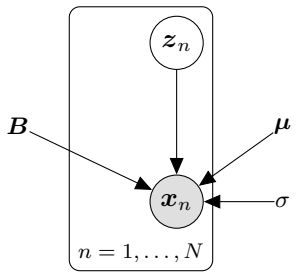


Figure 10.13 Graphical model for probabilistic PCA. The observations $\boldsymbol{x}_n$ explicitly depend on corresponding latent variables $\boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The model parameters $\boldsymbol{B}, \boldsymbol{\mu}$ and the likelihood parameter σ are shared across the dataset.

Overall, PPCA induces the following generative process:

$$\boldsymbol{z}_n \sim \mathcal{N}\big(\boldsymbol{z} \,|\, \boldsymbol{0}, \boldsymbol{I}\big) \tag{10.65}$$

$$\boldsymbol{x}_n \,|\, \boldsymbol{z}_n \sim \mathcal{N}\big(\boldsymbol{x} \,|\, \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I}\big) \tag{10.66}$$

To generate a data point that is typical given the model parameters, we follow an *ancestral sampling* scheme: We first sample a latent variable $\boldsymbol{z}_n$ from $p(\boldsymbol{z})$. Then, we use $\boldsymbol{z}_n$ in (10.64) to sample a data point conditioned on the sampled $\boldsymbol{z}_n$, i.e., $\boldsymbol{x}_n \sim p(\boldsymbol{x} \,|\, \boldsymbol{z}_n, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$.

ancestral sampling

This generative process allows us to write down the probabilistic model (i.e., the joint distribution of all random variables) as

$$p(\boldsymbol{x}, \boldsymbol{z} | \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) = p(\boldsymbol{x} | \boldsymbol{z}, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) p(\boldsymbol{z}) \,, \tag{10.67}$$

which immediately gives rise to the graphical model in Figure 10.13 using the results from Section 8.4.

Remark. Note the direction of the arrow that connects the latent variables $\boldsymbol{z}$ and the observed data $\boldsymbol{x}$: The arrow points from $\boldsymbol{z}$ to $\boldsymbol{x}$, which means that the PPCA model assumes a lower-dimensional latent cause $\boldsymbol{z}$ for high-dimensional observations $\boldsymbol{x}$. In the end, we are obviously interested in finding something out about $\boldsymbol{z}$ given some observations. To get there we will apply Bayesian inference to "invert" the arrow implicitly and go from observations to latent variables. ♢

Example 10.4 (Generating Data from Latent Variables)

Figure 10.14 shows the latent coordinates of the MNIST digits '8' found by PCA when using a two-dimensional principal subspace (blue dots). We can query any vector $\boldsymbol{z}_*$ in this latent space an generate an image $\tilde{\boldsymbol{x}}_* = \boldsymbol{B}\boldsymbol{z}_*$ that resembles the digit '8'. We show eight of such generated images with their corresponding latent space representation. Depending on where we query the latent space, the generated images look different (shape, rotation, size, ...). If we query away from the training data, we see more an more artefacts, e.g., the top-left and top-right digits. Note that the intrinsic dimensionality of these generated images is only two.

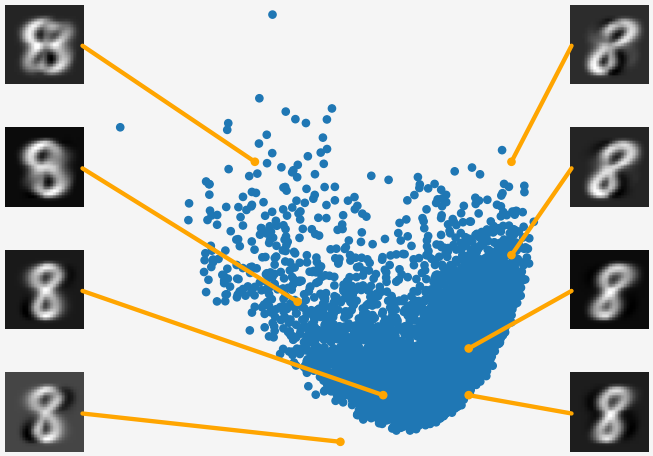


Figure 10.14 Generating new MNIST digits. The latent variables $\boldsymbol{z}$ can be used to generate new data $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z}$. The closer we stay to the training data the more realistic the generated data.

10.7.2 Likelihood and Joint Distribution

Using the results from Chapter 6, we obtain the marginal distribution of the data $\boldsymbol{x}$ by integrating out the latent variable $\boldsymbol{z}$ so that

$$\begin{aligned} p(\boldsymbol{x} \mid \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) &= \int p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \sigma^2) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \\ &= \int \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu},\, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{0},\, \boldsymbol{I}) \mathrm{d}\boldsymbol{z}\,. \end{aligned} \tag{10.68}$$

From Section 6.5, we know that the solution to this integral is a Gaussian distribution with mean

$$\mathbb{E}[\boldsymbol{x}] = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}] + \mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}] = \boldsymbol{\mu} \tag{10.69}$$

and with covariance matrix

$$\begin{aligned} \mathbb{V}[\boldsymbol{x}] &= \mathbb{V}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}] + \mathbb{V}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}] = \mathbb{V}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z}] + \sigma^2 \boldsymbol{I} \\ &= \boldsymbol{B}\mathbb{V}_{\boldsymbol{z}}[\boldsymbol{z}]\boldsymbol{B}^\top + \sigma^2 \boldsymbol{I} = \boldsymbol{B}\boldsymbol{B}^\top + \sigma^2 \boldsymbol{I}\,. \end{aligned} \tag{10.70}$$

The marginal distribution in (10.68) is the *PPCA likelihood*, which we can use for maximum likelihood or MAP estimation of the model parameters. PPCA likelihood

Remark. Although the conditional distribution in (10.64) is also a likelihood, we cannot use it for maximum likelihood estimation as it still depends on the latent variables. The likelihood function we require for maximum likelihood (or MAP) estimation should only be a function of the data $\boldsymbol{x}$ and the model parameters, but not on the latent variables. ♢

From Section 6.5 we also know that the joint distribution of a Gaussian random variable $\boldsymbol{z}$ and a linear/affine transformation $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{z}$ of it are jointly Gaussian distributed. We already know the marginals $p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z} \,|\, \boldsymbol{0},\, \boldsymbol{I}\right)$ and $p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{\mu},\, \boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I}\right)$. The missing cross-covariance is given as

$$\operatorname{Cov}[\boldsymbol{x}, \boldsymbol{z}] = \operatorname{Cov}_{\boldsymbol{z}}[\boldsymbol{B}\boldsymbol{z} + \boldsymbol{\mu}] = \boldsymbol{B}\operatorname{Cov}_{\boldsymbol{z}}[\boldsymbol{z}, \boldsymbol{z}] = \boldsymbol{B}\,. \tag{10.71}$$

Therefore, the probabilistic model of PPCA, i.e., the joint distribution of latent and observed random variables is explicitly given by

$$p(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{z}\end{bmatrix} \,\middle|\, \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I} & \boldsymbol{B}\\ \boldsymbol{B}^\top & \boldsymbol{I}\end{bmatrix}\right), \tag{10.72}$$

with a mean vector of length $D + M$ and a covariance matrix of size $(D + M) \times (D + M)$.

10.7.3 Posterior Distribution

The joint Gaussian distribution $p(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$ in (10.72) allows us to determine the posterior distribution $p(\boldsymbol{z} \,|\, \boldsymbol{x})$ immediately by applying the rules of Gaussian conditioning from Section 6.5.1. The posterior distribution of the latent variable given an observation $\boldsymbol{x}$ is then

$$p(\boldsymbol{z} \,|\, \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z} \,|\, \boldsymbol{m},\, \boldsymbol{C}\right), \tag{10.73a}$$

$$\boldsymbol{m} = \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{x} - \boldsymbol{\mu}), \tag{10.73b}$$

$$\boldsymbol{C} = \boldsymbol{I} - \boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{B}\,. \tag{10.73c}$$

Note that the posterior covariance does not depend on the observation $\boldsymbol{x}$.

For a new observation $\boldsymbol{x}_*$ in data space, we can use (10.73) to determine the posterior distribution of the corresponding latent variable $\boldsymbol{z}_*$. The covariance matrix $\boldsymbol{C}$ allows us to assess how confident the embedding is. A covariance matrix $\boldsymbol{C}$ with a small determinant (which measures volumes) tells us that the latent embedding $\boldsymbol{z}_*$ is fairly certain. If we obtain a posterior distribution $p(\boldsymbol{z}_* \,|\, \boldsymbol{x}_*)$ with much variance, we may be faced with an outlier. However, we can explore this posterior distribution to understand what other data points $\boldsymbol{x}$ are plausible under this posterior. To do this, we can exploit PPCA's generative process. The generative process underlying PPCA allows us to explore the posterior distribution on the latent variables by generating new data that are plausible under this posterior. This can be achieved as follows:

1. Sample a latent variable $\boldsymbol{z}_* \sim p(\boldsymbol{z} \,|\, \boldsymbol{x}_*)$ from the posterior distribution over the latent variables (10.73)
2. Sample a reconstructed vector $\tilde{\boldsymbol{x}}_* \sim p(\boldsymbol{x} \,|\, \boldsymbol{z}_*, \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$ from (10.64)

If we repeat this process many times, we can explore the posterior distribution (10.73) on the latent variables $\boldsymbol{z}_*$ and its implications on the

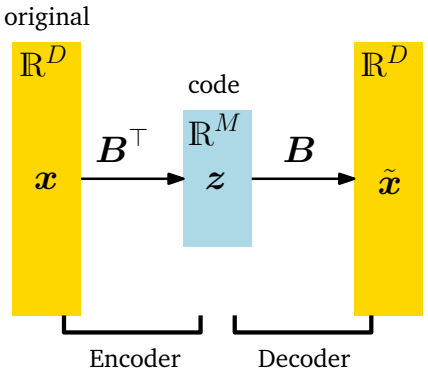


Figure 10.15 PCA can be viewed as a linear auto-encoder. It encodes the high-dimensional data $\boldsymbol{x}$ into a lower-dimensional representation (code) $\boldsymbol{z} \in \mathbb{R}^M$ and decode $\boldsymbol{z}$ using a decoder. The decoded vector $\tilde{\boldsymbol{x}}$ is the orthogonal projection of the original data $\boldsymbol{x}$ onto the M-dimensional principal subspace.

observed data. The sampling process effectively hypothesizes data, which is plausible under the posterior distribution.

10.8 Further Reading

We derived PCA from two perspectives: a) maximizing the variance in the projected space; b) minimizing the average reconstruction error. However, PCA can also be interpreted from different perspectives. Let us re-cap what we have done: We took high-dimensional data $\boldsymbol{x} \in \mathbb{R}^D$ and used a matrix $\boldsymbol{B}^\top$ to find a lower-dimensional representation $\boldsymbol{z} \in \mathbb{R}^M$. The columns of $\boldsymbol{B}$ are the eigenvectors of the data covariance matrix $\boldsymbol{S}$ that are associated with the largest eigenvalues. Once we have a low-dimensional representation $\boldsymbol{z}$, we can get a high-dimensional version of it (in the original data space) as $\boldsymbol{x} \approx \tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z} = \boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{x} \in \mathbb{R}^D$, where $\boldsymbol{B}\boldsymbol{B}^\top$ is a projection matrix.

We can also think of PCA as a linear *auto-encoder* as illustrated in Figure 10.15. An auto-encoder encodes the data $\boldsymbol{x}_n \in \mathbb{R}^D$ to a *code* $\boldsymbol{z}_n \in \mathbb{R}^M$ and tries to decode it to a $\tilde{\boldsymbol{x}}_n$ similar to $\boldsymbol{x}_n$. The mapping from the data to the code is called the *encoder*, the mapping from the code back to the original data space is called the *decoder*. If we consider linear mappings where the code is given by $\boldsymbol{z}_n = \boldsymbol{B}^\top\boldsymbol{x}_n \in \mathbb{R}^M$ and we are interested in minimizing the average squared error between the data $\boldsymbol{x}_n$ and its reconstruction $\tilde{\boldsymbol{x}}_n = \boldsymbol{B}\boldsymbol{z}_n$, $n = 1, \dots, N$, we obtain

auto-encoder
code
encoder
decoder

$$\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{x}_n - \boldsymbol{B}^\top\boldsymbol{B}\boldsymbol{x}_n\right\|^2 . \tag{10.74}$$

This means we end up with the same objective function as in (10.30) that we discussed in Section 10.3 so that we obtain the PCA solution when we minimize the squared auto-encoding loss. If we replace the linear mapping of PCA with a nonlinear mapping, we get a nonlinear auto-encoder. A prominent example of this is a deep auto-encoder where the linear functions are replaced with deep neural networks. In this context, the encoder

recognition network
inference network
generator

is also know as *recognition network* or *inference network*, whereas the decoder is also called a *generator*.

The code is a compressed version of the original data.

Another interpretation of PCA is related to information theory. We can think of the code as a smaller or compressed version of the original data point. When we reconstruct our original data using the code, we do not get the exact data point back, but a slightly distorted or noisy version of it. This means that our compression is "lossy". Intuitively we want to maximize the correlation between the original data and the lower-dimensional code. More formally, this is related to the mutual information. We would then get the same solution to PCA we discussed in Section 10.3 by maximizing the mutual information, a core concept in information theory (MacKay, 2003a).

In our discussion on PPCA, we assumed that the parameters of the model, i.e., $\boldsymbol{B}, \boldsymbol{\mu}$ and the likelihood parameter σ^2 are known. Tipping and Bishop (1999) describe how to derive maximum likelihood estimates for these parameter in the PPCA setting (note that we use a different notation in this chapter). The maximum likelihood parameters, when projecting D-dimensional data onto an M-dimensional subspace, are given by

The matrix $\boldsymbol{\Lambda} - \sigma^2\boldsymbol{I}$ in (10.76) is guaranteed to be positive semi-definite as the smallest eigenvalue of the data covariance matrix is bounded from below by the noise variance σ^2.

$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\,, \tag{10.75}$$

$$\boldsymbol{B}_{\text{ML}} = \boldsymbol{T}(\boldsymbol{\Lambda} - \sigma^2\boldsymbol{I})^{\frac{1}{2}}\boldsymbol{R}\,, \tag{10.76}$$

$$\sigma^2_{\text{ML}} = \frac{1}{D-M}\sum_{j=M+1}^{D}\lambda_j\,, \tag{10.77}$$

where $\boldsymbol{T} \in \mathbb{R}^{D\times M}$ contains M eigenvectors of the data covariance matrix, $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_M) \in \mathbb{R}^{M\times M}$ is a diagonal matrix with the eigenvalues associated with the principal axes on its diagonal, and $\boldsymbol{R} \in \mathbb{R}^{M\times M}$ is an arbitrary orthogonal matrix. The maximum likelihood solution $\boldsymbol{B}_{\text{ML}}$ is unique up to a arbitrary orthogonal transformation, e.g., we can right-multiply $\boldsymbol{B}_{\text{ML}}$ with any rotation matrix $\boldsymbol{R}$ so that (10.76) essentially is a singular value decomposition (see Section 4.5). An outline of the proof is given by Tipping and Bishop (1999).

The maximum likelihood estimate for $\boldsymbol{\mu}$ given in (10.75) is the sample mean of the data. The maximum likelihood estimator for the observation noise variance σ^2 given in (10.77) is the average variance in the orthogonal complement of the principal subspace, i.e., the average leftover variance that we cannot capture with the first M principal components are treated as observation noise.

In the noise-free limit where $\sigma \to 0$, PPCA and PCA provide identical solutions: Since the data covariance matrix $\boldsymbol{S}$ is symmetric, it can be diagonalized (see Section 4.4), i.e., there exists a matrix $\boldsymbol{T}$ of eigenvectors

of $\boldsymbol{S}$ so that

$$\boldsymbol{S} = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1}\,. \tag{10.78}$$

In the PPCA model, the data covariance matrix is the covariance matrix of the likelihood $p(\boldsymbol{X} \mid \boldsymbol{B}, \boldsymbol{\mu}, \sigma^2)$, which is $\boldsymbol{B}\boldsymbol{B}^\top + \sigma^2\boldsymbol{I}$, see (10.70). For $\sigma \to 0$, we obtain $\boldsymbol{B}\boldsymbol{B}^\top$ so that this data covariance must equal the PCA data covariance (and its factorization given in (10.78)) so that

$$\mathrm{Cov}[\boldsymbol{X}] = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1} = \boldsymbol{B}\boldsymbol{B}^\top \iff \boldsymbol{B} = \boldsymbol{T}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{R}\,, \tag{10.79}$$

so that we obtain exactly the maximum likelihood estimate in (10.76) for $\sigma = 0$.

From (10.76) and (10.78) it becomes clear that (P)PCA performs a decomposition of the data covariance matrix.

In a streaming setting, where data arrives sequentially, it is recommended to use the iterative Expectation Maximization (EM) algorithm for maximum likelihood estimation (Roweis, 1998).

To determine the dimensionality of the latent variables (the length of the code, the dimensionality of the lower-dimensional subspace onto which we project the data) Gavish and Donoho (2014) suggest the heuristic that, if we can estimate the noise variance σ^2 of the data, we should discard all singular values smaller than $\frac{4\sigma\sqrt{D}}{\sqrt{3}}$. Alternatively, we can use cross validation or the Bayesian model selection criteria (discussed in Section 8.5.2) to determine the true dimensionality of the data (Minka, 2001).

Similar to our discussion on linear regression in Chapter 9, we can place a prior distribution on the parameters of the model and integrate them out, thereby avoiding a) point estimates of the parameters and the issues that come with these point estimates (see Section 8.5) and b) allowing for an automatic selection of the appropriate dimensionality M of the latent space. In this *Bayesian PCA*, which was proposed by Bishop (1999), places a (hierarchical) prior $p(\boldsymbol{\mu}, \boldsymbol{B}, \sigma^2)$ on the model parameters. The generative process allows us to integrate the model parameters out instead of conditioning on them, which addresses overfitting issues. Since this integration is analytically intractable, Bishop (1999) proposes to use approximate inference methods, such as MCMC or variational inference. We refer to the work by Gilks et al. (1996) and Blei et al. (2017) for more details on these approximate inference techniques.

Bayesian PCA

In PPCA, we considered the linear model $\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{\epsilon}$ with $p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, i.e., all observation dimensions are affected by the same amount of noise. If we allow each observation dimension d to have a different variance σ_d^2 we obtain *factor analysis* (FA) (Spearman, 1904; Bartholomew et al., 2011). This means, FA gives the likelihood some more flexibility than PPCA, but still forces the data to be explained by the model parameters $\boldsymbol{B}$, $\boldsymbol{\mu}$. However, FA no longer allows for a closed-form solution to maximum likelihood so that we need to use an

factor analysis

An overly flexible likelihood would be able to explain more than just the noise.

iterative scheme, such as the EM algorithm, to estimate the model parameters. While in PPCA all stationary points are global optima, this no longer holds for FA. Compared to PPCA, FA does not change if we scale the data, but it does return different solutions if we rotate the data.

Independent Component Analysis

Independent Component Analysis (ICA) is also closely related to PCA. Starting again with the model $\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{\epsilon}$ we now change the prior on $\boldsymbol{z}_n$ to non-Gaussian distributions. ICA can be used for *blind-source separation*. Imagine you are in a busy train station with many people talking. Your ears play the role of microphones, and they linearly mix different speech signals in the train station. The goal of blind-source separation is to identify the constituent parts of the mixed signals. As discussed above in the context of maximum likelihood estimation for PPCA, the original PCA solution is invariant to any rotation. Therefore, PCA can identify the best lower-dimensional subspace in which the signals live, but not the signals themselves (Murphy, 2012). ICA addresses this issue by modifying the prior distribution $p(\boldsymbol{z})$ on the latent sources to require non-Gaussian priors $p(\boldsymbol{z})$. We refer to the book by Murphy (2012) for more details on ICA.

blind-source separation

PCA, factor analysis and ICA are three examples for dimensionality reduction with linear models. Cunningham and Ghahramani (2015) provide a broader survey of linear dimensionality reduction.

The (P)PCA model we discussed here allows for several important extensions. In Section 10.5, we explained how to do PCA when the input dimensionality D is significantly greater than the number N of data points. By exploiting the insight that PCA can be performed by computing (many) inner products, this idea can be pushed to the extreme by considering infinite-dimensional features. The *kernel trick* is the basis of *kernel PCA* and allows us to implicitly compute inner products between infinite-dimensional features (Schölkopf et al., 1998; Schölkopf and Smola, 2002).

kernel trick
kernel PCA

There are nonlinear dimensionality reduction techniques that are derived from PCA (Burges (2010) provide a good overview). The auto-encoder perspective of PCA that we discussed above can be used to render PCA as a special case of a *deep auto-encoder*. In the deep auto-encoder, both the encoder and the decoder are represented by multi-layer feedforward neural networks, which themselves are nonlinear mappings. If we set the activation functions in these neural networks to be the identity, the model becomes equivalent to PCA. A different approach to nonlinear dimensionality reduction is the *Gaussian Process Latent Variable Model* (GP-LVM) proposed by Lawrence (2005). The GP-LVM starts off with the latent-variable perspective that we used to derive PPCA and replaces the linear relationship between the latent variables $\boldsymbol{z}$ and the observations $\boldsymbol{x}$ with a Gaussian process (GP). Instead of estimating the parameters of the mapping (as we do in PPCA), the GP-LVM marginalizes out the model parameters and makes point estimates of the latent variables $\boldsymbol{z}$. Similar to Bayesian PCA, the *Bayesian GP-LVM* proposed by Titsias and Lawrence (2010) maintains

deep auto-encoder

Gaussian Process Latent Variable Model

Bayesian GP-LVM

a distribution on the latent variables $\boldsymbol{z}$ and uses approximate inference to integrate them out as well.

11

Density Estimation with Gaussian Mixture Models

In earlier chapters, we covered already two fundamental problems in machine learning: regression (Chapter 9 and dimensionality reduction (Chapter 10). In this chapter, we will have a look at a third pillar of machine learning: density estimation. On our journey, we introduce important concepts, such as the EM algorithm and a latent variable perspective of density estimation with mixture models.

When we apply machine learning to data we often aim to represent data in some way. A straightforward way is to take the data points themselves as the representation of the data, see Figure 11.1 for an example. However, this approach may be unhelpful if the dataset is huge or if we are interested in representing characteristics of the data. In density estimation, we represent the data compactly using a density, e.g., a Gaussian or Beta distribution. For example, we may be looking for the mean and variance of a data set in order to represent the data compactly using a Gaussian distribution. The mean and variance can be found using tools we discussed in Section 8.2: maximum likelihood or maximum-a-posteriori estimation. We can then use the mean and variance of this Gaussian to represent the distribution underlying the data, i.e., we think of the dataset to be a typical realization from this distribution if we were to sample from it.

In practice, the Gaussian (or similarly all other distributions we encoun-

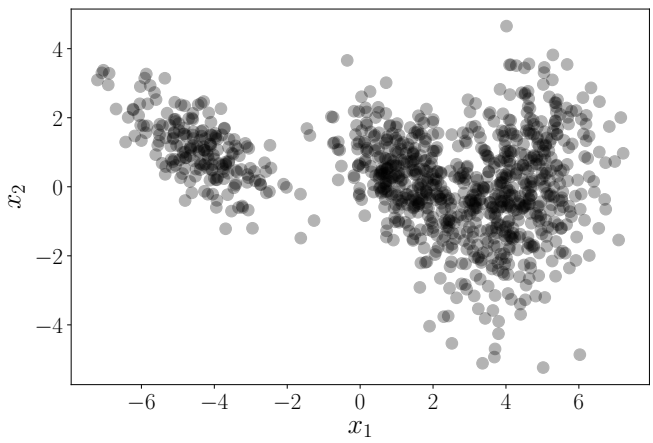


Figure 11.1 Two-dimensional data set that cannot be meaningfully represented by a Gaussian.

tered so far) have limited modeling capabilities. For example, a Gaussian approximation of the density that generated the data in Figure 11.1 would be a poor approximation. In the following, we will look at a more expressive family of distributions, which we can use for density estimation: *mixture models*.

mixture models

Mixture models can be used to describe a distribution $p(\boldsymbol{x})$ by a convex combination of K simple (base) distributions

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}) \tag{11.1}$$

$$0 \leqslant \pi_k \leqslant 1\,, \quad \sum_{k=1}^{K} \pi_k = 1\,, \tag{11.2}$$

where the components p_k are members of a family of basic distributions, e.g., Gaussians, Bernoullis or Gammas, and the π_k are *mixture weights*. Mixture models are more expressive than the corresponding base distributions because they allow for multimodal data representations, i.e., they can describe datasets with multiple "clusters", such as the example in Figure 11.1.

mixture weights

In the following, we will focus on Gaussian mixture models (GMMs), where the basic distributions are Gaussians. For a given dataset, we aim to maximize the likelihood of the model parameters to train the GMM. For this purpose we will use results from Chapter 5, Section 7.2 and Chapter 6. However, unlike other application we discussed earlier (linear regression or PCA), we will not find a closed-form maximum likelihood solution. Instead, we will arrive at a set of dependent simultaneous equations, which we can only solve iteratively.

11.1 Gaussian Mixture Model

Gaussian mixture model

A *Gaussian mixture model* is a density model where we combine a finite number of K Gaussian distributions $\mathcal{N}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$ so that

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{11.3}$$

$$0 \leqslant \pi_k \leqslant 1\,, \quad \sum_{k=1}^{K} \pi_k = 1\,. \tag{11.4}$$

This convex combination of Gaussian distribution gives us significantly more flexibility for modeling complex densities than a simple Gaussian distribution (which we recover from (11.3) for $K = 1$). An illustration is given in Figure 11.2. Here, the mixture density is given as

$$p(x) = 0.5\mathcal{N}\left(x \,|\, -2, \tfrac{1}{2}\right) + 0.2\mathcal{N}\left(x \,|\, 1, 2\right) + 0.3\mathcal{N}\left(x \,|\, 4, 1\right). \tag{11.5}$$

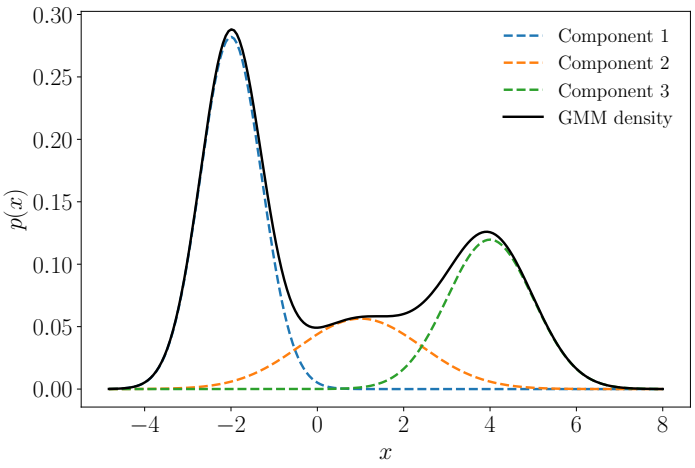


Figure 11.2 Gaussian mixture model. The Gaussian mixture distribution (black) is composed of a convex combination of Gaussian distributions and is more expressive than any individual component. Dashed lines represent the weighted Gaussian components.

11.2 Parameter Learning via Maximum Likelihood

Assume we are given a data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ where $\boldsymbol{x}_n$, $n = 1, \dots, N$ are drawn i.i.d. from an unknown distribution $p(\boldsymbol{x})$. Our objective is to find a good approximation/representation of this unknown distribution $p(\boldsymbol{x})$ by means of a Gaussian mixture model (GMM) with K mixture components. The parameters of the GMM are the K means $\boldsymbol{\mu}_k$, the covariances $\boldsymbol{\Sigma}_k$ and mixture weights π_k. We summarize all these free parameters in $\boldsymbol{\theta} := \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k,\ k = 1, \dots, K\}$.

Example 11.1 (Initial setting)

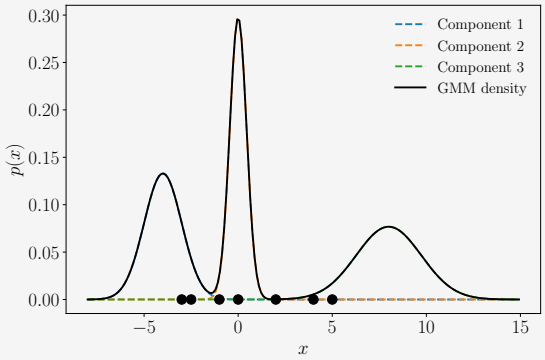


Figure 11.3 Initial setting: GMM (black) with mixture three mixture components (dashed) and seven data points (discs).

Throughout this chapter, we will have a simple running example that helps us illustrate and visualize important concepts.

We will look at a one-dimensional data set $\boldsymbol{X} = [-3, -2.5, -1, 0, 2, 4, 5]$ consisting of seven data points. We wish to find a GMM with $K = 3$ components that models the data. We initialize the individual components

as

$$p_1(x) = \mathcal{N}\big(x \,|\, -4,\, 1\big) \tag{11.6}$$
$$p_2(x) = \mathcal{N}\big(x \,|\, 0,\, 0.2\big) \tag{11.7}$$
$$p_3(x) = \mathcal{N}\big(x \,|\, 8,\, 3\big) \tag{11.8}$$

and assign them equal weights $\pi_1 = \pi_2 = \pi_3 = \frac{1}{3}$. The corresponding model (and the data points) are shown in Figure 11.3.

In the following, we detail how to obtain a maximum likelihood estimate $\boldsymbol{\theta}_{\text{ML}}$ of the model parameters $\boldsymbol{\theta}$. We start by writing down the likelihood, i.e., the probability of the data given the parameters. We exploit our i.i.d. assumption, which leads to the factorized likelihood

$$p(\boldsymbol{X} \,|\, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})\,, \qquad p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big) \tag{11.9}$$

where every individual likelihood term $p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})$ is a Gaussian mixture density. Then, we obtain the log-likelihood as

$$\log p(\boldsymbol{X} \,|\, \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta}) = \underbrace{\sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\big)}_{=:\mathcal{L}}\,. \tag{11.10}$$

We aim to find parameters $\boldsymbol{\theta}^*_{\text{ML}}$ that maximize the log-likelihood $\mathcal{L}$ defined in (11.10). Our "normal" procedure would be to compute the gradient $d\mathcal{L}/d\boldsymbol{\theta}$ of the log-likelihood with respect to the model parameters $\boldsymbol{\theta}$, set it to $\mathbf{0}$ and solve for $\boldsymbol{\theta}$. However, unlike our previous examples for maximum likelihood estimation (e.g., when we discussed linear regression in Section 9.2), we cannot obtain a closed-form solution. If we were to consider a single Gaussian as the desired density, the sum over k in (11.10) vanishes, and the log can be applied directly to the Gaussian component, such that we get

$$\log \mathcal{N}\big(\boldsymbol{x} \,|\, \boldsymbol{\mu},\, \boldsymbol{\Sigma}\big) = -\tfrac{D}{2}\log(2\pi) - \tfrac{1}{2}\log\det(\boldsymbol{\Sigma}) - \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\,. \tag{11.11}$$

This simple form allows us find closed-form maximum likelihood estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, as discussed in Chapter 8. However, in (11.10), we cannot move the $\log$ into the sum over k so that we cannot obtain a simple closed-form maximum likelihood solution. However, we can exploit an iterative scheme to find good model parameters $\boldsymbol{\theta}_{\text{ML}}$: the EM algorithm.

Any local optimum of a function exhibits the property that its gradient with respect to the parameters must vanish (necessary condition), see Chapter 7. In our case, we obtain the following necessary conditions when

we optimize the log-likelihood in (11.10) with respect to the GMM parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \mathbf{0} \iff \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \mathbf{0}\,, \tag{11.12}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \mathbf{0} \iff \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} = \mathbf{0}\,, \tag{11.13}$$

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \mathbf{0} \iff \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \pi_k} = \mathbf{0}\,. \tag{11.14}$$

For all three necessary conditions, by applying the chain rule (see Section 5.2.2), we require partial derivatives of the form

$$\frac{\partial \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{p(\boldsymbol{x}_n \mid \boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{11.15}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k,\ k = 1, \ldots, K\}$ comprises all model parameters and

$$\frac{1}{p(\boldsymbol{x}_n \mid \boldsymbol{\theta})} = \frac{1}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j,\ \boldsymbol{\Sigma}_j)}\,. \tag{11.16}$$

5996 In the following, we will compute the partial derivatives (11.12)–(11.14).

Theorem 11.1 (Update of the GMM Means)**.** *The update of the mean parameters $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$ of the GMM is given by*

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}} = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n\,, \tag{11.17}$$

where we define

$$r_{nk} := \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j,\ \boldsymbol{\Sigma}_j)}\,, \tag{11.18}$$

$$N_k := \sum_{n=1}^{N} r_{nk}\,. \tag{11.19}$$

Proof From (11.15), we see that the gradient of the log-likelihood with respect to the mean parameters $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$ requires us to compute the partial derivative

$$\frac{\partial p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \sum_{j=1}^{K} \pi_j \frac{\partial \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j,\ \boldsymbol{\Sigma}_j)}{\partial \boldsymbol{\mu}_k} = \pi_k \frac{\partial \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} \tag{11.20}$$

$$= \pi_k (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k) \tag{11.21}$$

5997 where we exploited that only the kth mixture component depends on $\boldsymbol{\mu}_k$.

We use our result from (11.21) in (11.15) and put everything together so that the desired partial derivative of $\mathcal{L}$ with respect to $\boldsymbol{\mu}_k$ is given as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n \mid \boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} \tag{11.22}$$

$$= \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \underbrace{\frac{\pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{=r_{nk}} \tag{11.23}$$

$$= \sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} . \tag{11.24}$$

Here, we used the identity from (11.16) and the result of the partial derivative in (11.21) to get to the second row. The values r_{nk} are often called *responsibilities*.

responsibilities

The responsibilities are closely related to the likelihood.

Remark. The responsibility r_{nk} of the kth mixture component for data point $\boldsymbol{x}_n$ is proportional to the likelihood

$$p(\boldsymbol{x}_n \mid \pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{11.25}$$

of the mixture component given the data point (the denominator in the definition of r_{nk} is constant for all mixture components and serves as a normalizer). Therefore, mixture components have a high responsibility for a data point when the data point could be a plausible sample from that mixture component. ♢

From the definition of r_{nk} in (11.18) it is clear that $[r_{n1}, \ldots, r_{nK}]^\top$ is a probability vector, i.e., $\sum_k r_{nk} = 1$ with $r_{nk} \geqslant 0$. This probability vector distributes probability mass among the K mixture component, and, intuitively, every r_{nk} expresses the probability that $\boldsymbol{x}_n$ has been generated by the kth mixture component.

We now solve for $\boldsymbol{\mu}_k$ so that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \mathbf{0}^\top$ and obtain

$$\sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n = \sum_{n=1}^{N} r_{nk} \boldsymbol{\mu}_k \iff \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n}{\boxed{\sum_{n=1}^{N} r_{nk}}} = \frac{1}{\boxed{N_k}} \sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n , \tag{11.26}$$

where we defined

$$N_k := \sum_{n=1}^{N} r_{nk} . \tag{11.27}$$

as the total responsibility of the kth mixture component across the entire dataset. This concludes the proof of Theorem 11.1. □

Intuitively, (11.17) can be interpreted as a Monte-Carlo estimate of the mean of weighted data points $\boldsymbol{x}_n$ where every $\boldsymbol{x}_n$ is weighted by the

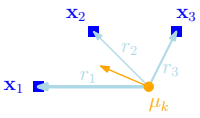
responsibility r_{nk} of the kth cluster for $\boldsymbol{x}_n$. Therefore, the mean $\boldsymbol{\mu}_k$ is pulled toward a data point $\boldsymbol{x}_n$ with strength given by r_{nk}. Intuitively, the means are pulled stronger toward data points for which the corresponding mixture component has a high responsibility, i.e., a high likelihood. Figure 11.4 illustrates this. We can also interpret the mean update in (11.17) as the expected value of all data points under the distribution given by

$$\boldsymbol{r}_k := [r_{1k}, \ldots, r_{Nk}]^\top / N_k \,, \tag{11.28}$$

which is a normalized probability vector, i.e.,

$$\boldsymbol{\mu}_k \leftarrow \mathbb{E}_{\boldsymbol{r}_k}[\boldsymbol{X}] \,. \tag{11.29}$$

Figure 11.4 Update of the mean parameter of mixture component in a GMM. The mean $\boldsymbol{\mu}$ is being pulled toward individual data points with the weights given by the corresponding responsibilities. The mean update is then a weighted average of the data points.



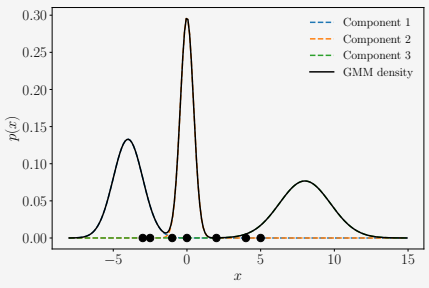
Example 11.2 (Responsibilities)

For our example from Figure 11.3 we compute the responsibilities r_{nk}

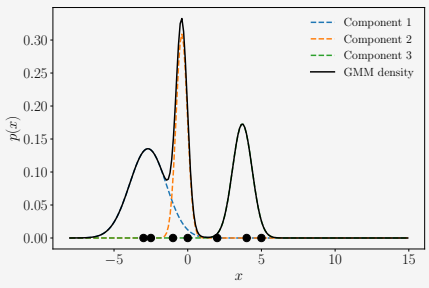
$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.057 & 0.943 & 0.0 \\ 0.001 & 0.999 & 0.0 \\ 0.0 & 0.066 & 0.934 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \in \mathbb{R}^{N \times K} \,. \tag{11.30}$$

Here, the nth row tells us the responsibilities of all mixture components for x_n. The sum of all K responsibilities for a data point (sum of every row) is 1. The kth column gives us an overview of the responsibility of the kth mixture component. We can see that the third mixture component (third column) is not responsible for any of the first four data points, but takes much responsibility of the remaining data points. The sum of all entries of a column gives us the values N_k, i.e., the total responsibility of the kth mixture component. In our example, we get $N_1 = 2.058$, $N_2 = 2.008$, $N_3 = 2.934$.

Example 11.3 (Mean Updates)



(a) GMM density and individual components prior to updating the mean values.



(b) GMM density and individual components after updating the mean values.

Figure 11.5 Effect of updating the mean values in a GMM. (a) GMM before updating the mean values; (b) GMM after updating the mean values μ_k while retaining the variances and mixture weights.

In our example from Figure 11.3, the mean values are updated as follows:

$$\mu_1 : -4 \to -2.7 \tag{11.31}$$
$$\mu_2 : 0 \to -0.4 \tag{11.32}$$
$$\mu_3 : 8 \to 3.7 \tag{11.33}$$

Here, we see that the means of the first and third mixture component move toward the regime of the data, whereas the mean of the second component does not change so dramatically. Figure 11.5 illustrates this change, where Figure 11.5(a) shows the GMM density prior to updating the means and Figure 11.5(b) shows the GMM density after updating the mean values μ_k.

The update of the mean parameters in (11.17) look fairly straightforward. However, note that the responsibilities r_{nk} are a function of $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ for all $j = 1, \ldots, K$, such that the updates in (11.17) depend on all parameters of the GMM, and a closed-form solution, which we obtained for linear regression in Section 9.2 or PCA in Chapter 10, cannot be obtained.

Theorem 11.2 (Updates of the GMM Covariances)**.** *The update of the covariance parameters $\boldsymbol{\Sigma}_k$, $k = 1, \ldots, K$ of the GMM is given by*

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top , \tag{11.34}$$

where r_{nk} and N_k are defined in (11.18) *and* (11.19)*, respectively.*

Proof To prove Theorem 11.2 our approach is to compute the partial derivatives of the log-likelihood $\mathcal{L}$ with respect to the covariances $\boldsymbol{\Sigma}_k$, set them to $\mathbf{0}$ and solve for $\boldsymbol{\Sigma}_k$. We start with our general approach

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} . \tag{11.35}$$

We already know $1/p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})$ from (11.16). To obtain the remaining partial derivative $\partial p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta}) / \partial \boldsymbol{\Sigma}_k$, we write down the definition of the Gaussian distribution $p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})$, see (11.9), and drop all terms but the kth. We then obtain

$$\frac{\partial p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} \tag{11.36a}$$
$$= \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \left((2\pi)^{-\frac{D}{2}} \det(\boldsymbol{\Sigma}_k)^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right) \right) \tag{11.36b}$$

$$= \pi_k (2\pi)^{-\frac{D}{2}} \left[\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \det(\boldsymbol{\Sigma}_k)^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\right) \right.$$
$$\left. + \det(\boldsymbol{\Sigma}_k)^{-\frac{1}{2}} \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\right) \right] . \quad (11.36c)$$

We now use the identities

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \det(\boldsymbol{\Sigma}_k)^{-\frac{1}{2}} = -\frac{1}{2} \det(\boldsymbol{\Sigma}_k)^{-\frac{1}{2}} \boldsymbol{\Sigma}_k^{-1} , \quad (11.37)$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) = -\boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \quad (11.38)$$

and obtain (after some re-arranging) the desired partial derivative required in (11.35) as

$$\frac{\partial p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} = \pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$\times \left[-\tfrac{1}{2}(\boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}) \right] . \quad (11.39)$$

Putting everything together, the partial derivative of the log-likelihood with respect to $\boldsymbol{\Sigma}_k$ is given by

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n \mid \boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_n \mid \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} \quad (11.40a)$$
$$= \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{=r_{nk}}$$
$$\times \left[-\tfrac{1}{2}(\boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}) \right] \quad (11.40b)$$
$$= -\frac{1}{2} \sum_{n=1}^{N} r_{nk} (\boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}) \quad (11.40c)$$
$$= -\frac{1}{2} \boldsymbol{\Sigma}_k^{-1} \underbrace{\sum_{n=1}^{N} r_{nk}}_{=N_k} + \frac{1}{2} \boldsymbol{\Sigma}_k^{-1} \left(\sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \right) \boldsymbol{\Sigma}_k^{-1} . \quad (11.40d)$$

We see that the responsibilities r_{nk} also appear in this partial derivative. Setting this partial derivative to $\mathbf{0}$, we obtain the necessary optimality condition

$$N_k \boldsymbol{\Sigma}_k^{-1} = \boldsymbol{\Sigma}_k^{-1} \left(\sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \right) \boldsymbol{\Sigma}_k^{-1} \quad (11.41a)$$
$$\iff N_k \boldsymbol{I} = \left(\sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \right) \boldsymbol{\Sigma}_k^{-1} \quad (11.41b)$$

$$\iff \boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top , \tag{11.41c}$$

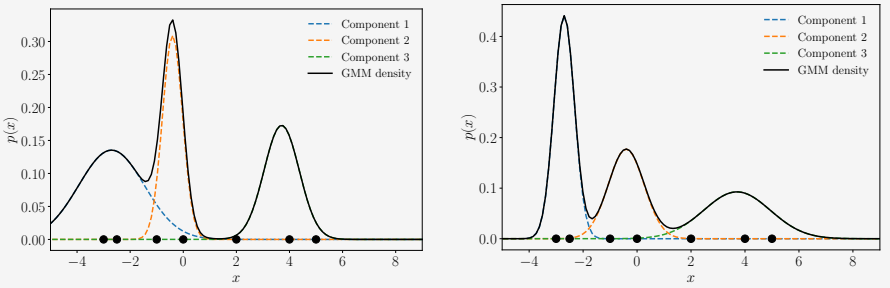
which gives us a simple update rule for $\boldsymbol{\Sigma}_k$ for $k = 1, \ldots, K$ and proves Theorem 11.2. □

Similar to the update of $\boldsymbol{\mu}_k$ in (11.17), we can interpret the update of the covariance in (11.34) as an expected value

$$\mathbb{E}_{\boldsymbol{r}_k}[\tilde{\boldsymbol{X}}_k^\top \tilde{\boldsymbol{X}}_k] \tag{11.42}$$

where $\tilde{\boldsymbol{X}}_k := [\boldsymbol{x}_1 - \boldsymbol{\mu}_k, \ldots, \boldsymbol{x}_N - \boldsymbol{\mu}_k]^\top$ is the data matrix $\boldsymbol{X}$ centered at $\boldsymbol{\mu}_k$, and $\boldsymbol{r}_k$ is the probability vector defined in (11.28).

Example 11.4 (Variance Updates)



(a) GMM density and individual components prior to updating the variances.

(b) GMM density and individual components after updating the variances.

Figure 11.6 Effect of updating the variances in a GMM. (a) GMM before updating the variances; (b) GMM after updating the variances while retaining the means and mixture weights.

In our example from Figure 11.3, the variances are updated as follows:

$$\sigma_1^2 : 1 \to 0.14 \tag{11.43}$$
$$\sigma_2^2 : 0.2 \to 0.44 \tag{11.44}$$
$$\sigma_3^2 : 3 \to 1.53 \tag{11.45}$$

Here, we see that the variances of the first and third component shrink significantly, the variance of the second component increases slightly.

Figure 11.6 illustrates this setting. Figure 11.6(a) is identical (but zoomed in) to Figure 11.5(b) and shows the GMM density and its individual components prior to updating the variances. Figure 11.6(b) shows the GMM density after updating the variances.

Similar to the update of the mean parameters, we can interpret (11.34) as a Monte-Carlo estimate of the weighted covariance of data points $\boldsymbol{x}_n$ associated with the kth mixture component, where the weights are the responsibilities r_{nk}. As with the updates of the mean parameters, this update depends on all $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j,\ j = 1, \ldots, K$, through the responsibilities r_{nk}, which prohibits a closed-form solution.

Theorem 11.3 (Update of the GMM Mixture Weights)**.** *The mixture weights of the GMM are updated as*

$$\pi_k = \frac{N_k}{N} \tag{11.46}$$

for $k = 1, \ldots, K$, where N is the number of data points and N_k is defined in (11.19).

Proof To find the partial derivative of the log-likelihood with respect to the weight parameters π_k, $k = 1, \ldots, K$, we take the constraint $\sum_k \pi_k = 1$ into account by using Lagrange multipliers (see Section 7.2). The Lagrangian is

$$\mathfrak{L} = \mathcal{L} + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \tag{11.47a}$$

$$= \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \tag{11.47b}$$

where $\mathcal{L}$ is the log-likelihood from (11.10) and the second term encodes for the equality constraint that all the mixture weights need to sum up to 1. We obtain the partial derivatives with respect to π_k and the Lagrange multiplier λ

$$\frac{\partial \mathfrak{L}}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} + \lambda \tag{11.48}$$

$$= \frac{1}{\pi_k} \underbrace{\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}}_{=N_k} + \lambda = \frac{N_k}{\pi_k} + \lambda \,, \tag{11.49}$$

$$\frac{\partial \mathfrak{L}}{\partial \lambda} = \sum_{k=1}^{K} \pi_k - 1 \,. \tag{11.50}$$

Setting both partial derivatives to $\mathbf{0}$ (necessary condition for optimum) yields the system of equations

$$\pi_k = -\frac{N_k}{\lambda} \,, \tag{11.51}$$

$$1 = \sum_{k=1}^{K} \pi_k \,. \tag{11.52}$$

Using (11.52) in (11.51) and solving for π_k, we obtain

$$\sum_{k=1}^{K} \pi_k = 1 \iff -\sum_{k=1}^{K} \frac{N_k}{\lambda} = 1 \iff -\frac{N}{\lambda} = 1 \iff \lambda = -N \,. \tag{11.53}$$

This allows us to substitute $-N$ for λ in (11.51) to obtain

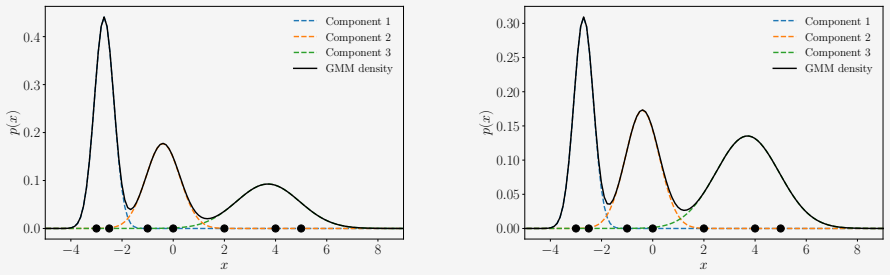
$$\pi_k = \frac{N_k}{N}\,, \tag{11.54}$$

which gives us the update for the weight parameters π_k and proves Theorem 11.3. □

We can identify the mixture weight in (11.46) as the ratio of the total responsibility of the kth cluster and the number of data points. Since $N = \sum_k N_k$ the number of data points can also be interpreted as the total responsibility of all mixture components together, such that π_k is the relative importance of the kth mixture component for the dataset.

Remark. Since $N_k = \sum_{i=1}^{N} r_{nk}$, the update equation (11.46) for the mixture weights π_k also depends on all $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, j = 1, \ldots, K$ via the responsibilities r_{nk}. ♢

Example 11.5 (Weight Parameter Updates)



(a) GMM density and individual components prior to updating the mixture weights.

(b) GMM density and individual components after updating the mixture weights.

Figure 11.7 Effect of updating the mixture weights in a GMM. (a) GMM before updating the mixture weights; (b) GMM after updating the mixture weights while retaining the means and variances. Note the different scalings of the vertical axes.

In our running example from Figure 11.3, the mixture weights are updated as follows:

$$\pi_1 : \tfrac{1}{3} \to 0.29 \tag{11.55}$$
$$\pi_2 : \tfrac{1}{3} \to 0.29 \tag{11.56}$$
$$\pi_3 : \tfrac{1}{3} \to 0.42 \tag{11.57}$$

Here we see that the third component gets more weight/importance, while the other components become slightly less important. Figure 11.7 illustrates the effect of updating the mixture weights. Figure 11.7(a) is identical to Figure 11.6(b) and shows the GMM density and its individual components prior to updating the mixture weights. Figure 11.7(b) shows the GMM density after updating the mixture weights.

Overall, having updated the means, the variances and the weights once,

we obtain the GMM shown in Figure 11.7(b). Compared with the initialization shown in Figure 11.3, we can see that the parameter updates caused the GMM density to shift some of its mass toward the data points.

After updating the means, variances and weights once, the GMM fit in Figure 11.7(b) is already remarkably better than its initialization from Figure 11.3. This is also evidenced by the log-likelihood values, which increased from 28.3 (initialization) to 14.4 (after one complete update cycle).

11.3 EM Algorithm

Unfortunately, the updates in (11.17), (11.34), and (11.46) do not constitute a closed-form solution for the updates of the parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$ of the mixture model because the responsibilities r_{nk} depend on those parameters in a complex way. However, the results suggest a simple *iterative scheme* for finding a solution to the parameters estimation problem via maximum likelihood. The Expectation Maximization algorithm (*EM algorithm*) was proposed by Dempster et al. (1977) and is a general iterative scheme for learning parameters (maximum likelihood or MAP) in mixture models and, more generally, latent-variable models.

EM algorithm

In our example of the Gaussian mixture model, we choose initial values for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$ and alternate until convergence between

- *E-step:* Evaluate the responsibilities r_{nk} (posterior probability of data point i belonging to mixture component k).
- *M-step:* Use the updated responsibilities to re-estimate the parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$.

Every step in the EM algorithm increases the log-likelihood function (Neal and Hinton, 1999). For convergence, we can check the log-likelihood or the parameters directly. A concrete instantiation of the EM algorithm for estimating the parameters of a GMM is as follows:

1. Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$
2. *E-step:* Evaluate responsibilities r_{nk} for every data point $\boldsymbol{x}_n$ using current parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$:
$$r_{nk} = \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\right)}{\sum_j \pi_j \mathcal{N}\left(\boldsymbol{x}_n \,|\, \boldsymbol{\mu}_j,\, \boldsymbol{\Sigma}_j\right)} \,. \tag{11.58}$$
3. *M-step:* Re-estimate parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ using the current responsibilities r_{nk} (from E-step):
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n \,, \tag{11.59}$$

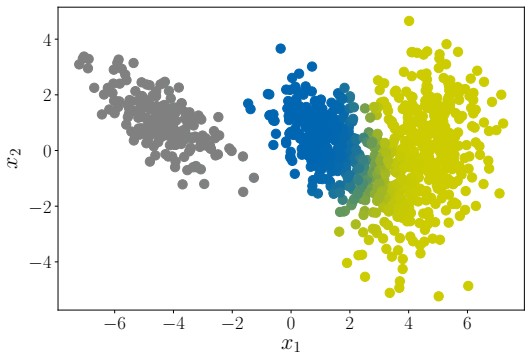
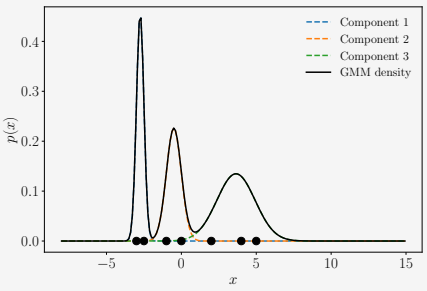


Figure 11.9 Data set colored according to the responsibilities of the mixture components when EM converges. While a single mixture component is clearly responsible for the data on the left, the overlap of the two data clusters on the right could have been generated by two mixture components.

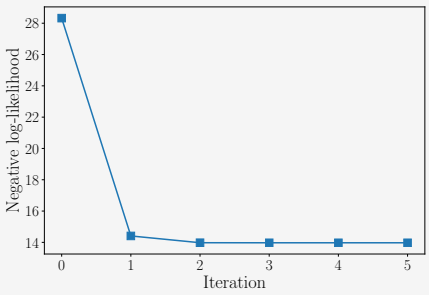
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top , \tag{11.60}$$

$$\pi_k = \frac{N_k}{N} . \tag{11.61}$$

Example 11.6 (GMM Fit)



(a) Final GMM fit. After five iterations, the EM algorithm converges and returns this mixture density.



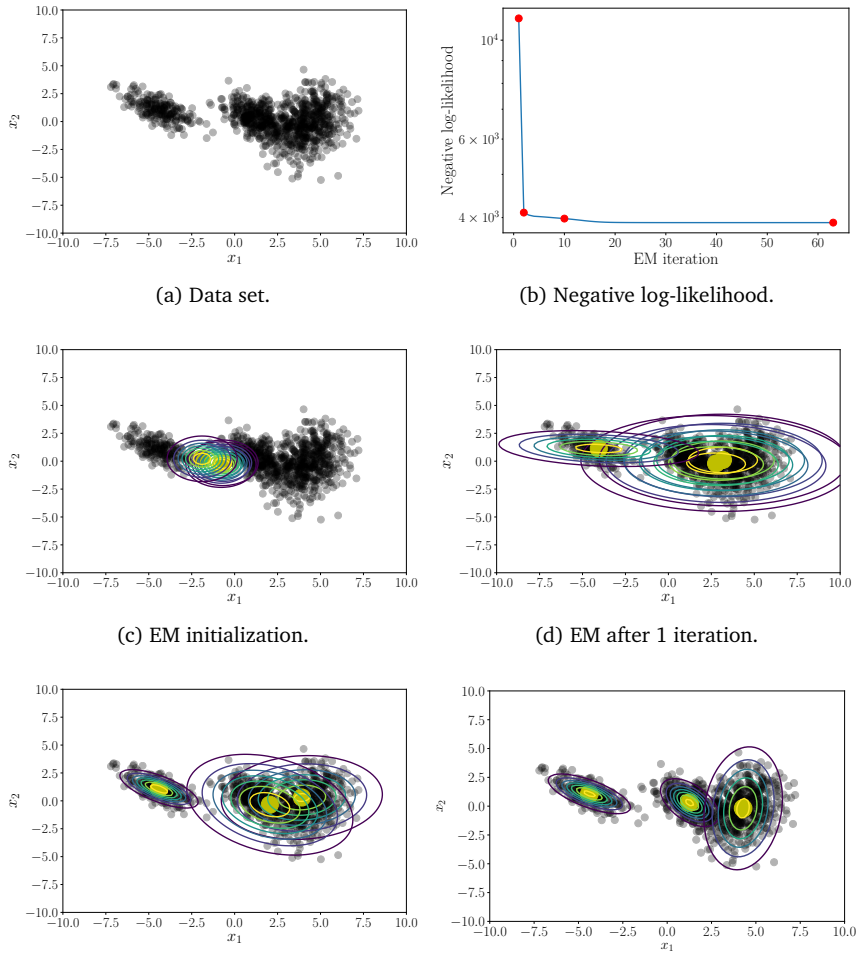
(b) Negative log-likelihood as a function of the EM iterations.

Figure 11.8 EM algorithm applied to the GMM from Figure 11.2.

When we run EM on our example from Figure 11.3, we obtain the final result shown in Figure 11.8(a) after five iterations, and Figure 11.8(b) shows how the negative log-likelihood evolves as a function of the EM iterations. The final GMM is given as

$$\begin{aligned} p(x) = 0.29\mathcal{N}\big(x \,|\, -2.75,\, 0.06\big) + 0.28\mathcal{N}\big(x \,|\, -0.50,\, 0.25\big) \\ + 0.43\mathcal{N}\big(x \,|\, 3.64,\, 1.63\big) . \end{aligned} \tag{11.62}$$

We applied the EM algorithm to the two-dimensional dataset shown in Figure 11.1 with $K = 3$ mixture components. Figure 11.9 visualizes the final responsibilities of the mixture components for the data points. It becomes clear that there are data points that cannot be uniquely assigned



(a) Data set.

(b) Negative log-likelihood.

(c) EM initialization.

(d) EM after 1 iteration.

(e) EM after 10 iterations

(f) EM after 63 iterations

Figure 11.10 Illustration of the EM algorithm for fitting a Gaussian mixture model with three components to a two-dimensional data set. (a) Data set; (b) Negative log likelihood (lower is better) as a function of the EM iterations. The red dots indicate the iterations for which the corresponding GMM fits are shown in (c)–(f). The yellow discs indicate the mean of the Gaussian distribution.

to a single (either blue or yellow) component, such that the responsibilities of these two clusters for those points are around 0.5. Figure 11.10 illustrates a few steps of the EM algorithm.

11.4 Latent Variable Perspective

We can look at the GMM from the perspective of a discrete latent variable model, i.e., where the latent variable $\boldsymbol{z}$ can attain only a finite set of values. This is in contrast to PCA where the latent variables were continuous valued numbers in $\mathbb{R}^M$. Let us assume that we have a mixture model with K components and that a data point $\boldsymbol{x}$ is generated by exactly one component. We can then use a discrete indicator variable $z_k \in \{0, 1\}$ that indicates whether the kth mixture component generated that data point

so that

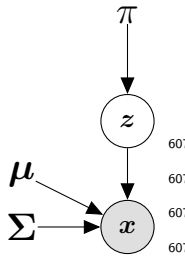
$$p(\boldsymbol{x} \mid z_k = 1) = \mathcal{N}\big(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)\,. \tag{11.63}$$

We define $\boldsymbol{z} := [z_1, \ldots, z_K]^\top \in \mathbb{R}^K$ as a vector consisting of $K-1$ many 0s and exactly one 1. Because of this, it also holds that $\sum_{k=1}^K z_k = 1$. Therefore, $\boldsymbol{z}$ is a *one-hot encoding* (also: 1-*of*-K *representation*). This allows us to write the conditional distribution as

$$p(\boldsymbol{x} \mid \boldsymbol{z}) = \prod_{k=1}^K \mathcal{N}\big(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)^{z_k}\,, \tag{11.64}$$

where $z_k \in \{0, 1\}$. Thus far, we assumed that the indicator variables z_k are known. However, in practice, this is not the case, and we place a prior distribution on $\boldsymbol{z}$.

Figure 11.11 Graphical model for a GMM with a single data point.



one-hot encoding

1-of-K representation

In the following, we will discuss the prior $p(\boldsymbol{z})$, the marginal $p(\boldsymbol{x}, \boldsymbol{z})$ and the posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$ for the case of observing a single data point $\boldsymbol{x}$ (the corresponding graphical model is shown in Figure 11.11) before extending the concepts to the general case where the dataset consists of N data points.

11.4.1 Prior

Given that we do not know which mixture component generated the data point, we treat the indicators $\boldsymbol{z}$ as a latent variable and place a prior $p(\boldsymbol{z})$ on it. Then the prior $p(z_k = 1) = \pi_k$ describes the probability that the kth mixture component generated data point $\boldsymbol{x}$. To ensure that our probability distribution is normalized, we require that $\sum_{k=1}^K \pi_k = 1$. We summarize the prior $p(\boldsymbol{z}) = \boldsymbol{\pi}$ in the probability vector $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]^\top$. Because of the one-hot encoding of $\boldsymbol{z}$, we can write the prior in a less obvious form

$$p(\boldsymbol{z}) = \prod_{k=1}^K \pi_k^{z_k}\,, \quad z_k \in \{0, 1\}\,, \tag{11.65}$$

but this form will become handy later on.

Remark (Sampling from a GMM). The construction of this latent variable model lends itself to a very simple sampling procedure to generate data:

1. Sample $\boldsymbol{z}_i \sim p(\boldsymbol{z} \mid \boldsymbol{\pi})$
2. Sample $\boldsymbol{x}_i \sim p(\boldsymbol{x} \mid \boldsymbol{z}_i)$

In the first step, we would select a mixture component at random according to $\boldsymbol{\pi}$; in the second step we would draw a sample from a single mixture component. This kind of sampling, where samples of random variables depend on samples from the variable's parents in the graphical model, is called *ancestral sampling*. This means, we can generate data from the mixture model by generating a latent variable value $k \in \{1, \ldots, K\}$ to identify a singe mixture component, and then generate a data point $\boldsymbol{x}_i$

ancestral sampling

by sampling from this mixture component. We can discard the samples of the latent variable so that we are left with the $\boldsymbol{x}_i$, which are valid samples from our mixture model. ♢

11.4.2 Marginal

If we marginalize out the latent variables $\boldsymbol{z}$ (by summing over all possible one-hot encodings), we obtain the marginal distribution

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\pi}) = \sum_{\boldsymbol{z}} \prod_{k=1}^{K} \left(\pi_k \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right)^{z_k} \tag{11.66a}$$

$$= \pi_1 \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) + \cdots + \pi_K \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K\right) \tag{11.66b}$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right), \tag{11.66c}$$

which is identical to the GMM we introduced in (11.3). Therefore, the latent variable model with latent indicators z_k is an equivalent way of thinking about a Gaussian mixture model.

The marginal distribution $p(\boldsymbol{x})$ is a Gaussian mixture model.

11.4.3 Posterior

Let us have a brief look at the posterior distribution on the latent variable $\boldsymbol{z}$. According to Bayes' theorem, the posterior is

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})}, \tag{11.67}$$

where $p(\boldsymbol{x})$ is given in (11.66c). With (11.64) and (11.65) we get the joint distribution as

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{z_k} \prod_{k=1}^{K} \pi_k^{z_k} \tag{11.68a}$$

$$= \prod_{k=1}^{K} \left(\pi_k \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right)^{z_k}. \tag{11.68b}$$

Here, we identify $p(\boldsymbol{x} \mid \boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)^{z_k}$ as the likelihood. This yields the posterior distribution for the kth indicator variable z_k

$$p(z_k = 1 \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid z_k = 1) p(z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1) p(\boldsymbol{x} \mid z_j = 1)} = \frac{\pi_k \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}, \tag{11.69}$$

which we identify as the responsibility of the kth mixture component for data point $\boldsymbol{x}$. Note that we omitted the explicit conditioning on the GMM parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ where $k = 1, \ldots, K$.

11.4.4 Extension to a Full Dataset

Thus far, we only discussed the case where the dataset consists only of a single data point $\boldsymbol{x}$. However, the concepts can be directly extended to the case of N data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, which we collect in the data matrix $\boldsymbol{X}$.

Every data point $\boldsymbol{x}_n$ possesses its own latent variable

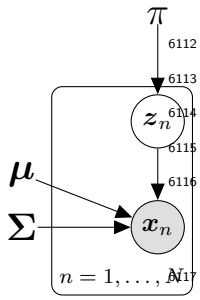
$$\boldsymbol{z}_n = [z_{n1}, \ldots, z_{nK}]^\top \in \mathbb{R}^K \,. \tag{11.70}$$

Previously (when we only considered a single data point $\boldsymbol{x}$) we omitted the index n, but now this becomes important. We collect all of these latent variables in the matrix $\boldsymbol{Z}$. We share the same prior distribution $\boldsymbol{\pi}$ across all data points. The corresponding graphical model is shown in Figure 11.12, where we use the plate notation.

The likelihood $p(\boldsymbol{X} \mid \boldsymbol{Z})$ factorizes over the data points, such that the joint distribution (11.68b) is given as

$$p(\boldsymbol{X}, \boldsymbol{Z}) = p(\boldsymbol{X} \mid \boldsymbol{Z})p(\boldsymbol{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \left(\pi_k \mathcal{N}\left(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k\right)\right)^{z_{nk}} \,. \tag{11.71}$$

Figure 11.12 Graphical model for a GMM with N data points.



Generally, the posterior distribution $p(z_k = 1 \mid \boldsymbol{x}_n)$ is the probability that the kth mixture component generated data point $\boldsymbol{x}_n$ and corresponds to the responsibility r_{nk} we introduced in (11.18). Now, the responsibilities also have not only an intuitive but also a mathematically justified interpretation as posterior probabilities.

11.4.5 EM Algorithm Revisited

The EM algorithm that we introduced as an iterative scheme for maximum likelihood estimation can be derived in a principled way from the latent variable perspective. Given a current setting $\boldsymbol{\theta}^{(t)}$ of model parameters, the E-step calculates the expected log-likelihood

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta}^{(t)}}[\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})] \tag{11.72a}$$

$$= \int \log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta}) p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta}^{(t)}) \mathrm{d}\boldsymbol{Z} \,, \tag{11.72b}$$

where the expectation of the log-joint distribution of latent variables $\boldsymbol{Z}$ and observations $\boldsymbol{X}$ is taken with respect to the posterior $p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta}^{(t)})$ of the latent variables. The M-step selects an updated set of model parameters $\boldsymbol{\theta}^{(t+1)}$ by maximizing (11.72b).

Although an EM iteration does increase the log-likelihood, there are no guarantees that EM converges to the maximum likelihood solution. It is possible that the EM algorithm converges to a local maximum of the log-likelihood. Different initializations of the parameters $\boldsymbol{\theta}$ could be used in multiple EM runs to reduce the risk of ending up in a bad local optimum. We do not go into further details here, but refer to the excellent expositions by Rogers and Girolami (2016) and Bishop (2006).

11.5 Further Reading

The GMM can be considered a generative model in the sense that it is straightforward to generate new data using ancestral sampling (Bishop, 2006). For given GMM parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, $k = 1, \ldots, K$, we sample an index k from the probability vector $[\pi_1, \ldots, \pi_K]^\top$ and then sample a data point $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. If we repeat this N times, we obtain a dataset that has been generated by a GMM. Figure 11.1 was generated using this procedure.

Throughout this chapter, we assumed that the number of components K is known. In practice, this is often not the case. However, we could use cross-validation, as discussed in Section 8.5, to find good models.

Gaussian mixture models are closely related to the K-means clustering algorithm. K-means also uses the EM algorithm to assign data points to clusters. If we treat the means in the GMM as cluster centers and ignore the covariances, we arrive at K-means. As also nicely described by MacKay (2003), K-means makes a "hard" assignments of data points to cluster centers $\boldsymbol{\mu}_k$, whereas a GMM makes a "soft" assignment via the responsibilities.

We only touched upon the latent variable perspective of GMMs and the EM algorithm. Note that EM can be used for parameter learning in general latent variable models, e.g., nonlinear state-space models (Ghahramani and Roweis, 1999; Roweis and Ghahramani, 1999) and for reinforcement learning as discussed by Barber (2012). Therefore, the latent variable perspective of a GMM is useful to derive the corresponding EM algorithm in a principled way (Bishop, 2006; Barber, 2012; Murphy, 2012).

We only discussed maximum likelihood estimation (via the EM algorithm) for finding GMM parameters. The standard criticisms of maximum likelihood also apply here:

- As in linear regression, maximum likelihood can suffer from severe overfitting. In the GMM case, this happens when the mean of a mixture component is identical to a data point and the covariance tends to $\mathbf{0}$. Then, the likelihood approaches infinity. Bishop (2006) and Barber (2012) discuss this issue in detail.
- We only obtain a point estimate of the parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for $k = 1, \ldots, K$, which does not give any indication of uncertainty in the parameter values. A Bayesian approach would place a prior on the parameters, which can be used to obtain a posterior distribution on the parameters. This posterior allows us to compute the model evidence (marginal likelihood), which can be used for model comparison, which gives us a principled way to determine the number of mixture components. Unfortunately, closed-form inference is not possible in this setting because there is no conjugate prior for this model. However, approximations, such as variational inference, can be used to obtain an approximate posterior (Bishop, 2006).

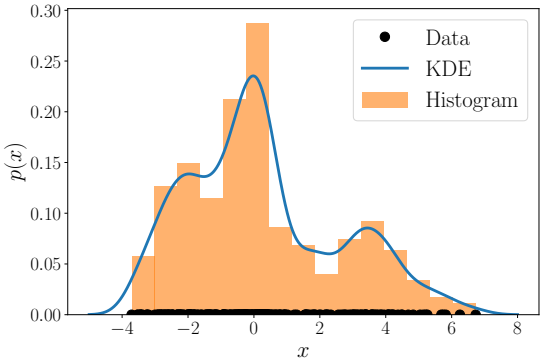


Figure 11.13 Histogram (orange bars) and kernel density estimation (blue line). The kernel density estimator (with a Gaussian kernel) produces a smooth estimate of the underlying density, whereas the histogram is simply an unsmoothed count measure of how many data points (black) fall into a single bin.

In this chapter, we discussed mixture models for density estimation. There is a plethora of density estimation techniques available. In practice we often use histograms and kernel density estimation.

Histograms

Histograms provide a non-parametric way to represent continuous densities and have been proposed by Pearson (1895). A histogram is constructed by "binning" the data space and count how many data points fall into each bin. Then a bar is drawn at the center of each bin, and the height of the bar is proportional to the number of data points within that bin. The bin size is a critical hyper-parameter, and a bad choice can lead to overfitting and underfitting. Cross-validation, as discussed in Section 8.1.4, can be used to determine a good bin size.

Kernel density estimation

Kernel density estimation, independently proposed by Rosenblatt (1956) and Parzen (1962), is a nonparametric way for density estimation. Given N i.i.d. samples, the kernel density estimator represents the underlying distribution as

$$p(\boldsymbol{x}) = \frac{1}{Nh}\sum_{n=1}^{N} k\left(\frac{\boldsymbol{x}-\boldsymbol{x}_n}{h}\right), \tag{11.73}$$

where k is a kernel function, i.e., a non-negative function that integrates to 1 and $h > 0$ is a smoothing/bandwidth parameter, which plays a similar role as the bin size in histograms. Note that we place a kernel on every single data point $\boldsymbol{x}_n$ in the dataset. Commonly used kernel functions are the uniform distribution and the Gaussian distribution. Kernel density estimates are closely related to histograms, but by choosing a suitable kernel, we can guarantee smoothness of the density estimate. Figure 11.13 illustrates the difference between a histogram and a kernel density estimator (with a Gaussian-shaped kernel) for a given data set of 250 data points.

12

Classification with Support Vector Machines

In many situations we want our machine learning algorithm to predict one of a number of outcomes. For example an email client that sorts mail into personal mail and junk mail, which has two outcomes. Another example is a telescope that identifies whether an object in the night sky is a galaxy, star or planet. There are usually a small number of outcomes, and more importantly there is usually no additional structure on these outcomes. In this chapter, we consider predictors that output binary values, that is, there are only two possible outcomes. This is in contrast to Chapter 9 where we considered a prediction problem with continuous-valued outputs. This machine learning task is called *binary classification*. For binary classification the set of possible values that the label/output can attain is binary, and for this chapter we denote them as $\{+1, -1\}$. In other words, we consider predictors of the form

An example of structure is if the outcomes were ordered, like in the case of small, medium and large t-shirts.

binary classification

$$f : \mathbb{R}^D \to \{+1, -1\}\,. \tag{12.1}$$

Recall from Chapter 8 that we represent each example $\boldsymbol{x}_n$ as a feature vector of D real numbers. The labels are often referred to as the positive and negative *classes*, respectively. One should be careful not to infer intuitive attributes of positiveness of the $+1$ class. For example, in a cancer detection task, a patient with cancer is often labelled $+1$. In principle, any two distinct values can be used, e.g., $\{\text{True}, \text{False}\}$, $\{0, 1\}$ or $\{\text{red}, \text{blue}\}$. The problem of binary classification is well studied, and we defer a survey of other approaches to Section 12.4.

Input example $\boldsymbol{x}_n$ may also be referred to as inputs, data points, features or instances.

classes

For probabilisitic models, it is mathematically convenient to use $\{0, 1\}$ as a binary representation. See remark after Example 6.15.

We present an approach known as the Support Vector Machine (SVM), which solves the binary classification task. Similar to regression, we have a supervised learning task, where we have a set of examples $\boldsymbol{x}_n \in \mathbb{R}^D$ along with their corresponding labels $y_n \in \{+1, -1\}$. Given the training data consisting of example-label pairs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, we would like to estimate parameters of the model that will give the best classification error. Similar to Chapter 9 we consider a linear model, and hide away the nonlinearity in a transformation ϕ of the examples (9.12). We will revisit ϕ later in this chapter in Section 12.3.3.

The SVM provides state of the art results in many applications, with sound theoretical guarantees (Steinwart and Christmann, 2008). In this book, the first reason we choose to discuss the SVM is to illustrate a

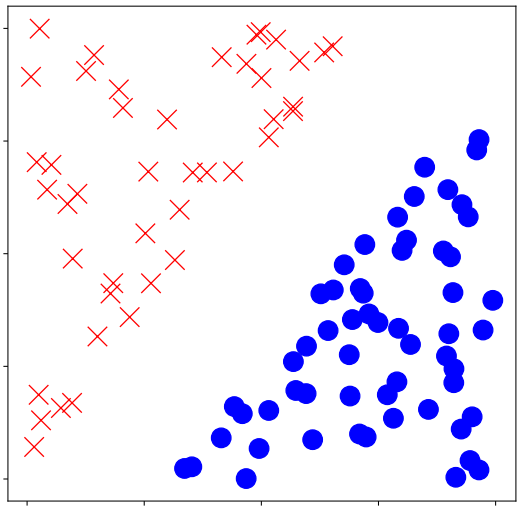


Figure 12.1 Example 2D data, illustrating the intuition of data where we can find a linear classifier that separates red crosses from blue dots.

geometric way to think about supervised machine learning. Whereas in Chapter 9 we considered the machine learning problem in terms of probabilistic models and attacked it using maximum likelihood estimation and Bayesian inference, here we will consider an alternative approach where we reason geometrically about the machine learning task. It relies heavily on concepts, such as inner products and projections, which we discussed in Chapter 3. In contrast to Chapter 9, the optimization problem for SVM does not admit an analytic solution. Hence, we resort to the optimization tools introduced in Chapter 7. This is the second reason for introducing the SVM – as an illustration of what to do when we cannot analytically derive a solution.

The SVM view of machine learning is also subtly different from the maximum likelihood view of Chapter 9. The maximum likelihood view proposes a model based on a probabilistic view of the data distribution, from which an optimization problem is derived. In contrast, the SVM view starts by designing a particular function that is to be optimized during training, based on geometric intuitions. In other words, we start by designing an objective function that is to be minimized on training data, following the principles of empirical risk minimization 8.1. This can also be understood as designing a particular loss function.

Let us derive the optimization problem corresponding to training an SVM on example-label pairs. Intuitively, we imagine binary classification data which can be separated by a hyperplane as illustrated in Figure 12.1, where the example (a vector of dimension 2) is used to indicate the location and the label is represented as different symbols (and colours). Hyperplane is a word that is commonly used in machine learning, and we saw them in Section 2.8 introduced as an affine subspace, which is the phrase used in linear algebra. The examples consists of two classes that

have features (the components of the vector representing the example) arranged in such a way as to allow us to separate/classify them by drawing a straight line.

In the following, we start by formalizing this idea of finding a linear separator. We introduce the idea of the margin and then extend linear separators to allow for examples to fall on the wrong side. We present two equivalent ways of formalizing the SVM: the geometric view (Section 12.2.4) and the loss function view (Section 12.2.5). We derive the dual version of the SVM using Lagrange multipliers (Section 7.2). The dual SVM allows us to observe a third way of formalizing the SVM: in terms of the convex hulls of the examples of each class (Section 12.3.2). We conclude by briefly describing kernels and how to numerically solve the nonlinear kernel-SVM optimization problem.

12.1 Separating Hyperplanes

Given two examples represented as vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, one way to compute the similarity between them is using a inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. Recall from Section 3.2 that inner products measure the angle between two vectors. The value of the inner product also depends on the length (norm) of each vector. Furthermore, inner products allow us to rigorously define geometric concepts such as orthogonality and projections.

The main idea behind many classification algorithms is to represent data in $\mathbb{R}^D$ and then partition this space. In the case of binary classification, the space would be split into two parts corresponding to the positive and negative classes, respectively. We consider a particularly convenient partition, which is to split the space into two halves using a hyperplane. Let example $\boldsymbol{x} \in \mathbb{R}^D$ be an element of the data space. Consider a function $f : \mathbb{R}^D \to \mathbb{R}$ parametrized by $\boldsymbol{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}$ as follows

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \,. \tag{12.2}$$

Recall from Section 2.8 that hyperplanes are affine subspaces. Therefore we define the hyperplane that separates the two classes in our binary classification problem as

$$\left\{ \boldsymbol{x} \in \mathbb{R}^D : f(\boldsymbol{x}) = 0 \right\}. \tag{12.3}$$

An illustration of the hyperplane is shown in Figure 12.2 where the vector $\boldsymbol{w}$ is a vector normal to the hyperplane and b the intercept. We can derive that $\boldsymbol{w}$ is a normal vector to the hyperplane in (12.3) by choosing any two examples $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ on the hyperplane and showing that the vector between them is orthogonal to $\boldsymbol{w}$. In the form of an equation,

$$\begin{aligned} f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b) &= \langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b - (\langle \boldsymbol{w}, \boldsymbol{x}_b \rangle + b) && (12.4) \\ &= \langle \boldsymbol{w}, \boldsymbol{x}_a - \boldsymbol{x}_b \rangle \,, && (12.5) \end{aligned}$$

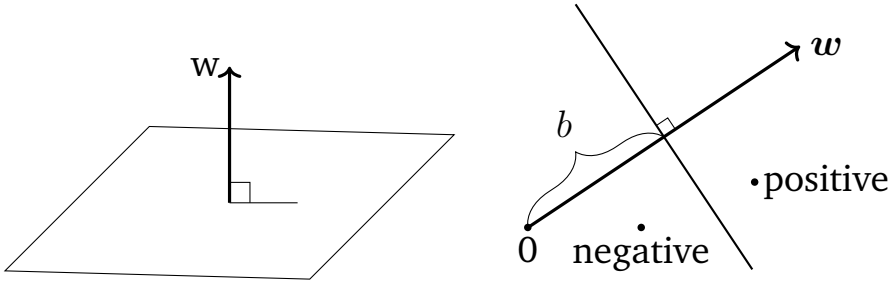


Figure 12.2 Equation of a separating hyperplane (12.3). (left) The standard way of representing the equation in 3D. (right) For ease of drawing, we look at the hyperplane edge on.

where the second line is obtained by the linearity of the inner product (Section 3.2). Since we have chosen $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ to be on the hyperplane, this implies that $f(\boldsymbol{x}_a) = 0$ and $f(\boldsymbol{x}_b) = 0$ and hence $\langle \boldsymbol{w}, \boldsymbol{x}_a - \boldsymbol{x}_b \rangle = 0$. Recall that two vectors are orthogonal when their inner product is zero, therefore we obtain that $\boldsymbol{w}$ is orthogonal to any vector on the hyperplane.

Remark. Recall from Chapter 1 that we can think of vectors in different ways. In this chapter, we think of the parameter vector $\boldsymbol{w}$ as an arrow indicating a direction. That is we consider $\boldsymbol{w}$ to be a geometric vector. In contrast, we think of the example vector $\boldsymbol{x}$ as a point (as indicated by its coordinates). That is we consider $\boldsymbol{x}$ to be the coordinates of a vector with respect to the standard basis. ♢

When presented with a test example, we classify the example as positive or negative by deciding on which side of the hyperplane it occurs. Note that (12.3) not only defines a hyperplane, it additionally defines a direction. In other words, it defines the positive and negative side of the hyperplane. Therefore, to classify a test example $\boldsymbol{x}_{\text{test}}$, we calculate the value of the function $f(\boldsymbol{x}_{\text{test}})$ and classify the example as $+1$ if $f(\boldsymbol{x}_{\text{test}}) \geqslant 0$ and -1 otherwise. Thinking geometrically, the positive examples lie "above" the hyperplane and the negative examples "below" the hyperplane.

When training the classifier, we want to ensure that the examples with positive labels are on the positive side of the hyperplane, i.e.,

$$\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b \geqslant 0 \quad \text{when} \quad y_n = +1 \tag{12.6}$$

and the examples with negative labels are on the negative side,

$$\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b < 0 \quad \text{when} \quad y_n = -1\,. \tag{12.7}$$

Refer to Figure 12.2 for a geometric intuition of positive and negative examples. These two conditions are often presented in a single equation,

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 0\,. \tag{12.8}$$

The equation (12.8) above is equivalent to (12.6) and (12.7) when we multiply both sides of (12.6) and (12.7) with $y_n = 1$ and $y_n = -1$, respectively.

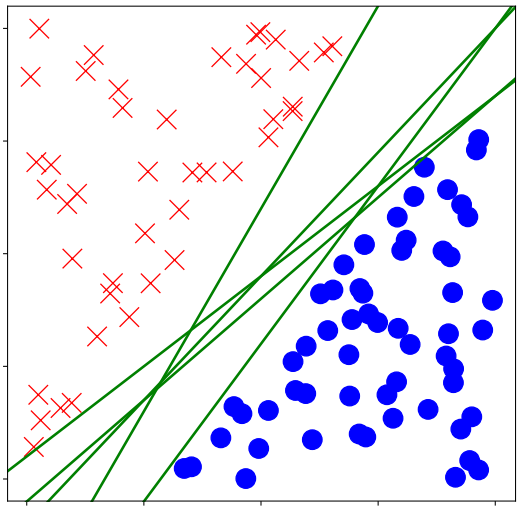


Figure 12.3 Possible separating hyperplanes. There are many linear classifiers (green lines) that separate red crosses from blue dots.

12.2 Primal Support Vector Machine

Based on the concept of distances from points to a hyperplane, we now are in a position to discuss the support vector machine. For a dataset $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ that is linearly separable, we have many candidate hyperplanes (refer to Figure 12.3) that solve our classification problem without any (training) errors. In other words, for a given training set we have many candidate classifiers. One idea is to choose the separating hyperplane that maximizes the margin between the positive and negative examples. In the following, we use the concept of a hyperplane, see also Section 2.8, and derive the distance between an example and a hyperplane. Recall that the closest point on the hyperplane to a given point (example $\boldsymbol{x}_n$) is obtained by the orthogonal projection (Section 3.7). We will see in the next section how to use the orthogonal projection to derive the margin.

A classifier with large margin turns out to generalize well (Steinwart and Christmann, 2008).

12.2.1 Concept Of The Margin

The concept of the *margin* is intuitively simple: It is the distance of the separating hyperplane to the closest examples in the dataset, assuming that the dataset is linearly separable. However, when trying to formalize this distance, there is a technical wrinkle that is confusing. The technical wrinkle is that we need to define a scale at which to measure the distance. A potential scale is to consider the scale of the data, i.e., the raw values of $\boldsymbol{x}_n$. There are problems with this, as we could change the units of measurement of $\boldsymbol{x}_n$ and change the values in $\boldsymbol{x}_n$, and, hence, change the distance to the hyperplane. As we will see shortly, we define the scale based on the equation of the hyperplane (12.3) itself.

margin

There could be two or more closest examples to a hyperplane.

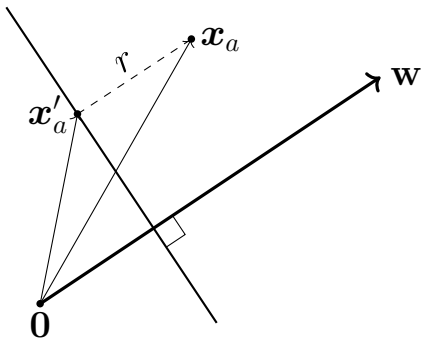


Figure 12.4 Vector addition to express distance to hyperplane: $\boldsymbol{x}_a = \boldsymbol{x}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$.

Consider a hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$, and an example $\boldsymbol{x}_a$ as illustrated in Figure 12.4. Without loss of generality, we can consider the example $\boldsymbol{x}_a$ to be on the positive side of the hyperplane, i.e., $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b > 0$. We would like to derive the distance $r > 0$ of $\boldsymbol{x}_a$ from the hyperplane. We do so by considering the orthogonal projection (Section 3.7) of $\boldsymbol{x}_a$ onto the hyperplane, which we denote by $\boldsymbol{x}'_a$. Since $\boldsymbol{w}$ is orthogonal to the hyperplane, we know that the distance r is just a scaling of this vector $\boldsymbol{w}$. However, we need to use a vector of unit length (its norm must be 1), and obtain this by dividing $\boldsymbol{w}$ by its norm, $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Using vector addition (Section 2.4) we obtain

$$\boldsymbol{x}_a = \boldsymbol{x}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \,. \tag{12.9}$$

Another way of thinking about r is that it is the coordinate of $\boldsymbol{x}_a$ in the subspace spanned by $\boldsymbol{w}$. We have now expressed the distance of $\boldsymbol{x}_a$ from the hyperplane as r, and if we choose $\boldsymbol{x}_a$ to be the point closest to the hyperplane, this distance r is the margin.

Recall that we would like the positive examples to be further than r from the hyperplane, and the negative examples to be further than distance r (in the negative direction) from the hyperplane. Analogously to the combination of (12.6) and (12.7) into (12.8), we have

$$y_n(\langle \boldsymbol{w}, x_n \rangle + b) \geqslant r \,. \tag{12.10}$$

In other words we can combine the requirements that examples are further than r from the hyperplane (in the positive and negative direction) into one single inequality.

Since we are interested only in the direction, we add an assumption to our model that the parameter vector $\boldsymbol{w}$ is of unit length, that is, $\|\boldsymbol{w}\| = 1$ where we use the Euclidean norm $\|\boldsymbol{w}\| = \sqrt{\boldsymbol{w}^\top w}$ (Section 3.1). Collecting the three requirements into one constrained optimization problem, we

A reader familiar with other presentations of the margin would notice that our definition of $\|\boldsymbol{w}\| = 1$ is different from the presentation in for example Schölkopf and Smola (2002). We will show that the two approaches are equivalent in Section 12.2.3.

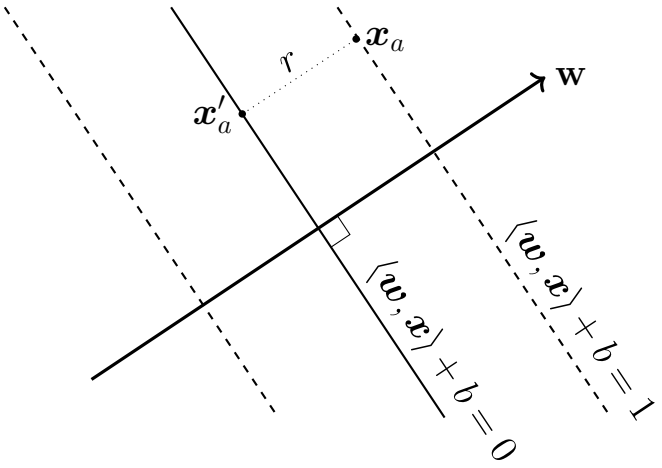


Figure 12.5 Derivation of the margin: $r = \frac{1}{\|\boldsymbol{w}\|}$.

obtain the following

$$\max_{\boldsymbol{w},b,r} \underbrace{r}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant r}_{\text{data fitting}}, \quad \underbrace{\|\boldsymbol{w}\| = 1}_{\text{normalization}} \quad , r > 0, \tag{12.11}$$

which says that we want to maximize the margin r, while ensuring that the data lies on the correct side of the hyperplane.

Remark. The idea of the margin turns out to be highly pervasive in machine learning. It was used by Vladimir Vapnik and Alexey Chervonenkis to show that when the margin is large, the "complexity" of the function class is low, and, hence, learning is possible (Vapnik, 2000). It turns out that the concept is useful for various different approaches for theoretically analyzing generalization error (Shalev-Shwartz and Ben-David, 2014; Steinwart and Christmann, 2008). ♢

12.2.2 Traditional Derivation Of The Margin

In the previous section, we derived (12.11) by making the observation that we are only interested in the direction of $\boldsymbol{w}$ and not its length, leading to the assumption that $\|\boldsymbol{w}\| = 1$. In this section, we derive the margin maximization problem by making a different assumption. Instead of choosing that the parameter vector is normalised, we choose a scale for the data. We choose this scale such that the value of the predictor $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ is 1 at the closest example. Let us also consider $\boldsymbol{x}_a$ to be the example in the dataset that is closest to the hyperplane.

Recall that we currently consider linearly separable data.

Figure 12.5 is the same as Figure 12.4, except that now we have rescaled the axes, such that we have the example $\boldsymbol{x}_a$ exactly on the margin, i.e., $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b = 1$. Since $\boldsymbol{x}'_a$ is the orthogonal projection of $\boldsymbol{x}_a$ onto the

hyperplane, it must by definition lie on the hyperplane, i.e.,

$$\langle \boldsymbol{w}, \boldsymbol{x}'_a \rangle + b = 0\,. \tag{12.12}$$

By substituting (12.9) into (12.12) we obtain

$$\left\langle \boldsymbol{w}, \boldsymbol{x}_a - r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right\rangle + b = 0\,. \tag{12.13}$$

Multiplying out the inner product, we get

$$\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b - r\frac{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|} = 0\,, \tag{12.14}$$

where we exploited the linearity of the inner product (see Section 3.2). Observe that the first term is unity by our assumption of scale, that is, $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b = 1$. From (3.18) in Section 3.1 we recall that $\langle \boldsymbol{w}, \boldsymbol{w} \rangle = \|\boldsymbol{w}\|^2$, and hence the second term reduces to $r\|\boldsymbol{w}\|$. Using these simplifications, we obtain

$$r = \frac{1}{\|\boldsymbol{w}\|}\,, \tag{12.15}$$

where we have derived the distance r in terms of the normal vector $\boldsymbol{w}$ of the hyperplane. At first glance this equation is counterintuitive as we seem to have derived the distance from the hyperplane in terms of the length of the vector $\boldsymbol{w}$, but we do not yet know this vector. One way to think about it is to consider the distance r to be a temporary variable that we only use for this derivation. In fact, for the rest of this section we will refer to the distance to the hyperplane by $\frac{1}{\|\boldsymbol{w}\|}$. In Section 12.2.3 we will see that the choice that the margin equals 1 is equivalent to our previous assumption of $\|\boldsymbol{w}\| = 1$ in Section 12.2.1.

We can also think of the distance as the projection error that incurs when projecting $\boldsymbol{x}_a$ onto the hyperplane.

Similar to the argument to obtain (12.10), we want the positive examples to be further than 1 from the hyperplane, and the negative examples to be further than distance 1 (in the negative direction) from the hyperplane

$$y_n(\langle \boldsymbol{w}, x_n \rangle + b) \geqslant 1\,. \tag{12.16}$$

Combining the margin maximization with the fact that examples needs to be on the correct side of the hyperplane (based on their labels) gives us

$$\max_{w,b} \frac{1}{\|\boldsymbol{w}\|} \tag{12.17}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 \quad \text{for all} \quad n = 1, \ldots, N. \tag{12.18}$$

Instead of maximizing the reciprocal of the norm as in (12.17), we often minimize the squared norm. We also often include a constant $\frac{1}{2}$ that does not affect the optimal $\boldsymbol{w}, b$ but yields a tidier form when we take the derivative. Then, our objective becomes

The squared norm results in a convex quadratic programming problem for the SVM (Section 12.3.4).

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{12.19}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 \quad \text{for all} \quad n = 1, \ldots, N\,. \tag{12.20}$$

Equation (12.19) is known as the *hard margin SVM*. The reason for the expression "hard" is because the above formulation does not allow for any violations of the margin condition. We will see in Section 12.2.4 that this "hard" condition can be relaxed to accommodate violations.

hard margin SVM

12.2.3 Why We Can Set The Margin To 1

In Section 12.2.1 we argue that we would like to maximize some value r, which represents the distance of the closest example to the hyperplane. In Section 12.2.2 we scaled the data such that the closest example is of distance 1 to the hyperplane. Here we relate the two derivations, and show that they are actually equivalent.

Theorem 12.1. *Maximizing the margin r where we consider normalized weights as in* (12.11)*,*

$$\max_{\boldsymbol{w},b,r} \underbrace{r}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant r}_{\text{data fitting}}, \quad \underbrace{\|\boldsymbol{w}\| = 1}_{\text{normalization}} \quad , r > 0 \tag{12.21}$$

is equivalent to scaling the data such that the margin is unity

$$\min_{\boldsymbol{w},b} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1}_{\text{data fitting}}\,. \tag{12.22}$$

Proof Consider (12.21), and note that because the square is a monotonic transformation for non-negative arguments, the maximum stays the same if we consider r^2 in the objective. Since $\|\boldsymbol{w}\| = 1$ we can reparameterize the equation with a new weight vector $\boldsymbol{w}'$ that is not normalized by explicitly using $\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}$,

$$\max_{\boldsymbol{w}',b,r} \quad r^2 \quad \text{subject to} \quad y_n\left(\left\langle \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}, \boldsymbol{x}_n \right\rangle + b\right) \geqslant r, \quad r > 0. \tag{12.23}$$

In (12.23) we have explicitly written that distances are non-negative. We can divide the first constraint by r,

Note that $r > 0$ because we assumed linear separability, and hence there is no issue to divide by r.

$$\max_{\boldsymbol{w}',b,r} \quad r^2 \quad \text{subject to} \quad y_n\left(\left\langle \underbrace{\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|\, r}}_{\boldsymbol{w}''}, \boldsymbol{x}_n \right\rangle + \underbrace{\frac{b}{r}}_{b''}\right) \geqslant 1, \quad r > 0 \tag{12.24}$$

renaming the parameters to $\boldsymbol{w}''$ and b''. Since $\boldsymbol{w}'' = \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\| r}$, rearranging for r gives

$$\|\boldsymbol{w}''\| = \left\| \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|\, r} \right\| = \left| \frac{1}{r} \right| \cdot \left\| \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|} \right\| = \frac{1}{r}. \tag{12.25}$$

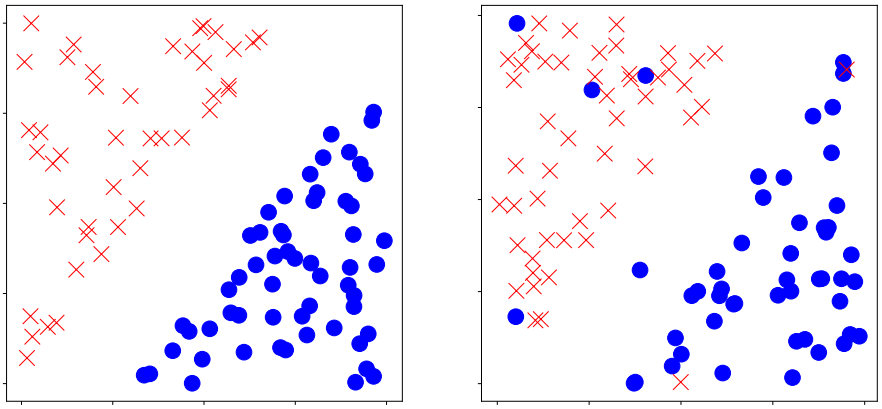


Figure 12.6 (left) linearly separable data, with a large margin. (right) non-separable data.

Substituting into (12.24), we obtain

$$\max_{\boldsymbol{w}'',b''} \quad \frac{1}{\|\boldsymbol{w}''\|^2} \quad \text{subject to} \quad y_n\left(\langle \boldsymbol{w}'', \boldsymbol{x}_n\rangle + b''\right) \geqslant 1. \tag{12.26}$$

The final step is to observe that maximizing $\frac{1}{\|\boldsymbol{w}''\|^2}$ yields the same solution as minimizing $\frac{1}{2}\|\boldsymbol{w}''\|^2$. □

12.2.4 Soft Margin SVM: Geometric View

We may wish to allow some examples to fall within the margin region, or even to be on the wrong side of the hyperplane (as illustrated in Figure 12.6). This also naturally provides us with an approach that works when we do not have linearly separable data.

soft margin SVM

The resulting model is called the *soft margin SVM*. In this section, we derive the resulting optimization problem using geometric arguments. In Section 12.2.5, we will derive the same optimization problem using the idea of a loss function. Using Lagrange multipliers (Section 7.2), we will derive the dual optimization problem of the SVM in Section 12.3. This dual optimization problem allows us to observe a third interpretation of the SVM, as a hyperplane that bisects the line between convex hulls corresponding to the positive and negative data examples (Section 12.3.2).

slack variable

The key geometric idea is to introduce a *slack variable* ξ_n corresponding to each example $(\boldsymbol{x}_n, y_n)$ that allows a particular example to be within the margin or even on the wrong side of the hyperplane (refer to Figure 12.7). We subtract the value of ξ_n from the margin, constraining ξ_n to be non-negative. To encourage correct classification of the samples, we add ξ_n to the objective

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \tag{12.27}$$

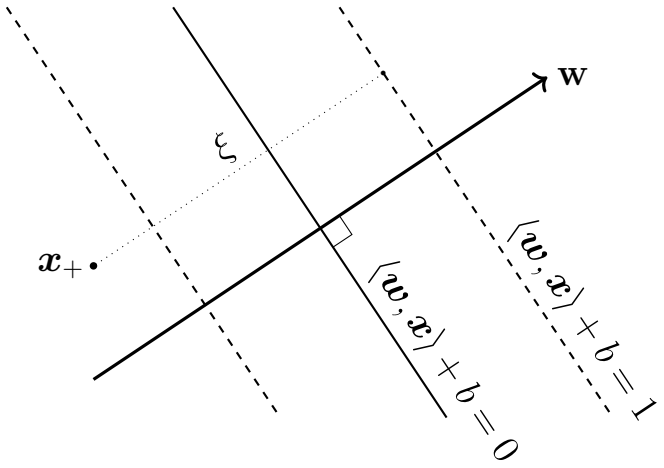


Figure 12.7 Soft Margin SVM allows examples to be within the margin or on the wrong side of the hyperplane. The slack variable ξ measures the distance of a positive example $\boldsymbol{x}_+$ to the positive margin hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 1$ when $\boldsymbol{x}_+$ is on the wrong side.

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 - \xi_n \quad \text{for all} \quad n = 1, \ldots, N \tag{12.28}$$

$$\xi_n \geqslant 0 \quad \text{for all} \quad n = 1, \ldots, N. \tag{12.29}$$

In contrast to the optimization problem (12.19) from the previous section (the hard margin SVM), this one is called the *soft margin SVM*. The parameter $C > 0$ trades off the size of the margin and the total amount of slack that we have. This parameter is called the *regularization parameter* since, as we will see in the following section, the margin term in the objective function (12.27) is a regularization term. The margin term $\|\boldsymbol{w}\|^2$ is called the *regularizer*, and in many books on numerical optimization, the regularization parameter multiplied with this term (Section 8.1.3). This is in contrast to our formulation in this section. Some care needs to be taken when interpreting the regularizer, as a large value of C implies low regularization, as we give the slack variables larger weight.

soft margin SVM

regularization parameter

regularizer

Remark. One detail to note is that in the formulation of the SVM (12.27) $\boldsymbol{w}$ is regularized but b is not regularized. We can see this by observing that the regularization term does not contain b. The unregularized term b complicates theoretical analysis (Steinwart and Christmann, 2008, Chapter 1) and decreases computational efficiency (Fan et al., 2008). ♢

There are alternative parametrizations of this regularization, which is why (12.27) is also often referred to as the C-SVM.

12.2.5 Soft Margin SVM: Loss Function View

Recall from Section 9.2.1 that when performing maximum likelihood estimation we usually consider the negative log likelihood. Furthermore since the likelihood term for linear regression with Gaussian noise is Gaussian, the negative log likelihood for each example is a squared error function (9.8). The squared error function is the term that is minimized when looking for the maximum likelihood solution. Let us consider the error function point of view, which is also known as the *loss function* point of view.

loss function

In contrast to Chapter 9 where we consider regression problems (the output of the predictor is a real number), in this chapter we consider binary classification problems (the output of the predictor is one of two labels $\{+1, -1\}$). Therefore the error function or the loss function for each single (example, label) pair needs to be appropriate for binary classification. For example, the squared loss that is used for regression (9.9b) is not suitable for binary classification.

Remark. The ideal loss function between binary labels is to count the number of mismatches between the prediction and the label. That is for a predictor f applied to an example $\boldsymbol{x}_n$, we compare the output $f(\boldsymbol{x}_n)$ with the label y_n. We define the loss to be zero if they match, and one if they do not match. This is denoted by $\mathbf{1}(f(\boldsymbol{x}_n) \neq y_n)$ and is called the zero-one loss. Unfortunately the zero-one loss results in a difficult optimization problem for finding the best parameters $\boldsymbol{w}, b$. ♢

What is the loss function corresponding to the SVM? Consider the error between the output of a predictor $f(\boldsymbol{x}_n)$ and the label y_n. The loss should capture how much we care about the error that is made on the training data. An equivalent way to derive (12.27) is to use the *hinge loss*

hinge loss

$$\ell(t) = \max\{0, 1 - t\} \quad \text{where} \quad t = yf(\boldsymbol{x}) = y(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)\,. \tag{12.30}$$

If $f(\boldsymbol{x})$ is on the correct side (based on y) of the hyperplane, and further than distance 1, this means that $t \geqslant 1$ and the hinge loss returns a value of zero. If $f(\boldsymbol{x})$ is on the correct side but close to the hyperplane, that is, $0 < t < 1$, then the example $\boldsymbol{x}$ is within the margin and the hinge loss returns a positive value. When the example is on the wrong side of the hyperplane ($t < 0$) the hinge loss returns an even larger value, which increases linearly. In other words, we pay a penalty once we are closer than the margin, even if the prediction is correct, and the penalty increases linearly. An alternative way to express the hinge loss is by considering it as two linear pieces

$$\ell(t) = \begin{cases} 0 & \textbf{if} \quad t \geqslant 1 \\ 1 - t & \textbf{if} \quad t < 1 \end{cases}, \tag{12.31}$$

as illustrated in Figure 12.8. The loss corresponding to the hard margin SVM 12.19 is defined as follows

$$\ell(t) = \begin{cases} 0 & \textbf{if} \quad t \geqslant 1 \\ \infty & \textbf{if} \quad t < 1 \end{cases}. \tag{12.32}$$

This loss can be interpreted as never allowing any examples inside the margin.

For a given training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ we would like to minimize the total loss, while regularizing the objective with ℓ_2 regularization (see Section 8.1.3). Using the hinge loss (12.30) gives us the uncon-

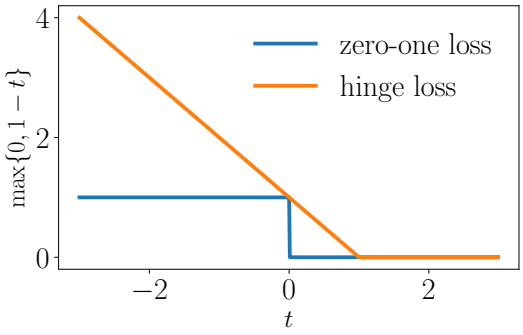


Figure 12.8 Hinge Loss is a convex envelope of zero-one loss.

strained optimization problem

$$\min_{\boldsymbol{w},b} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{regularizer}} + \underbrace{C\sum_{n=1}^{N}\max\{0, 1-y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b)\}}_{\text{error term}} . \tag{12.33}$$

The first term in (12.33) is called the regularization term or the *regularizer* (see Section 9.2.3), and the second term is called the *loss term* or the *error term*. Recall from Section 12.2.4 that the term $\frac{1}{2}\|\boldsymbol{w}\|^2$ is actually the term arising from the margin. In other words, margin maximization can be interpreted as a regularizer.

regularizer
loss term
error term

Margin maximization can be interpreted as a regularizer.

In principle, the unconstrained optimization problem in (12.33) can be directly solved with (sub-)gradient descent methods as described in Section 7.1. To see that (12.33) and (12.27) are equivalent, observe that the hinge loss (12.30) essentially consists of two linear parts, as expressed in (12.31). Therefore, we can equivalently replace minimization of the hinge loss with two constraints, i.e.,

$$\min_t \max\{0, 1-t\} \tag{12.34}$$

is equivalent to

$$\begin{aligned} \min_{\xi,t} \quad & \xi \\ \text{subject to} \quad & \xi \geqslant 0 \\ & \xi \geqslant 1-t\,. \end{aligned} \tag{12.35}$$

By substituting this into (12.33) and rearranging one of the constraints, we obtain exactly the soft margin SVM (12.27).

12.3 Dual Support Vector Machine

The description of the SVM in the previous sections, in terms of the variables $\boldsymbol{w}$ and b, is known as the primal SVM. Recall that we are considering

input vectors $\boldsymbol{x}$, which have dimension D, i.e., we are looking at input examples with D features. Since $\boldsymbol{w}$ is of the same dimension as $\boldsymbol{x}$, this means that the number of parameters (the dimension of $\boldsymbol{w}$) of the optimization problem grows linearly with the number of features.

In the following, we consider an equivalent optimization problem (the so-called dual view) which is independent of the number of features. We see a similar idea appear in Chapter 10 where we express the learning problem in a way that does not scale with the number of features. This is useful for problems where we have more features than the number of examples. Instead the number of parameters increases with the number of examples in the training set. The dual SVM also has the additional advantage that it easily allows kernels to be applied, as we shall see at the end of this chapter. The word "dual" appears often in mathematical literature, and in this particular case it refers to convex duality. The following subsections are essentially an application of convex duality as discussed in Section 7.2.

12.3.1 Convex Duality Via Lagrange Multipliers

Recall the primal soft margin SVM (12.27). We call the variables $\boldsymbol{w}$, b and ξ corresponding to the primal SVM the primal variables. We use $\alpha_n \geqslant 0$ as the Lagrange multiplier corresponding to the constraint (12.28) that the examples are classified correctly and $\gamma_n \geqslant 0$ as the Lagrange multiplier corresponding to the non-negativity constraint of the slack variable, see (12.29). The Lagrangian is then given by

In Chapter 7 we used λ as Lagrange multipliers. In this section we follow the notation commonly chosen in SVM literature, and use α and γ.

$$\mathfrak{L}(\boldsymbol{w}, b, \xi, \alpha, \gamma) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \underbrace{-\sum_{n=1}^{N}\alpha_n(y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) - 1 + \xi_n)}_{\text{constraint (12.28)}} \underbrace{-\sum_{n=1}^{N}\gamma_n\xi_n}_{\text{constraint (12.29)}} . \tag{12.36}$$

Differentiating the Lagrangian (12.36) with respect to the three primal variables $\boldsymbol{w}$, b and ξ respectively, we obtain

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n , \tag{12.37}$$

$$\frac{\partial \mathfrak{L}}{\partial b} = \sum_{n=1}^{N}\alpha_n y_n , \tag{12.38}$$

$$\frac{\partial \mathfrak{L}}{\partial \xi_n} = C - \alpha_n - \gamma_i . \tag{12.39}$$

We now find the maximum of the Lagrangian by setting each of these partial derivatives to zero. By setting (12.37) to zero we find

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n \,, \tag{12.40}$$

which is a particular instance of the *representer theorem* (Kimeldorf and Wahba, 1970). Equation (12.40) says that the optimal weight vector in the primal is a linear combination of the examples. Recall from Section 2.6.1 that this means that the solution of the optimization problem lies in the span of training data. Additionally the constraint obtained by setting 12.38 to zero implies that the optimal weight vector is an affine combination of the examples. The representer theorem turns out to hold for very general settings of regularized empirical risk minimization (Hofmann et al., 2008; Argyriou and Dinuzzo, 2014). The theorem has more general versions (Schölkopf et al., 2001), and necessary and sufficient conditions on its existance can be found in Yu et al. (2013).

representer theorem

The representer theorem is actually a collection of theorems saying that the solution of minimizing empirical risk lies in the subspace (Section 2.4.3) defined by the examples.

Remark. The representer theorem (12.40) also provides an explaination of the name Support Vector Machine. The examples $\boldsymbol{x}_n$ whose corresponding parameters $\alpha_n = 0$ do not contribute to the solution $\boldsymbol{w}$ at all. The other examples, where $\alpha_n > 0$, are called *support vectors* since they "support" the hyperplane. ♢

support vectors

By substituting the expression for $\boldsymbol{w}$ into the Lagrangian (12.36), we obtain the dual

$$\begin{aligned}\mathfrak{D}(\xi, \alpha, \gamma) = &\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{N} y_i \alpha_i \left\langle \sum_{j=1}^{N} y_j \alpha_j \boldsymbol{x}_j, \boldsymbol{x}_i \right\rangle \\ &+ C\sum_{i=1}^{N}\xi_i - b\sum_{i=1}^{N} y_i \alpha_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i \xi_i - \sum_{i=1}^{N}\gamma_i \xi_i \,.\end{aligned} \tag{12.41}$$

Note that there are no longer any terms involving the primal variable $\boldsymbol{w}$. By setting (12.38) to zero, we obtain $\sum_{n=1}^{N} y_n \alpha_n = 0$. Therefore, the term involving b also vanishes. Recall that inner products are symmetric and linear (see Section 3.2). Therefore, the first two terms in (12.41) are over the same objects. These terms (coloured blue) can be simplified, and we obtain the Lagrangian

$$\mathfrak{D}(\xi, \alpha, \gamma) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{N}(C - \alpha_i - \gamma_i)\xi_i \,. \tag{12.42}$$

The last term in this equation is a collection of all terms that contain slack variables ξ_i. By setting (12.39) to zero, we see that the last term in (12.41) is also zero. Furthermore, by using the same equation and recalling that the Lagrange multipliers γ_i are non-negative, we conclude that $\alpha_i \leqslant C$.

We now obtain the dual optimization problem of the SVM, which is expressed exclusively in terms of the Lagrange multipliers α_i. Recall from Lagrangian duality (Theorem 7.1) that we maximize the dual problem. This is equivalent to minimizing the negative dual problem, such that we end up with the *dual SVM*

dual SVM

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \boldsymbol{x}_i, \boldsymbol{x}_j \right\rangle - \sum_{i=1}^{N} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leqslant \alpha_i \leqslant C \quad \text{for all} \quad i = 1, \dots, N\,. \end{aligned} \tag{12.43}$$

The equality constraint in (12.43) is obtained from setting (12.38) to zero. The inequality constraint $\alpha_i \geqslant 0$ is the condition imposed on Lagrange multipliers of inequality constraints (Section 7.2). The inequality constraint $\alpha_i \leqslant C$ is discussed in the previous paragraph.

The set of inequality constraints in the SVM are called "box constraints" because they limit the vector $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^\top \in \mathbb{R}^N$ of Lagrange multipliers to be inside the box defined by 0 and C on each axis. These axis-aligned boxes are particularly efficient to implement in numerical solvers (Dostál, 2009, Chapter 5).

Once we obtain the dual parameters $\boldsymbol{\alpha}$ we can recover the primal parameters $\boldsymbol{w}$ by using the representer theorem (12.40). Let us call the optimal primal parameter $\boldsymbol{w}^*$. However there remains the question on how to obtain the parameter b^*. Consider an example $(\boldsymbol{x}_n)$ that lies exactly on the margin's boundary, that is, $\langle \boldsymbol{w}^*, \boldsymbol{x}_n \rangle + b = y_n$. Recall that y_n is either $+1$ or -1, and therefore the only unknown is b which can be computed by

It turns out examples that lie exactly on the margin are examples whose dual parameters lie strictly inside the box constraints, $0 < \alpha_i < C$. This is derived using the Karush Kuhn Tucker conditions, for example in Schölkopf and Smola (2002).

$$b^* = y_n - \langle \boldsymbol{w}^*, \boldsymbol{x}_n \rangle\,. \tag{12.44}$$

Remark. In principle there may be no examples that lie exactly on the margin. In this case we should compute $|y_n - \langle \boldsymbol{w}^*, \boldsymbol{x}_n \rangle|$ for all support vectors and take the median value of this absolute value difference to be the value of b^*. A derivation of this fact can be found in `http://fouryears.eu/2012/06/07/the-svm-bias-term-conspiracy/`. ♢

12.3.2 Soft Margin SVM: Convex Hull View

Another approach to obtain the SVM is to consider an alternative geometric argument. Consider the set of examples $\boldsymbol{x}_n$ with the same label. We would like to build a convex boundary around this set of examples that is the smallest possible. This is called the convex hull and is illustrated in Figure 12.9.

Let us first build some intuition about a convex combination of points.

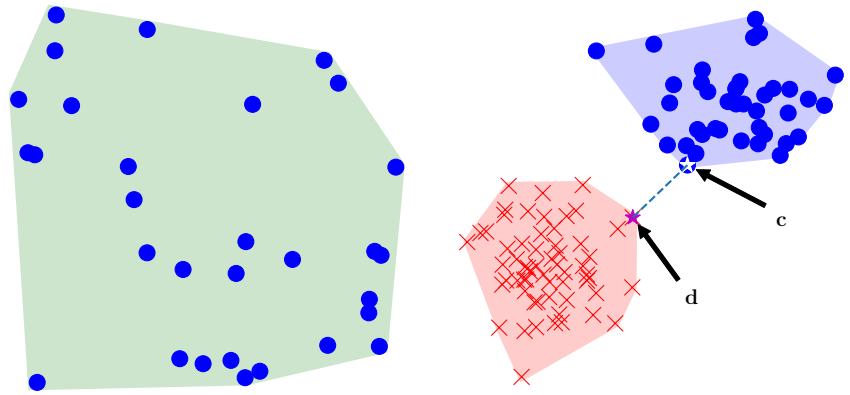


Figure 12.9 (left) Convex hull of points (right) A convex hull around positive and negative examples.

Consider two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and corresponding non-negative weights $\alpha_1, \alpha_2 \geqslant 0$ such that $\alpha_1 + \alpha_2 = 1$. The equation $\alpha_1\boldsymbol{x}_1 + \alpha_2\boldsymbol{x}_2$ describes each point on a line between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Consider what happens when we add a third point $\boldsymbol{x}_3$ along with a weight $\alpha_3 \geqslant 0$ such that $\sum_{n=1}^{3} \alpha_n = 1$. The convex combination of these three points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ span a two dimensional area. The convex hull of this area is the triangle formed by the edges corresponding to each pair of of points. As we add more points, and the number of points become greater than the number of dimensions, some of the points will be inside the convex hull, as we can see in the left of Figure 12.9.

In general, building a convex boundary of points (called the *convex hull*) can be done by introducing non-negative weights $\alpha_n \geqslant 0$ corresponding to each example $\boldsymbol{x}_n$. Then the convex hull can be described as the set

convex hull

$$\operatorname{conv}(\boldsymbol{X}) = \left\{ \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \right\} \quad \text{with} \quad \sum_{n=1}^{N} \alpha_n = 1 \quad \text{and} \quad \alpha_n \geqslant 0, \tag{12.45}$$

for all $n = 1, \ldots, N$. If the two clouds of points corresponding to the positive and negative classes are well separated, then the convex hulls do not overlap. Given the training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ we form two convex hulls, corresponding to the positive and negative classes respectively. We pick a point $\boldsymbol{c}$, which is in the convex hull of the set of positive examples, and is closest to the negative class distribution. Similarly we pick a point $\boldsymbol{d}$ in the convex hull of the set of negative examples, and is closest to the positive class distribution. Refer to the right of Figure 12.9. We draw a vector from $\boldsymbol{d}$ to $\boldsymbol{c}$

$$\boldsymbol{w} = \boldsymbol{c} - \boldsymbol{d}\,. \tag{12.46}$$

Picking the points $\boldsymbol{c}$ and $\boldsymbol{d}$ as above, and requiring them to be closest to each other is the same as saying that we want to minimize the length/ norm of $\boldsymbol{w}$, such that we end up with the corresponding optimization

problem

$$\operatorname{argmin}_{\boldsymbol{w}} \|\boldsymbol{w}\| = \operatorname{argmin}_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 . \tag{12.47}$$

Since $\boldsymbol{c}$ must be in the positive convex hull, it can be expressed as a convex combination of the positive examples, i.e., for non-negative coefficients α_n^+

$$\boldsymbol{c} = \sum_{y_n=+1} \alpha_n^+ \boldsymbol{x}_n . \tag{12.48}$$

Similarly, for the examples with negative labels we obtain

$$\boldsymbol{d} = \sum_{y_n=-1} \alpha_n^- \boldsymbol{x}_n . \tag{12.49}$$

By substituting (12.46), (12.48) and (12.49) into (12.47), we obtain the following objective function

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \left\| \sum_{y_n=+1} \alpha_n^+ \boldsymbol{x}_n - \sum_{y_n=-1} \alpha_n^- \boldsymbol{x}_n \right\|^2 . \tag{12.50}$$

Let $\boldsymbol{\alpha}$ be the set of all coefficients, i.e., the concatenation of $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$. Recall that we require that for each convex hull that their coefficients sum to one,

$$\sum_{y_n=+1} \alpha_n^+ = 1 \quad \text{and} \quad \sum_{y_n=-1} \alpha_n^- = 1 . \tag{12.51}$$

This implies the constraint

$$\sum_{n=1}^{N} y_n \alpha_n = 0. \tag{12.52}$$

This result can be seen by multiplying out the individual classes

$$\begin{aligned} \sum_{n=1}^{N} y_n \alpha_n &= \sum_{y_n=+1} (+1)\alpha_n^+ + \sum_{y_n=-1} (-1)\alpha_n^- \\ &= \sum_{y_n=+1} \alpha_n^+ - \sum_{y_n=-1} \alpha_n^- = 1 - 1 = 0 . \end{aligned} \tag{12.53}$$

The objective function (12.50) and the constraint (12.52), along with the assumption that $\boldsymbol{\alpha} \geqslant \mathbf{0}$, give us a constrained (convex) optimization problem. This optimization problem can be shown to be the same as that of the dual hard margin SVM (Bennett and Bredensteiner, 2000a).

Remark. To obtain the soft margin dual, we consider the reduced hull. The reduced hull *reduced hull* is similar to the convex hull but has an upper bound to the size of the coefficients $\boldsymbol{\alpha}$. The maximum possible value of the elements of $\boldsymbol{\alpha}$ restricts the size that the convex hull can take. In other words, the bound on $\boldsymbol{\alpha}$ shrinks the convex hull to a smaller volume (Bennett and Bredensteiner, 2000b). ♢

12.3.3 Kernels

Consider the formulation of the dual SVM (12.43). Notice that the inner product in the objective occurs only between examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. There are no inner products between the examples and the parameters. Therefore if we consider a set of features $\boldsymbol{\phi}(\boldsymbol{x}_i)$ to represent $\boldsymbol{x}_i$, the only change in the dual SVM will be to replace the inner product. This modularity, where the choice of the classification method (the SVM) and the choice of the feature representation $\boldsymbol{\phi}(\boldsymbol{x})$ can be considered separately, provides flexibility for us to explore the two problems independently.

Since $\boldsymbol{\phi}(\boldsymbol{x})$ could be a non-linear function, we can use the SVM (which assumes a linear classifier) to construct nonlinear classifiers. This provides a second avenue, in addition to the soft margin, for users to deal with a dataset that is not linearly separable. It turns out that there are many algorithms and statistical methods, which have this property that we observed in the dual SVM: the only inner products are those that occur between examples. Instead of *explicitly* defining a non-linear feature map $\boldsymbol{\phi}(\cdot)$ and computing the resulting inner product between examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we define a similarity function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For a certain class of similarity functions called *kernels*, the definition of the similarity function *implicitly* defines a non-linear feature map $\boldsymbol{\phi}(\cdot)$. Kernels are by definition functions $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for which there exists a Hilbert space $\mathcal{H}$ and $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$ a feature map such that

kernels

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle_{\mathcal{H}} \,. \tag{12.54}$$

There is a unique reproducing kernel Hilbert space associated with every kernel k (Aronszajn, 1950; Berlinet and Thomas-Agnan, 2004). In this unique association $\boldsymbol{\phi}(\boldsymbol{x}) = k(\cdot, \boldsymbol{x})$ is called the canonical feature map. This is known as the *kernel trick* (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004), as it hides away the explicit non-linear feature map. The matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$, resulting from the inner products or the application of $k(\cdot, \cdot)$ to a dataset, is called the *Gram matrix*, and is often just referred to as the *kernel matrix*. Kernels must be symmetric and positive semi-definite, i.e., every kernel matrix $\boldsymbol{K}$ must be symmetric and positive semi-definite (Section 3.2.3):

kernel trick

Gram matrix

kernel matrix

$$\forall \boldsymbol{z} \in \mathbb{R}^N \qquad \boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} \geqslant 0 \,. \tag{12.55}$$

Some popular examples of kernels for multivariate real-valued data $\boldsymbol{x}_i \in \mathbb{R}^D$ are the polynomial kernel, the Gaussian radial basis function kernel, and the rational quadratic kernel. Figure 12.10 illustrates the effect of different kernels on separating hyperplanes on an example dataset.

Remark. Unfortunately for the fledgling machine learner, there are multiple meanings of the word kernel. In this chapter, the word kernel comes from the idea of the Reproducing Kernel Hilbert Space (RKHS) (Aronszajn, 1950; Saitoh, 1988). We have discussed the idea of the kernel in

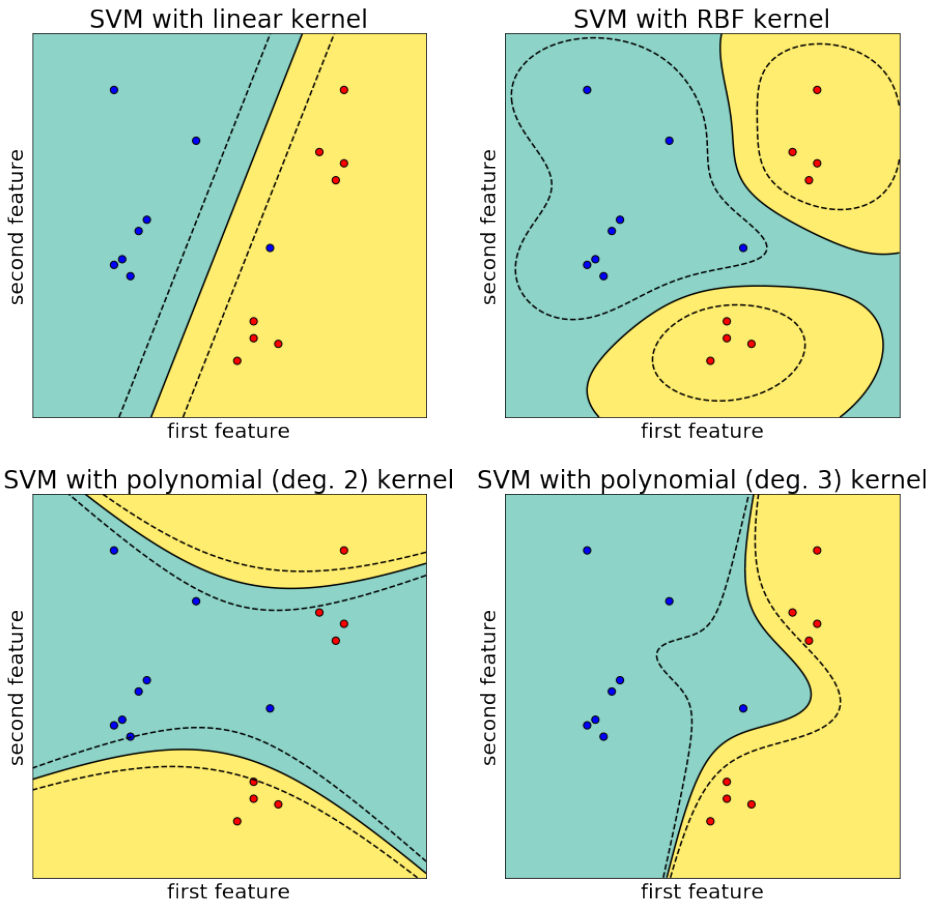


Figure 12.10 Support Vector Machine with different kernels. Note that while the decision boundary is nonlinear, the underlying problem being solved is for a linear separating hyperplane (albeit with a nonlinear kernel).

linear algebra (Section 2.7.3), where the kernel is another word for the nullspace. The third common use of the word kernel in machine learning is the smoothing kernel in kernel density estimation (Section 11.5). ♢

Since the explicit representation $\phi(\boldsymbol{x})$ is mathematically equivalent to the kernel representation $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ a practitioner will often design the kernel function, such that it can be computed more efficiently than the inner product between explicit feature maps. For example, consider the polynomial kernel, where the number of terms in the explicit expansion grows very quickly (even for polynomials of low degree) when the input dimension is large. The kernel function only requires one multiplication per input dimension, which can provide significant computational savings.

Another useful aspect of the kernel trick is that there is no need for the original data to be already represented as multivariate real-valued data. Note that the inner product is defined on the output of the function $\phi(\cdot)$, but does not restrict the input to real numbers. Hence, the function $\phi(\cdot)$ and the kernel function $k(\cdot, \cdot)$ can be defined on any object, e.g., sets,

sequences, strings and graphs (Ben-Hur et al., 2008; Gärtner, 2008; Shi et al., 2009; Vishwanathan et al., 2010).

12.3.4 Numerical Solution

We conclude our discussion of SVMs by looking at how to express the problems derived in this chapter in terms of the concepts presented in Chapter 7. We consider two different approaches for finding the optimal solution for the SVM. First we consider the loss view of SVM 8.1.2 and express this as an unconstrained optimization problem. Then we express the constrained versions of the primal and dual SVMs as quadratic programs in standard form 7.3.2.

Consider the loss function view of the SVM (12.33). This is a convex unconstrained optimization problem, but the hinge loss (12.30) is not differentiable. Therefore, we apply a subgradient approach for solving it. However, the hinge loss is differentiable almost everywhere, except for one single point at the hinge $t = 1$. At this point, the gradient is a set of possible values that lie between 0 and -1. Therefore, the subgradient g of the hinge loss is given by

$$g(t) = \begin{cases} -1 & t < 1 \\ [-1, 0] & t = 1 \\ 0 & t > 1 \end{cases} . \tag{12.56}$$

Using this subgradient above, we can apply the optimization methods presented in Section 7.1.

Both the primal and the dual SVM result in a convex quadratic programming problem (constrained optimization). Note that the primal SVM in (12.27) has optimization variables that have the size of the dimension D of the input examples. The dual SVM in (12.43) has optimization variables that have the size of the number N of examples.

To express the primal SVM in the standard form (7.35) for quadratic programming, let us assume that we use the dot product (3.6) as the inner product. We rearrange the equation for the primal SVM (12.27), such that the optimization variables are all on the right and the inequality of the constraint matches the standard form. This yields the optimization

Recall from Section 3.2 that in this book, we use the phrase dot product to mean the inner product on Euclidean vector space.

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N} \xi_n \tag{12.57}$$

$$\text{subject to} \quad \begin{matrix} -y_n \boldsymbol{x}_n^\top \boldsymbol{w} - y_n b - \xi_n \leqslant -1 \\ -\xi_n \leqslant 0 \end{matrix} \tag{12.58}$$

for all $n = 1, \ldots, N$. By concatenating the variables $\boldsymbol{w}, b, \boldsymbol{x}_n$ into one single vector, and carefully collecting the terms, we obtain the following matrix form of the soft margin SVM. In the following optimization problem, the minimization is over $[\boldsymbol{w}^\top, b, \boldsymbol{\xi}^\top]^\top \in \mathbb{R}^{D+1+N}$, and we have used

the notation: $\boldsymbol{I}_m$ to represent the identity matrix of size $m \times m$, $\mathbf{0}_{m,n}$ to represent the matrix of zeros of size $m \times n$, and $\mathbf{1}_{m,n}$ to represent the matrix of ones of size $m \times n$. The soft margin SVM can be written in the following vector form:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2} \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{I}_D & \mathbf{0}_{D,N+1} \\ \mathbf{0}_{N+1,D} & \mathbf{0}_{N+1,N+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{D+1,1} & C\mathbf{1}_{N,1} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} \tag{12.59}$$

$$\text{subject to } \begin{bmatrix} -\boldsymbol{Y}\boldsymbol{X} & -\boldsymbol{y} & -\boldsymbol{I}_N \\ \mathbf{0}_{N,D+1} & & -\boldsymbol{I}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} \leqslant \begin{bmatrix} -\mathbf{1}_{N,1} \\ \mathbf{0}_{N,1} \end{bmatrix}, \tag{12.60}$$

where $\boldsymbol{y}$ is the vector of labels $[y_1, \ldots, y_N]^\top$, $\boldsymbol{Y} = \operatorname{diag}(\boldsymbol{y})$ is an N by N matrix where the elements of the diagonal are from $\boldsymbol{y}$, and $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is the matrix obtained by concatenating all the examples.

We can similarly perform a collection of terms for the dual version of the SVM (12.43). To express the dual SVM in standard form, we first have to express the kernel matrix $\boldsymbol{K}$ such that each entry is $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Or if we are using an explicit feature representation $K_{ij} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. For convenience of notation we introduce a matrix with zeros everywhere except on the diagonal, where we store the labels, that is, $\boldsymbol{Y} = \operatorname{diag}(\boldsymbol{y})$. The dual SVM can be written as

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} - \mathbf{1}_{N,1}^\top \boldsymbol{\alpha} \tag{12.61}$$

$$\text{subject to } \begin{bmatrix} \boldsymbol{y}^\top \\ -\boldsymbol{y}^\top \\ -\boldsymbol{I}_N \\ \boldsymbol{I}_N \end{bmatrix} \boldsymbol{\alpha} \leqslant \begin{bmatrix} \mathbf{0}_{N+2,1} \\ C\mathbf{1}_{N,1} \end{bmatrix}. \tag{12.62}$$

Remark. In Section 7.3.1 and 7.3.2 we introduced the standard forms of the constraints to be inequality constraints. We will express the dual SVM's equality constraint as two inequality constraints, i.e.,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \text{is replaced by} \quad \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \quad \text{and} \quad \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b} \tag{12.63}$$

Particular software implementations of convex optimization methods may provide the ability to express equality constraints. ♢

Since there are many different possible views of the SVM, there are many approaches for solving the resulting optimization problem. The approach presented here, expressing the SVM problem in standard convex optimization form, is not often used in practice. The two main implementations of SVM solvers are (Chang and Lin, 2011) (which is open source) and (Joachims, 1999). Since SVMs have a clear and well defined optimization problem, many approaches based on numerical optimization

techniques (Nocedal and Wright, 2006) can be applied (Shawe-Taylor and Sun, 2011).

12.4 Further Reading

The SVM is one of many approaches for studying binary classification. Other approaches include the perceptron, logistic regression, Fisher discriminant, nearest neighbor, naive Bayes, and random forest (Bishop, 2006; Murphy, 2012). A short tutorial on SVMs and kernels on discrete sequences can be found in Ben-Hur et al. (2008). The development of SVMs is closely linked to empirical risk minimization 8.1, and hence the SVM has strong theoretical properties (Vapnik, 2000; Steinwart and Christmann, 2008). The book about kernel methods (Schölkopf and Smola, 2002) includes many details of support vector machines and how to optimize them. A broader book about kernel methods (Shawe-Taylor and Cristianini, 2004) also includes many linear algebra approaches for different machine learning problems.

An alternative derivation of the dual SVM can be obtained using the idea of the Legendre-Fenchel transform (Section 7.3.3). The derivation considers each term of the unconstrained formulation of the SVM (12.33) separately and calculates their convex conjugates (Rifkin and Lippert, 2007). Readers interested in the functional analysis view (also the regularization methods view) of SVMs are referred to the work by Wahba (1990). Theoretical exposition of kernels (Manton and Amblard, 2015; Aronszajn, 1950; Schwartz, 1964; Saitoh, 1988) require a basic grounding of linear operators (Akhiezer and Glazman, 1993). The idea of kernels have been generalized to Banach spaces (Zhang et al., 2009) and Kreĭn spaces (Ong et al., 2004; Loosli et al., 2016).

Observe that the hinge loss has three equivalent representations, as shown by (12.30) and (12.31), as well as the constrained optimization problem in (12.35). The formulation (12.30) is often used when comparing the SVM loss function with other loss functions (Steinwart, 2007). The two piece formulation (12.31) is convenient for computing subgradients, as each piece is linear. The third formulation (12.35), as seen in Section 12.3.4, enables the use of convex quadratic programming (Section 7.3.2) tools.

Since binary classification is a well studied task in machine learning, other words are also sometimes used, such as discrimination, separation or decision. To further add to the confusion, there are three quantities that can be the output of a binary classifier. First is the output of the linear function itself. This output can be used for ranking the examples, and binary classification can be thought of as picking a threshold on the ranked examples (Shawe-Taylor and Cristianini, 2004). The second quantity that is often considered the output of a binary classifier is after the output is passed through a non-linear function to constrain its value to a bounded range.

A common non-linear function is the sigmoid function (Bishop, 2006). When the non-linearity results in well calibrated probabilities (Gneiting and Raftery, 2007; Reid and Williamson, 2011), this is called class probability estimation. The third output of a binary classifier is the final binary decision, which is the one most commonly assumed to be the output of the classifier.

References

Abel, Niels H. 1826. *Démonstration de l'Impossibilité de la Résolution Algébrique des Équations Générales qui Passent le Quatrième Degré*. Grøndahl & Søn. pages 315

Adhikari, Ani, and DeNero, John. 2018. *Computational and Inferential Thinking: The Foundations of Data Science*. Gitbooks. pages 238

Agarwal, Arvind, and III, Hal Daumé. 2010. A Geometric View of Conjugate Priors. *Machine Learning*, **81**(1), 99–113. pages 205

Agresti, A. 2002. *Categorical Data Analysis*. Wiley. pages 254

Akaike, Hirotugu. 1974. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, **19**(6), 716–723. pages 268

Akhiezer, N.I., and Glazman, I.M. 1993. *Theory of Linear Operators in Hilbert Space*. Dover Publications, Inc. pages 371

Alpaydin, Ethem. 2010. *Introduction to Machine Learning*. The MIT Press. pages 2

Amari, Shun-ichi. 2016. *Information Geometry and Its Applications*. Springer. pages 188

Argyriou, Andreas, and Dinuzzo, Francesco. 2014. A Unifying View of Representer Theorems. Pages 748–756 of: Xing, Eric P., and Jebara, Tony (eds), *Proceedings of the 31st International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 32, no. 2. Bejing, China: PMLR. pages 363

Aronszajn, N. 1950. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, **68**, 337–404. pages 367, 371

Axler, Sheldon. 2015. *Linear Algebra Done Right*. third edn. Springer. pages 90

Bakir, Gökhan, Hofmann, Thomas, Schölkopf, Bernhard, Smola, Alexander J., Taskar, Ben, and Vishwanathan, S.V.N (eds). 2007. *Predicting Structured Data*. MIT Press. pages 263

Barber, David. 2012. *Bayesian Reasoning and Machine Learning*. Cambridge University Press. pages 2, 210, 258, 294, 347

Barndorff-Nielsen, Ole. 2014. *Information and Exponential Families: In Statistical Theory*. John Wiley and Sons. pages 209

Bartholomew, David, Knott, Martin, and Moustaki, Irini. 2011. *Latent Variable Models and Factor Analysis: A Unified Approach*. Wiley & Sons. pages 327

Beck, Amir, and Teboulle, Marc. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, **31**(3), 167–175. pages 233

Belabbas, Mohamed-Ali, and Wolfe, Patrick J. 2009. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, pnas–0810600105. pages 133

Belkin, Mikhail, and Niyogi, Partha. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, **15**(6), 1373–1396. pages 133

Ben-Hur, Asa, Ong, Cheng Soon, Sonnenburg, Sören, Schölkopf, Bernhard, and Rätsch, Gunnar. 2008. Support Vector Machines and Kernels for Computational Biology. *PLoS Computational Biology*, **4**(10), e1000173. pages 369, 371

Bennett, Kristin P., and Bredensteiner, Erin J. 2000a. Duality and Geometry in SVM Classifiers. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. pages 366

Bennett, Kristin P., and Bredensteiner, Erin J. 2000b. Geometry in Learning. In: Gorini, Catherine A. (ed), *In Geometry at Work*. The Mathematical Association of America. pages 366

Berlinet, Alain, and Thomas-Agnan, Christine. 2004. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer. pages 367

Bertsekas, Dimitri P. 1999. *Nonlinear Programming*. Athena Scientific. pages 233

Bertsekas, Dimitri P. 2009. *Convex Optimization Theory*. Athena Scientific. pages 233

Bickel, Peter J., and Doksum, Kjell. 2006. *Mathematical Statistics, Basic Ideas and Selected Topics, Vol I*. Prentice Hall. pages 210

Bickson, Danny, Dolev, Danny, Shental, Ori, Siegel, Paul H., and Wolf, Jack K. 2007. Linear Detection via Belief Propagation. In: *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. pages 263

Billingsley, Patrick. 1995. *Probability and Measure*. Wiley. pages 176, 210

Bishop, Christopher M. 1995. *Neural Networks for Pattern Recognition*. Clarendon Press. pages 295

Bishop, Christopher M. 1999. Bayesian PCA. Pages 382–388 of: *Advances in Neural Information Processing Systems*. pages 326

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag. pages vii, 2, 90, 167, 171, 173, 184, 206, 210, 258, 259, 263, 279, 294, 346, 347, 371, 372

Blei, David M., Kucukelbir, Alp, and McAuliffe, Jon D. 2017. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, **112**(518), 859–877. pages 258, 326

Bonnans, J. Frédéric, Gilbert, J. Charles, Lemaréchal, Claude, and Sagastizábal, Claudia A. 2006. *Numerical Optimization: Theoretical and Practical Aspects*. 2nd edn. Springer Verlag. pages 233

Borwein, Jonathan M., and Lewis, Adrian S. 2006. *Convex Analysis and Nonlinear Optimization*. 2nd edn. Canadian Mathematical Society. pages 233

Bottou, Leon. 1998. Online Algorithms and Stochastic Approximations. Pages 1–34 of: *Online Learning and Neural Networks*. Cambridge University Press. pages 219

Bottou, Léon, Curtis, Frank E., and Nocedal, Jorge. 2018. Optimization Methods for Large-Scale Machine Learning. *ArXiv*. pages 219, 233

Boucheron, Stephane, Lugosi, Gabor, and Massart, Pascal. 2013. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press. pages 173

Boyd, Stephen, and Vandenberghe, Lieven. 2004. *Convex Optimization*. Cambridge University Press. pages 217, 221, 223, 233

Boyd, Stephen, and Vandenberghe, Lieven. 2018. *Introduction to Applied Linear Algebra*. Cambridge University Press. pages 90

Brooks, Steve, Gelman, Andrew, Jones, Galin L., and Meng, Xiao-Li (eds). 2011. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC. pages 258

Brown, Lawrence D. 1986. *Fundamentals of Statistical Exponential Families: with Applications in Statistical Decision Theory*. Lecture Notes - Monograph Series. Institute of Mathematical Statistics. pages 207

Bryson, Arthur E. 1961. A Gradient Method for Optimizing Multi-stage Allocation Processes. In: *Proceedings of the Harvard University Symposium on Digital Computers and Their Applications*. pages 156, 158

Bubeck, Sébastien. 2015. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, **8**(3-4), 231–357. pages 233

Bühlmann, Peter, and Geer, Sara Van De. 2011. *Statistics for High-Dimensional Data*. Springer. pages 250

Burges, Christopher. 2010. Dimension Reduction: A Guided Tour. *Foundations and Trends in Machine Learning*, **2**(4), 275–365. pages 327

Carroll, J Douglas, and Chang, Jih-Jie. 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of ?Eckart-Young? decomposition. *Psychometrika*, **35**(3), 283–319. pages 133, 134

Casella, George, and Berger, Roger L. 2002. *Statistical Inference*. Duxbury. pages 182, 188, 191, 255

Çinlar, Erhan. 2011. *Probability and Stochastics*. Springer. pages 210

Chang, Chih-Chung, and Lin, Chih-Jen. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. pages 370

Cheeseman, Peter. 1985. In Defense of Probability. In: *IJCAI*. pages 255

Chollet, Francois, and Allaire, J. J. 2018. *Deep Learning with R*. Manning Publications. pages 2

Codd, Edgar F. 1990. *The Relational Model for Database Management*. Addison-Wesley Longman Publishing. pages 238

Cunningham, John P., and Ghahramani, Zoubin. 2015. Linear Dimensionality Reduction: Survey, Insights, and Generalizations. *Journal of Machine Learning Research*, **16**, 2859–2900. pages 327

Datta, Biswa Nath. 2010. *Numerical Linear Algebra and Applications*. Vol. 116. Siam. pages 125

Davidson, Anthony C., and Hinkley, David V. 1997. *Bootstrap Methods and their Application*. Cambridge University Press. pages 250

Dean, Jeffrey, Corrado, Greg S, Monga, Rajat, Chen, Kai, Devin, Matthieu, Le, Quoc V, Mao, Mark Z, Ranzato, Marc Aurelio, Senior, Andrew, Tucker, Paul, Yang, Ke, and Ng, Andrew Y. 2012. Large Scale Distributed Deep Networks. Pages 1–11 of: *Advances in Neural Information Processing Systems*. pages 219

Deisenroth, Marc P., and Mohamed, Shakir. 2012. Expectation Propagation in Gaussian Process Dynamical Systems. Pages 2618–2626 of: *Advances in Neural Information Processing Systems*. pages 263

Deisenroth, Marc P., and Ohlsson, Henrik. 2011. A General Perspective on Gaussian Filtering and Smoothing: Explaining Current and Deriving New Algorithms. In: *Proceedings of the American Control Conference*. pages 196

Deisenroth, Marc P., Fox, Dieter, and Rasmussen, Carl E. 2015. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(2), 408–423. pages 87

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, **39**(1), 1–38. pages 341

Deng, Li, Seltzer, Michael L, Yu, Dong, Acero, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey E. 2010. Binary Coding of Speech Spectrograms using a Deep Auto-Encoder. Pages 1692–1695 of: *Interspeech*. pages 79

Devroye, Luc. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag. pages 201

Domingos, Pedro. 2012. A few useful things to know about machine learning. *Communications of the ACM*, **55**, 78–87. pages 12

Donoho, David L, and Grimes, Carrie. 2003. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, **100**(10), 5591–5596. pages 133

Dostál, Zdeněk. 2009. *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*. Springer-Verlag. pages 364

Douven, Igor. 2017. Abduction. In: Zalta, Edward N. (ed), *The Stanford Encyclopedia of Philosophy*, summer 2017 edn. Metaphysics Research Lab, Stanford University. pages 243

Downey, Allen B. 2014. *Think Stats: Exploratory Data Analysis*. 2nd edn. O'Reilly Media. pages 209

Dreyfus, Stuart. 1962. The Numerical Solution of Variational Problems. *Journal of Mathematical Analysis and Applications*, **5**(1), 30–45. pages 156, 158

Drumm, Volker, and Weil, Wolfgang. 2001. *Lineare Algebra und Analytische Geometrie*. Lecture Notes, Universität Karlsruhe. pages 18, 53

Dudley, R. M. 2002. *Real Analysis and Probability*. Cambridge University Press. pages 210

Efron, Bradley, and Hastie, Trevor. 2016. *Computer Age Statistical Inference: Algorithms, Evidence and Data Science*. Cambridge University Press. pages 202, 253, 255

Efron, Bradley, and Tibshirani, Robert J. 1993. *An Introduction to the Bootstrap*. Chapman and Hall/CRC. pages 250

Elliott, Conal. 2009. Beautiful differentiation. In: *International Conference on Functional Programming (ICFP)*. pages 166

Evgeniou, Theodoros, Pontil, Massimiliano, and Poggio, Tomaso. 2000. Statistical Learning Theory: A Primer. *International Journal of Computer Vision*, **38**(1), 9–13. pages 249

Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, and Lin, Chih-Jen. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, **9**, 1871–1874. pages 359

Gal, Yarin, van der Wilk, Mark, and Rasmussen, Carl E. 2014. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In: *Advances in Neural Information Processing Systems*. pages 219

Gärtner, Thomas. 2008. *Kernels for Structured Data*. World Scientific. pages 369

Gavish, Matan, and Donoho, David L. 2014. The Optimal Hard Threshold for Singular Values is $4\sqrt{3}$. *IEEE Transactions on Information Theory*, **60**(8), 5040–5053. pages 326

Ghahramani, Zoubin. 2015. Probabilistic Machine Learning and Artificial Intelligence. *Nature*, **521**, 452–459. pages 178, 258

Ghahramani, Zoubin, and Roweis, Sam T. 1999. Learning Nonlinear Dynamical Systems using an EM Algorithm. In: Kearns, M. S., Solla, S. A., and Cohn, D. A. (eds), *Advances in Neural Information Processing Systems*, vol. 11. The MIT Press. pages 347

Gilks, Walter R., Richardson, Sylvia, and Spiegelhalter, David J. 1996. *Markov Chain Monte Carlo in Practice*. Chapman & Hall. pages 326

Gneiting, Tilmann, and Raftery, Adrian E. 2007. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, **102**(477), 359–378. pages 372

Goh, Gabriel. 2017. Why Momentum Really Works. *Distill*. pages 217, 233

Gohberg, Israel, Goldberg, Seymour, and Krupnik, Nahum. 2012. *Traces and determinants of linear operators*. Vol. 116. Birkhäuser. pages 99

Gonçalves, Hugo. 2014. *Legendre and Legendre-Fenchel transforms*. Accessed on 3 March 2018. pages 233

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. pages 210, 240, 263, 295

Griewank, Andreas, and Walther, Andrea. 2003. Introduction to Automatic Differentiation. *PAMM*, **2(1)**, 45–49. pages 166

Griewank, Andreas, and Walther, Andrea. 2008. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. second edn. SIAM, Philadelphia. pages 166

Grinstead, Charles M., and Snell, J. Laurie. 1997. *Introduction to Probability*. American Mathematical Society. pages 172, 194, 209

Hacking, Ian. 2001. *Probability and Inductive Logic*. Cambridge University Press. pages 209

Hall, Peter. 1992. *The Bootstrap and Edgeworth Expansion*. Springer. pages 250

Hasselblatt, Boris, and Katok, Anatole. 2003. *A first course in dynamics with a Panorama of Recent Developments*. Cambridge University Press. pages 172

Hazan, Elad. 2015. Introduction to Online Convex Optimization. *Foundations and Trends*, **2**(3-4), 157–325. pages 233

Hensman, James, Fusi, Nicolò, and Lawrence, Neil D. 2013. Gaussian Processes for Big Data. In: Nicholson, A., and Smyth, P. (eds), *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press. pages 219

Herbrich, Ralf, Minka, Tom, and Graepel, Thore. 2007. TrueSkill(TM): A Bayesian Skill Rating System. Pages 569–576 of: *Advances in Neural Information Processing Systems*. MIT Press. pages 263

Hiriart-Urruty, Jean-Baptiste, and Lemaréchal, Claude. 2001. *Fundamentals of Convex Analysis*. Springer. pages 229, 233

Hoffman, Matthew D., Blei, David M., Wang, Chong, and Paisley, John. 2013. Stochastic Variational Inference. *Journal of Machine Learning Research*, **14**(1), 1303–1347. pages 219

Hofmann, Thomas, Schölkopf, Bernhard, and Smola, Alexander J. 2008. Kernel methods in machine learning. *Ann. Statist.*, **36**(3), 1171–1220. pages 363

Hogben, L. (ed). 2013. *Handbook of Linear Algebra*. 2nd edn. Discrete Mathematics and Its Applications. Chapman and Hall. pages 18

Hogben, Leslie. 2006. *Handbook of linear algebra*. Chapman and Hall/CRC. pages 103

Hotelling, Harold. 1933. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, **24**, 417–441. pages 79, 90, 297, 301

Imbens, Guido W., and Rubin, Donald B. 2015. *Causal Inference for statistics, social and biomedical sciences*. Cambridge University Press. pages 263

Jacod, Jean, and Protter, Philip. 2004. *Probability Essentials*. 2nd edn. Springer-Verlag. pages 171, 210

Jaynes, Edwin T. 2003. *Probability Theory: The Logic of Science*. Cambridge University Press. pages 169, 170, 172, 178, 179, 210, 255

Jefferys, Willian H., and Berger, James O. 1992. Ockham's Razor and Bayesian Analysis. *American Scientist*, **80**, 64–72. pages 265

Joachims, Thorsten. 1999. Making large-Scale SVM Learning Practical. In: Schölkopf, B., Burges, C., and Smola, A. (eds), *Advances in Kernel Methods - Support Vector Learning*. MIT Press. pages 370

Julier, Simon J., and Uhlmann, Jeffrey K. 1997. A New Extension of the Kalman Filter to Nonlinear Systems. Pages 182–193 of: *Proceedings of AeroSense: 11th Symposium on Aerospace/Defense Sensing, Simulation and Controls*. pages 167

Kalman, Dan. 1996. A singularly valuable decomposition: the SVD of a matrix. *The College Mathematics Journal*, **27**(1), 2–23. pages 115

Kalman, Rudolf E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME—Journal of Basic Engineering*, **82**(Series D), 35–45. pages 196

Katz, Victor J. 2004. *A History of Mathematics*. Pearson/Addison-Wesley. pages 105

Kelley, Henry J. 1960. Gradient Theory of Optimal Flight Paths. *Ars Journal*, **30**(10), 947–954. pages 156, 158

Kimeldorf, George S., and Wahba, Grace. 1970. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, **41**(2), 495–502. pages 363

Kittler, J., and Föglein, J. 1984. Contextual Classification of Multispectral Pixel Data. *Image and Vision Computing*, **2**(1), 13–29. pages 263

Kolda, Tamara G, and Bader, Brett W. 2009. Tensor decompositions and applications. *SIAM review*, **51**(3), 455–500. pages 134

Koller, Daphne, and Friedman, Nir. 2009. *Probabilistic Graphical Models*. MIT Press. pages 263

Lang, Serge. 1987. *Linear Algebra*. Springer-Verlag, New York. pages 111

Lawrence, Neil. 2005. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, **6**(Nov.), 1783–1816. pages 328

Leemis, Lawrence M., and McQueston, Jacquelyn T. 2008. Univariate Distribution Relationships. *The American Statistician*, **62**(1), 45–53. pages 202, 204

Lehmann, Erich L., and Romano, Joseph P. 2005. *Testing Statistical Hypotheses*. Springer. pages 191

Lehmann, Erich Leo, and Casella, George. 1998. *Theory of Point Estimation*. Springer. pages 207, 210, 253

Liesen, Jörg, and Mehrmann, Volker. 2015. *Linear Algebra*. Springer Undergraduate Mathematics Series. Springer. pages 18

Loosli, Gaëlle, Canu, Stéphane, and Ong, Cheng Soon. 2016. Learning SVM in Kreĭn Spaces. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, **38**(6), 1204–1216. pages 371

Luenberger, David G. 1969. *Optimization by Vector Space Methods*. John Wiley and Sons. pages 233

MacKay, Davic J. C. 2003a. *Information Theory, Inference, and Learning Algorithms*. The Edinburgh Building, Cambridge CB2 2RU, UK: Cambridge University Press. pages 265, 325, 347

MacKay, David J. C. 1992. Bayesian Interpolation. *Neural Computation*, **4**, 415–447. pages 265

MacKay, David J. C. 1998. Introduction to Gaussian Processes. Pages 133–165 of: Bishop, C. M. (ed), *Neural Networks and Machine Learning*, vol. 168. Berlin, Germany: Springer. pages 296

MacKay, David J. C. 2003b. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. pages 2, 210

Magnus, Jan R., and Neudecker, Heinz. 2007. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. 3rd edn. John Wiley & Sons. pages 166

Manton, Jonathan H., and Amblard, Pierre-Olivier. 2015. A Primer on Reproducing Kernel Hilbert Spaces. *Foundations and Trends in Signal Processing*, **8**(1–2), 1–126. pages 371

Markovsky, Ivan. 2011. *Low rank approximation: algorithms, implementation, applications*. Springer Science & Business Media. pages 134

Maybeck, Peter S. 1979. *Stochastic Models, Estimation, and Control*. Mathematics in Science and Engineering, vol. 141. Academic Press, Inc. pages 167

McCullagh, Peter, and Nelder, John A. 1989. *Generalized Linear Models*. second edn. CRC Press. pages 254

McEliece, Robert J., MacKay, David J. C., and Cheng, Jung-Fu. 1998. Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm. *IEEE Journal on Selected Areas in Communications*, **16**(2), 140–152. pages 263

Meyer, Carl D. 2000. *Matrix Analysis and Applied Linear Algebra*. Vol. 71. SIAM. pages 103

Mika, Sebastian, Rätsch, Gunnar, Weston, Jason, Schölkopf, Bernhard, and Müller, Klaus-Robert. 1999. Fisher discriminant analysis with kernels. Pages 41–48 of: *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* Ieee. pages 133

Minka, Tom. 2001. Automatic Choice of dimensionality of PCA. In: *Neural Information Processing Systems (NIPS)*. pages 326

Mitchell, Tom. 1997. *Machine Learning*. McGraw Hill. pages 247

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K., Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature*, **518**(Feb.), 529–533. pages 219

Moonen, Marc, and De Moor, Bart. 1995. *SVD and Signal Processing, III: Algorithms, Architectures and Applications*. Elsevier. pages 134

Müller, Andreas C., and Guido, Sarah. 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Publishing. pages 2

Murphy, Kevin P. 2012. *Machine Learning: A Proabilistic Perspective*. Cambridge, MA, USA: MIT Press. pages 2, 197, 204, 210, 258, 265, 267, 268, 294, 327, 347, 371

Neal, Radford M. 1996. *Bayesian Learning for Neural Networks*. Ph.D. thesis, Department of Computer Science, University of Toronto. pages 296

Neal, Radford M., and Hinton, Geoffrey E. 1999. A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants. Pages 355–368 of: Jordan, M. I. (ed), *Learning in Graphical Models*. MIT Press. pages 341

Nelsen, Roger. 2006. *An Introduction to Copulas*. Springer. pages 191

Neumaier, Arnold. 1998. Solving Ill-Conditioned and Singular Linear Systems: A Tutorial on Regularization. *SIAM Review*, **40**, 636–666. pages 250

Nocedal, Jorge, and Wright, Stephen J. 2006. *Numerical Optimization*. Springer Series in Operations Research. Springer. pages 161, 233, 371

Nowozin, Sebastian, Gehler, Peter V., Jancsary, Jeremy, and Lampert, Christoph H. (eds). 2014. *Advanced Structured Prediction*. MIT Press. pages 263

O'Hagan, Anthony. 1991. Bayes-Hermite Quadrature. *Journal of Statistical Planning and Inference*, **29**, 245–260. pages 268

Ong, Cheng Soon, Mary, Xavier, Canu, Stéphane, , and Smola, Alexander J. 2004. Learning with non-positive kernels. Pages 639–646 of: *International Conference on Machine Learning (ICML)*. pages 371

Ormoneit, Dirk, Sidenbladh, Hedvig, Black, Michael J, and Hastie, Trevor. 2001. Learning and tracking cyclic human motion. Pages 894–900 of: *Advances in Neural Information Processing Systems*. pages 134

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Tech. rept. Stanford InfoLab. pages 108, 315

Parzen, Emanuel. 1962. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, **33**(3), 1065–1076. pages 348

Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. pages 170, 262

Pearl, Judea. 2009. *Causality: Models, Reasoning and Inference*. 2nd edn. Cambridge University Press. pages 259, 263

Pearson, Karl. 1895. Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **186**, 343–414. pages 348

Pearson, Karl. 1901a. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11), 559–572. pages 133

Pearson, Karl. 1901b. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, **2**(11), 559–572. pages 78, 90, 297, 307

Peters, Jonas, Janzing, Dominik, and Schölkopf, Bernhard. 2017. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press. pages 263

Petersen, K. B., and Pedersen, M. S. 2012 (nov). *The Matrix Cookbook*. Tech. rept. Technical University of Denmark. Version 20121115. pages 155

Pollard, David. 2002. *A User's Guide to Measure Theoretic Probability*. Cambridge University Press. pages 210

Polyak, Roman A. 2016. The Legendre Transformation in Modern Optimization. Pages 437–507 of: *Optimization and Its Applications in Control and Data Sciences. Springer Optimization and Its Applications*. pages 233

Press, William H., Teukolsky, Saul A., Vetterling, William T., and Flannery, Brian P. 2007. *Numerical Recipes: The Art of Scientific Computing*. third edn. Cambridge University Press. pages 125, 133

Raschka, Sebastian, and Mirjalili, Vahid. 2017. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*. Packt Publishing. pages 2

Rasmussen, Carl E., and Ghahramani, Zoubin. 2003. Bayesian Monte Carlo. Pages 489–496 of: Becker, S., Thrun, S., and Obermayer, K. (eds), *Advances in Neural Information Processing Systems 15*. Cambridge, MA, USA: The MIT Press. pages 268

Rasmussen, Carl E., and Williams, Christopher K. I. 2006. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: The MIT Press. pages 90, 196, 210, 295, 296

Rasmussen, Carl Edward, and Ghahramani, Zoubin. 2001. Occam's Razor. Pages 294–300 of: *Advances in Neural Information Processing Systems 13*. The MIT Press. pages 268

Reid, Mark, and Williamson, Robert C. 2011. Information, Divergence and Risk for Binary Experiments. *Journal of Machine Learning Research*, **12**, 731–817. pages 372

Rezende, Danilo Jimenez, and Mohamed, Shakir. 2015. Variational Inference with Normalizing Flows. In: *International Conference on Machine Learning*. pages 209

Rifkin, Ryan M., and Lippert, Ross A. 2007. Value Regularization and Fenchel Duality. *Journal of Machine Learning Research*, **8**, 441–479. pages 371

Rockafellar, R. Tyrrell. 1970. *Convex Analysis*. Princeton University Press. pages 233

Rogers, Simon, and Girolami, Mark. 2016. *A First Course in Machine Learning*. Second edn. Chapman and Hall/CRC. pages 2, 346

Rosenbaum, Paul R. 2017. *Observation & Experiment: An Introduction to Causal Inference*. Harvard University Press. pages 263

Rosenblatt, Murray. 1956. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, **27**(3), 832–837. pages 348

Roweis, Sam, and Ghahramani, Zoubin. 1999. A Unifying Review of Linear Gaussian Models. *Neural Computation*, **11**(2), 305–345. pages 197, 347

Roweis, Sam T. 1998. EM Algorithms for PCA and SPCA. Pages 626–632 of: *Advances in Neural Information Processing Systems*. pages 326

Roy, Anindya, and Banerjee, Sudipto. 2014. *Linear algebra and matrix analysis for statistics*. Chapman and Hall/CRC. pages 115

Rubinstein, Reuven Y, and Kroese, Dirk P. 2016. *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons. pages 133

Ruffini, Paolo. 1799. *Teoria Generale delle Equazioni, in cui si Dimostra Impossibile la Soluzione Algebraica delle Equazioni Generali di Grado Superiore al Quarto*. Stamperia di S. Tommaso d'Aquino. pages 315

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. 1986. Learning Representations by Back-propagating Errors. *Nature*, **323**(6088), 533–536. pages 156, 158, 217

Saitoh, Saburou. 1988. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical. pages 367, 371

Schölkopf, Bernhard, and Smola, Alexander J. 2002. *Learning with Kernels—Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: The MIT Press. pages 90, 295, 327, 354, 364, 367, 371

Schölkopf, Bernhard, Smola, Alexander, and Müller, Klaus-Robert. 1997. Kernel principal component analysis. Pages 583–588 of: *International Conference on Artificial Neural Networks*. Springer. pages 90

Schölkopf, Bernhard, Smola, Alexander J., and Müller, Klaus-Robert. 1998. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, **10**(5), 1299–1319. pages 327

Schölkopf, Bernhard, Herbrich, Ralf, and Smola, Alexander J. 2001. A generalized representer theorem. Pages 416–426 of: *International Conference on Computational Learning Theory (COLT)*. pages 363

Schwartz, Laurent. 1964. Sous espaces Hilbertiens d'espaces vectoriels topologiques et noyaux associés. *Journal d'Analyse Mathématique*, **13**, 115–256. in French. pages 371

Schwarz, Gideon E. 1978. Estimating the Dimension of a Model. *Annals of Statistics*, **6**(2), 461–464. pages 268

Shalev-Shwartz, Shai, and Ben-David, Shai. 2014. *Understanding Machine Leanring: From Theory to Algorithms*. Cambridge University Press. pages 2, 173, 249, 355

Shawe-Taylor, John, and Cristianini, Nello. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press. pages 367, 371

Shawe-Taylor, John, and Sun, Shiliang. 2011. A review of optimization methodologies in support vector machines. *Neurocomputing*, **74**(17), 3609–3618. pages 371

Shental, O., Bickson, D., P. H. Siegel and, J. K. Wolf, and Dolev, D. 2008. Gaussian Belief Propagatio Solver for Systems of Linear Equations. In: *IEEE International Symposium on Information Theory*. pages 263

Shewchuk, Jonathan Richard. 1994 (August). *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Carnegie Mellon University, Edition 1 1/4. pages 232

Shi, Jianbo, and Malik, Jitendra. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, **22**(8), 888–905. pages 133

Shi, Qinfeng, Petterson, James, Dror, Gideon, Langford, John, Smola, Alex, and Vishwanathan, S.V.N. 2009. Hash Kernels for Structured Data. *Journal of Machine Learning Research*, 2615–2637. pages 369

Shiryayev, A. N. 1984. *Probability*. Springer. pages 210

Shor, Naum Z. 1985. *Minimization Methods for Non-differentiable Functions*. Springer. pages 233

Shotton, Jamie, Winn, John, Rother, Carsten, and Criminisi, Antonio. 2006. TextonBoost: Joint Appearance, Shape and Context Modeling for Mulit-Class Object Recognition and Segmentation. In: *Proceedings of the European Conference on Computer Vision*. pages 263

Spearman, Charles. 1904. "General Intelligence," Objectively Determined and Measured. *The American Journal of Psychology*, **15**(2), 201–292. pages 327

Spiegelhalter, David, and Smith, A. F. M. 1980. Bayes Factors and Choice Criteria for Linear Models. *Journal of the Royal Statistical Society B*, **42**(2), 213–220. pages 265

Steinwart, Ingo. 2007. How to Compare Different Loss Functions and Their Risks. *Constructive Approximation*, **26**, 225–287. pages 371

Steinwart, Ingo, and Christmann, Andreas. 2008. *Support Vector Machines*. Springer. pages 349, 353, 355, 359, 371

Stoer, Josef, and Burlirsch, Roland. 2002. *Introduction to Numerical Analysis*. Springer. pages 90, 268

Strang, Gilbert. 1993. The fundamental theorem of linear algebra. *The American Mathematical Monthly*, **100**(9), 848–855. pages 115

Strang, Gilbert. 2003. *Introduction to Linear Algebra*. 3rd edn. Wellesley-Cambridge Press. pages 18, 76, 294

Stray, Jonathan. 2016. *The Curious Journalist's Guide to Data*. Tow Center for Digital Journalism at Columbia's Graduate School of Journalism. pages 238

Strogatz, Steven. 2014. Writing about Math for the Perplexed and the Traumatized. *Notices of the American Mathematical Society*, **61**(3), 286–291. pages 2

Sucar, Luis E., and Gillies, Duncan F. 1994. Probabilistic Reasoning in High-Level Vision. *Image and Vision Computing*, **12**(1), 42–60. pages 263

Szeliski, Richard, Zabih, Ramin, Scharstein, Daniel, Veksler, Olga, Kolmogorov, Vladimir, Agarwala, Aseem, Tappen, Marshall, and Rother, Carsten. 2008. A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-based Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(6), 1068–1080. pages 263

Tenenbaum, Joshua B, De Silva, Vin, and Langford, John C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science*, **290**(5500), 2319–2323. pages 133

Tibshirani, Robert. 1996. Regression Selection and Shrinkage via the Lasso. *Journal of the Royal Statistical Society B*, **58**(1), 267–288. pages 280, 296

Tipping, Michael E., and Bishop, Christopher M. 1999. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B*, **61**(3), 611–622. pages 197, 320, 325

Titsias, Michalis K., and Lawrence, Neil D. 2010. Bayesian Gaussian Process Latent Variable Model. Pages 844–851 of: Teh, Y. W., and Titterington, D. M. (eds), *Proceedings of the International Conference on Artificial Intelligence and Statistics*. JMLR W&CP, vol. 9. pages 328

Toussaint, Marc. 2012. *Some Notes on Gradient Descent*. pages 216, 233

Trefethen, Lloyd N, and Bau III, David. 1997. *Numerical Linear Algebra*. Vol. 50. Siam. pages 134, 216

Tucker, Ledyard R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, **31**(3), 279–311. pages 134

Vapnik, Vladimir. 2000. *The Nature of Statistical Learning Theory*. Springer Verlag. pages 355, 371

Vapnik, Vladimir N. 1998. *Statistical Learning Theory*. Wiley. pages 249

Vapnik, Vladimir N. 1999. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, **10**(5), 988–999. pages 249

Vishwanathan, S.V.N., Schraudolph, Nicol N., Kondor, Risi, and Borgwardt, Karsten M. 2010. Graph Kernels. *Journal of Machine Learning Research*, **11**, 1201–1242. pages 369

von Luxburg, Ulrike, and Schölkopf, Bernhard. 2011. *Statistical Learning Theory: Models, Concepts, and Results*. Vol. 10. Amsterdam, Netherlands: Elsevier North Holland. Pages 651–706. pages 249

Wahba, Grace. 1990. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics. pages 371

Wasserman, Larry. 2004. *All of Statistics*. Springer. pages 189

Wasserman, Larry. 2007. *All of Nonparametric Statistics*. Springer. pages 207

Wickham, Hadley. 2014. Tidy Data. *Journal of Statistical Software*, **59**. pages 238

Williams, Christopher K. I. 1997. Computing with Infinite Networks. In: *Neural Information Processing Systems*. pages 296

Yu, Yaoliang, Cheng, Hao, Schuurmans, Dale, and Szepesvári, Csaba. 2013. Characterizing the representer theorem. Pages 570–578 of: *International Conference on Machine Learning (ICML)*. pages 363

Zhang, Haizhang, Xu, Yuesheng, and Zhang, Jun. 2009. Reproducing Kernel Banach Spaces for Machine Learning. *Journal of Machine Learning Research*, **10**, 2741–2775. pages 371

Zia, Royce K. P., Redish, Edward F., and McKay, Susan R. 2009. Making sense of the Legendre transform. *American Journal of Physics*, **77**(614). pages 233

Index