

1

650

Introduction and Motivation

651 Machine learning is about designing algorithms that automatically extract
652 valuable from data. The emphasis here is on “automatic”, i.e., machine
653 learning is concerned about general-purpose methodologies that can be
654 applied to many datasets, while producing something that is meaningful.
655 There are three concepts that are at the core of machine learning: data, a
656 model and learning

657 Since machine learning is inherently data driven, *data* is at the core
658 of machine learning. The goal of machine learning is to design general-
659 purpose methodologies to extract valuable patterns from data, ideally
660 without much domain-specific expertise. For example, given a large corpus
661 of documents (e.g., books in many libraries), machine learning methods
662 can be used to automatically find relevant topics that are shared across
663 documents (Hoffman et al., 2010). To achieve this goal, we design *mod-*
664 *els* that are typically related to the process that generates data, similar to
665 the dataset we are given. For example, in a regression setting, the model
666 would describe a function that maps inputs to real-valued outputs. To
667 paraphrase Mitchell (1997): A model is said to learn from data if its per-
668 formance on a given task improves after the data is taken into account.
669 The goal is to find good models that generalize well to yet unseen data,
670 which we may care about in the future. *Learning* can be understood as a
671 way to automatically find patterns and structure in data by optimizing the
672 parameters of the model.

673 While machine learning has seen many success stories, and software is
674 readily available to design and train rich and flexible machine learning
675 systems, we believe that the mathematical foundations of machine learning
676 are important in order to understand fundamental principles upon
677 which more complicated machine learning systems are built. Understan-
678 ding these principles can facilitate creating new machine learning solutions,
679 understanding and debugging existing approaches and learning about the
680 inherent assumptions and limitations of the methodologies we are work-
681 ing with.

13

682 1.1 Finding Words for Intuitions

683 A challenge we face regularly in machine learning is that concepts and
 684 words are slippery, and a particular component of the machine learning
 685 system can be abstracted to different mathematical concepts. For example,
 686 the word “algorithm” is used in at least two different senses in the con-
 687 text of machine learning. In the first sense, we use the phrase “machine
 688 learning algorithm” to mean a system that makes predictions based on in-
 689 put data. We refer to these algorithms as *predictors*. In the second sense,
 690 we use the exact same phrase “machine learning algorithm” to mean a
 691 system that adapts some internal parameters of the predictor so that it
 692 performs well on future unseen input data. Here we refer to this adapta-
 693 tion as *training* a system.

694 This book will not resolve the issue of ambiguity, but we want to high-
 695 light upfront that, depending on the context, the same expressions can
 696 mean different things. However, we attempt to make the context suffi-
 697 ciently clear to reduce the level of ambiguity.

698 The first part of this book introduces the mathematical concepts and
 699 foundations needed to talk about the three main components of a machine
 700 learning system: data, models and learning. We will briefly outline these
 701 components here, and we will revisit them again in Chapter 8 once we
 702 have discussed the necessary mathematical concepts.

703 While not all data is numerical it is often useful to consider data in
 704 a number format. In this book, we assume that *data* has already been
 705 appropriately converted into a numerical representation suitable for read-
 706 ing into a computer program. Therefore, we think of data as vectors. As
 707 another illustration of how subtle words are, there are (at least) three
 708 different ways to think about vectors: a vector as an array of numbers (a
 709 computer science view), a vector as an arrow with a direction and magni-
 710 tude (a physics view), and a vector as an object that obeys addition and
 711 scaling (a mathematical view).

712 A *model* is typically used to describe a process for generating data, sim-
 713 ilar to the dataset at hand. Therefore, good models can also be thought of
 714 simplified versions of the real (unknown) data-generating process, captur-
 715 ing aspects that are relevant for modeling the data and extracting hidden
 716 patterns from it. A good model can then be used to predict what would
 717 happen in the real world without performing real-world experiments.

718 We now come to the crux of the matter, the *learning* component of
 719 machine learning. Assume we are given a dataset and a suitable model.
 720 *Training* the model means to use the data available to optimize some pa-
 721 rameters of the model with respect to a utility function that evaluates how
 722 well the model predicts the training data. Most training methods can be
 723 thought of as an approach analogous to climbing a hill to reach its peak.
 724 In this analogy, the peak of the hill corresponds to a maximum of some
 725 desired performance measure. However, in practice, we are interested in

726 the model to perform well on unseen data. Performing well on data that
727 we have already seen (training data) may only mean that we found a
728 good way to memorize the data. However, this may not generalize well to
729 unseen data, and, in practical applications, we often need to expose our
730 machine learning system to situations that it has not encountered before.

731 Let us summarize the main concepts of machine learning that we cover
732 in this book:

- 733 • We represent data as vectors.
- 734 • We choose an appropriate model, either using the probabilistic or optimiza-
- 735 • We learn from available data by using numerical optimization methods
736 with the aim that the model performs well on data not used for training.

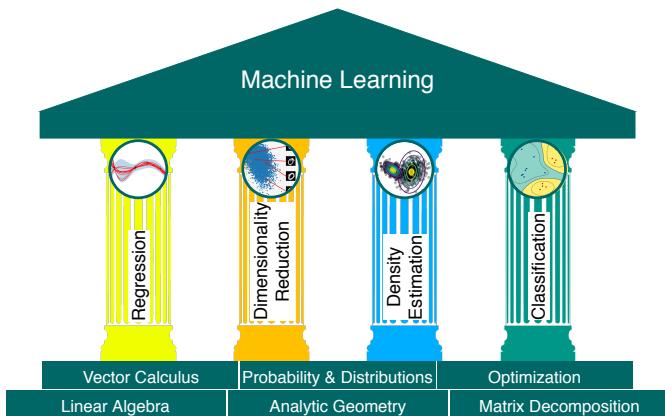
738 1.2 Two Ways to Read this Book

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

- 741 • **Bottom-up:** Building up the concepts from foundational to more ad-
- 742 vanced. This is often the preferred approach in more technical fields,
743 such as mathematics. This strategy has the advantage that the reader
744 at all times is able to rely on their previously learned concepts. Unfor-
745 tunately, for a practitioner many of the foundational concepts are not
746 particularly interesting by themselves, and the lack of motivation means
747 that most foundational definitions are quickly forgotten.
- 748 • **Top-down:** Drilling down from practical needs to more basic require-
749 ments. This goal-driven approach has the advantage that the reader
750 knows at all times why they need to work on a particular concept, and
751 there is a clear path of required knowledge. The downside of this strat-
752 egy is that the knowledge is built on potentially shaky foundations, and
753 the reader has to remember a set of words for which they do not have
754 any way of understanding.

755 We decided to write this book in a modular way to separate foundational
756 (mathematical) concepts from applications so that this book can be read
757 in both ways. The book is split into two parts, where Part I lays the math-
758 ematical foundations and Part II applies the concepts from Part I to a set
759 of fundamental machine learning problems, which form four pillars of
760 machine learning as illustrated in Figure 1.1: regression, dimensionality
761 reduction, density estimation, and classification. Chapters in Part I mostly
762 build upon the previous ones, but it is possible to skip a chapter and work
763 backward if necessary. Chapters in Part II are only loosely coupled and
764 can be read in any order. There are many pointers forward and backward
765 between the two parts of the book to link mathematical concepts with
766 machine learning algorithms.

Figure 1.1 The foundations and four pillars of machine learning.



767 *Of course there are more than two ways to read this book.* Most readers
 768 learn using a combination of top-down and bottom-up approaches, some-
 769 times building up basic mathematical skills before attempting more com-
 770 plex concepts, but also choosing topics based on applications of machine
 771 learning.

772 *Part I is about Mathematics*

773 The four pillars of machine learning we cover in this book (see Figure 1.1)
 774 require a solid mathematical foundation, which is laid out in Part I.

775 We represent numerical data as vectors and represent a table of such
 776 data as a matrix. The study of vectors and matrices is called *linear algebra*,
 777 which we introduce in Chapter 2. The collection of vectors as a matrix is
 778 also described there.

779 Given two vectors representing two objects in the real world we want
 780 to make statements about their similarity. The idea is that vectors that
 781 are similar should be predicted to have similar outputs by our machine
 782 learning algorithm (our predictor). To formalize the idea of similarity be-
 783 tween vectors, we need to introduce operations that take two vectors as
 784 input and return a numerical value representing their similarity. The con-
 785 struction of similarity and distances is central to *analytic geometry* and is
 786 discussed in Chapter 3.

787 In Chapter 4, we introduce some fundamental concepts about mat-
 788 rices and *matrix decomposition*. Some operations on matrices are extremely
 789 useful in machine learning, and they allow for an intuitive interpretation
 790 of the data and more efficient learning.

791 We often consider data to be noisy observations of some true underly-
 792 ing signal. We hope that by applying machine learning we can identify the
 793 signal from the noise. This requires us to have a language for quantify-
 794 ing what “noise” means. We often would also like to have predictors that
 795 allow us to express some sort of uncertainty, e.g., to quantify the confi-
 796 dence we have about the value of the prediction at a particular test data

797 point. Quantification of uncertainty is the realm of *probability theory* and
 798 is covered in Chapter 6.

799 To train machine learning models, we typically find parameters that
 800 maximize some performance measure. Many optimization techniques re-
 801 quire the concept of a gradient, which tells us the direction in which to
 802 search for a solution. Chapter 5 is about *vector calculus* and details the con-
 803 cept of gradients, which we subsequently use in Chapter 7, where we talk
 804 about *optimization* to find maxima/minima of functions.

805 *Part II is about Machine Learning*

806 The second part of the book introduces *four pillars of machine learning* as
 807 shown in Figure 1.1.

808 We illustrate how the mathematical concepts introduced in the first part
 809 of the book are the foundation for each pillar. Broadly speaking, the chap-
 810 ters are ordered by difficulty (in ascending order).

811 In Chapter 8, we restate the three components of machine learning
 812 (data, models and parameter estimation) in a mathematical fashion. In
 813 addition, we provide some guidelines for building experimental set-ups
 814 that guard against overly optimistic evaluations of machine learning sys-
 815 tems. Recall that the goal is to build a predictor that performs well on
 816 unseen data.

817 In Chapter 9, we will have a close look at *linear regression*, where our
 818 objective is to find functions that map inputs $\mathbf{x} \in \mathbb{R}^D$ to corresponding
 819 observed function values $y \in \mathbb{R}$, which we can interpret as the labels of
 820 their respective inputs. We will discuss classical model fitting (parameter
 821 estimation) via maximum likelihood and maximum a posteriori estimation
 822 as well as Bayesian linear regression where we integrate the parameters
 823 out instead of optimizing them.

824 Chapter 10 focuses on *dimensionality reduction*, the second pillar in Fig-
 825 ure 1.1, using principal component analysis. The key objective of dimen-
 826 sionality reduction is to find a compact, lower-dimensional representation
 827 of high-dimensional data $\mathbf{x} \in \mathbb{R}^D$, which is often easier to analyze than
 828 the original data. Unlike regression, dimensionality reduction is only con-
 829 cerned about modeling the data – there are no labels associated with a
 830 data point \mathbf{x} .

831 In Chapter 11, we will move to our third pillar: *density estimation*. The
 832 objective of density estimation is to find a probability distribution that de-
 833 scribes a given dataset. We will focus on Gaussian mixture models for this
 834 purpose, and we will discuss an iterative scheme to find the parameters of
 835 this model. As in dimensionality reduction, there are no labels associated
 836 with the data points $\mathbf{x} \in \mathbb{R}^D$. However, we do not seek a low-dimensional
 837 representation of the data. Instead, we are interested in a density model
 838 that describes the data.

839 Chapter 12 concludes the book with an in-depth discussion of the fourth
 840 pillar: *classification*. We will discuss classification in the context of support

841 vector machines. Similar to regression (Chapter 9) we have inputs x and
842 corresponding labels y . However, unlike regression where the labels were
843 real-valued, the labels in classification are integers, which requires special
844 care.

845 **1.3 Exercises and Feedback**

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

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

853 <https://mml-book.com>

854 where tutorials, errata and additional materials can be found. Mistakes
855 can be reported and feedback provided using the URL above.

2

773

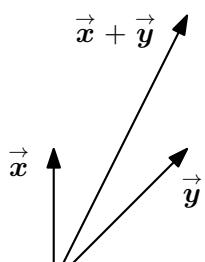
Linear Algebra

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

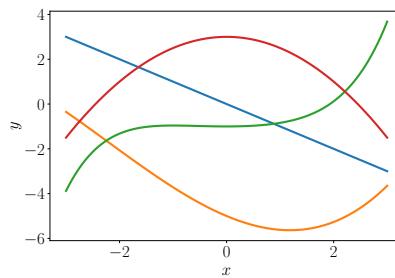
777 Linear algebra is the study of vectors and certain rules to manipulate
 778 vectors. The vectors many of us know from school are called “geometric
 779 vectors”, which are usually denoted by having a small arrow above the
 780 letter, e.g., \vec{x} and \vec{y} . In this book, we discuss more general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

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

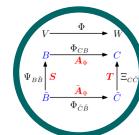
- 786 1 Geometric vectors. This example of a vector may be familiar from school.
 787 Geometric vectors are directed segments, which can be drawn, see Figure
 788 2.1(a). Two geometric vectors \vec{x} , \vec{y} can be added, such that $\vec{x} + \vec{y} = \vec{z}$ is another geometric vector. Furthermore, multiplication by
 789 a scalar λ \vec{x} , $\lambda \in \mathbb{R}$ is also a geometric vector. In fact, it is the original
 790 vector scaled by λ . Therefore, geometric vectors are instances of the
 791 vector concepts introduced above.
- 793 2 Polynomials are also vectors, see Figure 2.1(b): Two polynomials can
 794 be added together, which results in another polynomial; and they can
 795 be multiplied by a scalar $\lambda \in \mathbb{R}$, and the result is a polynomial as well.
 796 Therefore, polynomials are (rather unusual) instances of vectors. Note



(a) Geometric vectors.



(b) Polynomials.



algebra

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

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.

- 800 3 Audio signals are vectors. Audio signals are represented as a series of
801 numbers. We can add audio signals together, and their sum is a new
802 audio signal. If we scale an audio signal, we also obtain an audio signal.
803 Therefore, audio signals are a type of vector, too.
- 804 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,

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in \mathbb{R}^3 \quad (2.1)$$

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

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.

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

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.

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,

matrix

Pavel Grinfeld’s
series on linear
algebra:
<http://tinyurl.com/nahclwm>
Gilbert Strang’s
course on linear
algebra:
<http://tinyurl.com/29p5q8j>

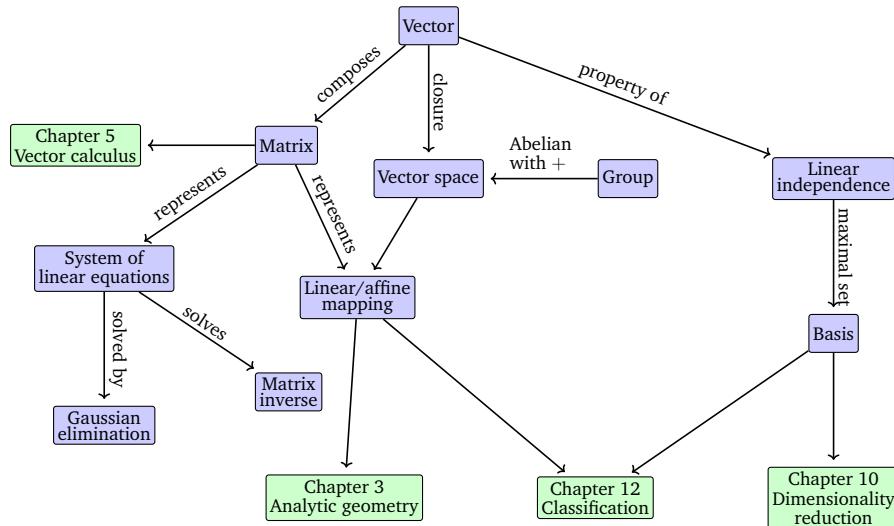


Figure 2.2 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

- ⁸³³ we will use projections (to be introduced in Section 3.7) for dimensionality reduction with Principal Component Analysis (PCA). In Chapter 9, we
⁸³⁴ will discuss linear regression where linear algebra plays a central role for
⁸³⁵ solving least-squares problems.

2.1 Systems of Linear Equations

- ⁸³⁷ Systems of linear equations play a central part of linear algebra. Many
⁸³⁸ problems can be formulated as systems of linear equations, and linear
⁸³⁹ 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

$$a_{11}x_1 + \dots + a_{in}x_n \quad (2.2)$$

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:

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{m1}x_1 + \dots + a_{mn}x_n &= b_m \end{aligned} \quad (2.3)$$

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

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

Example 2.2

The system of linear equations

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

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

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

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

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

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

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

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

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

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

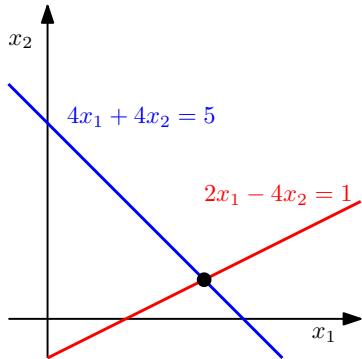


Figure 2.3 The solution space of a system of two linear equations with two variables can be geometrically interpreted as the intersection of two lines. Every linear equation represents a line.

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:

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

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

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* \mathbf{A} is an $m \cdot n$ -tuple of elements a_{ij} , $i = 1, \dots, m$, $j = 1, \dots, n$, which is ordered

matrix

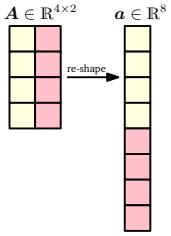
according to a rectangular scheme consisting of m rows and n columns:

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

We sometimes write $\mathbf{A} = ((a_{ij}))$ to indicate that the matrix \mathbf{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*.

$\mathbb{R}^{m \times n}$ is the set of all real-valued (m, n) -matrices. $\mathbf{A} \in \mathbb{R}^{m \times n}$ can be equivalently represented as $\mathbf{a} \in \mathbb{R}^{mn}$ by stacking all n columns of the matrix into a long vector, see Figure 2.4.

rows
columns
row vector
column vector
Figure 2.4 A matrix
 \mathbf{A} can be
represented as a
long vector \mathbf{a} by
stacking its
columns.
873



Note the size of the matrices.

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

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

2.2.1 Matrix Addition and Multiplication

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

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

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

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

This means, to compute element c_{ij} we multiply the elements of the i th row of \mathbf{A} with the j th column of \mathbf{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 \mathbf{A} can be multiplied with a $k \times m$ -matrix \mathbf{B} , but only from the left side:

$$\underbrace{\mathbf{A}}_{n \times k} \underbrace{\mathbf{B}}_{k \times m} = \underbrace{\mathbf{C}}_{n \times m} \quad (2.13)$$

The product \mathbf{BA} is not defined if $m \neq n$ since the neighboring dimensions do not match. \diamond

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 \mathbf{A} , \mathbf{B} was chosen appropriately). This kind of element-wise multiplication often appears in programming languages when we multiply (multi-dimensional) arrays with each other. \diamond

Example 2.3

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

$$\mathbf{AB} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 2 & 5 \end{bmatrix} \in \mathbb{R}^{2 \times 2}, \quad (2.14)$$

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

From this example, we can already see that matrix multiplication is not commutative, i.e., $\mathbf{AB} \neq \mathbf{BA}$, see also Figure 2.5 for an illustration.

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

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (2.16)$$

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

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:

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

- Distributivity:

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

$$\mathbf{A}(\mathbf{C} + \mathbf{D}) = \mathbf{AC} + \mathbf{AD} \quad (2.18b)$$

- Neutral element:

$$\forall \mathbf{A} \in \mathbb{R}^{m \times n} : \mathbf{I}_m \mathbf{A} = \mathbf{A} \mathbf{I}_n = \mathbf{A} \quad (2.19)$$

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

Figure 2.5 Even if both matrix multiplications \mathbf{AB} and \mathbf{BA} are defined, the dimensions of the results can be different.

identity matrix

892

2.2.2 Inverse and Transpose

A square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ possesses the same number of columns and rows. 893
894
895

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

inverse 896 Unfortunately, not every matrix \mathbf{A} possesses an inverse \mathbf{A}^{-1} . If this inverse does exist, \mathbf{A} is called *regular/invertible/non-singular*, otherwise *singular/non-invertible*. 897
898

regular 897 *Remark* (Existence of the Inverse of a 2×2 -Matrix). Consider a matrix

$$\mathbf{A} := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in \mathbb{R}^{2 \times 2}. \quad (2.20)$$

If we multiply \mathbf{A} with

$$\mathbf{B} := \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \quad (2.21)$$

we obtain

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

so that

$$\mathbf{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \quad (2.23)$$

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. ◇ 899
900
901

Example 2.4 (Inverse Matrix)

The matrices

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

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

transpose 902
903 **Definition 2.4** (Transpose). For $\mathbf{A} \in \mathbb{R}^{m \times n}$ the matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ with $b_{ij} = a_{ji}$ is called the *transpose* of \mathbf{A} . We write $\mathbf{B} = \mathbf{A}^\top$.

The main diagonal (sometimes called “principal diagonal”⁸⁹⁵ “primary diagonal”⁹⁰⁶) of a matrix \mathbf{A} is the collection of entries A_{ij} where $i = j$. 904 For a square matrix \mathbf{A}^\top is the matrix we obtain when we “mirror” \mathbf{A} on its main diagonal. In general, \mathbf{A}^\top can be obtained by writing the columns of \mathbf{A} as the rows of \mathbf{A}^\top .

Some important properties of inverses and transposes are:

$$\mathbf{AA}^{-1} = \mathbf{I} = \mathbf{A}^{-1}\mathbf{A} \quad (2.25)$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \quad (2.26)$$

$$(\mathbf{A} + \mathbf{B})^{-1} \neq \mathbf{A}^{-1} + \mathbf{B}^{-1} \quad (2.27)$$

$$(\mathbf{A}^\top)^\top = \mathbf{A} \quad (2.28)$$

$$(\mathbf{A} + \mathbf{B})^\top = \mathbf{A}^\top + \mathbf{B}^\top \quad (2.29)$$

$$(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top \quad (2.30)$$

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

A matrix \mathbf{A} is *symmetric* if $\mathbf{A} = \mathbf{A}^\top$. 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.

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

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

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}. \quad (2.31)$$

◇

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 $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\lambda \in \mathbb{R}$. Then $\lambda \mathbf{A} = \mathbf{K}$, $K_{ij} = \lambda a_{ij}$. Practically, λ scales each element of \mathbf{A} . For $\lambda, \psi \in \mathbb{R}$ it holds:

- Distributivity:

$$(\lambda + \psi)\mathbf{C} = \lambda\mathbf{C} + \psi\mathbf{C}, \quad \mathbf{C} \in \mathbb{R}^{m \times n}$$

$$\lambda(\mathbf{B} + \mathbf{C}) = \lambda\mathbf{B} + \lambda\mathbf{C}, \quad \mathbf{B}, \mathbf{C} \in \mathbb{R}^{m \times n}$$

- Associativity:

$$(\lambda\psi)\mathbf{C} = \lambda(\psi\mathbf{C}), \quad \mathbf{C} \in \mathbb{R}^{m \times n}$$

$$\lambda(\mathbf{BC}) = (\lambda\mathbf{B})\mathbf{C} = \mathbf{B}(\lambda\mathbf{C}) = (\mathbf{BC})\lambda, \quad \mathbf{B} \in \mathbb{R}^{m \times n}, \mathbf{C} \in \mathbb{R}^{n \times k}.$$

Note that this allows us to move scalar values around.

- $(\lambda\mathbf{C})^\top = \mathbf{C}^\top\lambda^\top = \mathbf{C}^\top\lambda = \lambda\mathbf{C}^\top$ since $\lambda = \lambda^\top$ for all $\lambda \in \mathbb{R}$.

Example 2.5 (Distributivity)

If we define

$$\mathbf{C} := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (2.32)$$

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

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

$$= \begin{bmatrix} \lambda & 2\lambda \\ 3\lambda & 4\lambda \end{bmatrix} + \begin{bmatrix} \psi & 2\psi \\ 3\psi & 4\psi \end{bmatrix} = \lambda\mathbf{C} + \psi\mathbf{C} \quad (2.33b)$$

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 $\mathbf{Ax} = \mathbf{b}$, see (2.3), and the product \mathbf{Ax} is a (linear) combination of the columns of \mathbf{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, $i = 1, \dots, m$, $j = 1, \dots, n$. Thus far, we saw that matrices can be used as a compact way of formulating systems of linear equations so that we can write $\mathbf{Ax} = \mathbf{b}$, 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, \dots, x_4 , such that $\sum_{i=1}^4 x_i c_i = b$, where we define c_i to be the i th column of the matrix and 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

$$b = \begin{bmatrix} 42 \\ 8 \end{bmatrix} = 42 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 8 \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (2.38)$$

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)

$$\begin{bmatrix} 8 \\ 2 \end{bmatrix} = 8 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.39)$$

so that $\mathbf{0} = 8c_1 + 2c_2 - 1c_3 + 0c_4$ 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.,

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \left(\lambda_1 \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix} \right) = \lambda_1(8c_1 + 2c_2 - c_3) = \mathbf{0}. \quad (2.40)$$

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

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \left(\lambda_2 \begin{bmatrix} -4 \\ 12 \\ 0 \\ -1 \end{bmatrix} \right) = \lambda_2(-4c_1 + 12c_2 - c_4) = \mathbf{0} \quad (2.41)$$

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

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

⁹³⁹ Remark. The general approach we followed consisted of the following three steps:

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

particular solution
special solution

general solution

943 3 Combine the solutions from 1. and 2. to the general solution.

944 Neither the general nor the particular solution is unique. \diamond

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

956 2.3.2 Elementary Transformations

elementary 957 Key to solving a system of linear equations are *elementary transformations*
 transformations 958 that keep the solution set the same, but that transform the equation system
 959 into a simpler form:

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

Example 2.6

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

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

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

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

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

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ -2 & 4 & -2 & -1 & 4 & -3 \\ 1 & -2 & 0 & -3 & 4 & a \end{array} \right] \begin{matrix} \\ -4R_1 \\ +2R_1 \\ -R_1 \end{matrix}$$

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

$$\begin{aligned} & \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & -1 & -2 & 3 & a \end{array} \right] \begin{matrix} \\ \\ \\ -R_2 - R_3 \end{matrix} \\ \rightsquigarrow & \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & 0 & 0 & 0 & a+1 \end{array} \right] \begin{matrix} \\ \\ \cdot(-1) \\ \cdot(-\frac{1}{3}) \end{matrix} \\ \rightsquigarrow & \left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 3 & -2 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & a+1 \end{array} \right] \end{aligned}$$

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

$$\begin{aligned} x_1 - 2x_2 + x_3 - x_4 + x_5 &= 0 \\ x_3 - x_4 + 3x_5 &= -2 \\ x_4 - 2x_5 &= 1 \\ 0 &= a+1 \end{aligned} \quad . \quad (2.44)$$

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

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

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

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

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

row-echelon form (REF)

particular solution

general solution

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. ◇

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.

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. ◇

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 $\mathbf{b} = \sum_{i=1}^P \lambda_i \mathbf{p}_i$, where \mathbf{p}_i , $i = 1, \dots, P$, 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

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

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 $\mathbf{x} = [2, 0, -1, 1, 0]^\top$. ◇

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.

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

Gaussian
elimination

Example 2.7 (Reduced Row Echelon Form)

Verify that the following matrix is in reduced row echelon form (the pivots are in **bold**):

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & 3 & 0 & 0 & 3 \\ 0 & 0 & \mathbf{1} & 0 & 9 \\ 0 & 0 & 0 & \mathbf{1} & -4 \end{bmatrix} \quad (2.48)$$

The key idea for finding the solutions of $\mathbf{A}\mathbf{x} = \mathbf{0}$ 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 $\mathbf{A}\mathbf{x} = \mathbf{0}, \mathbf{x} \in \mathbb{R}^5$ are given by

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

2.3.3 The Minus-1 Trick

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

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

$$\mathbf{A} = \begin{bmatrix} 0 & \cdots & 0 & \mathbf{1} & * & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * \\ \vdots & & \vdots & 0 & 0 & \cdots & 0 & \mathbf{1} & * & \cdots & * & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & 0 & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & 0 & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \mathbf{1} & * & \cdots & * \end{bmatrix}, \quad (2.50)$$

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, \dots, j_k with the pivots (marked in **bold**) are the standard unit vectors $e_1, \dots, e_k \in \mathbb{R}^k$. We extend this matrix to an $n \times n$ -matrix $\tilde{\mathbf{A}}$ by adding $n - k$ rows of the form

$$[0 \ \cdots \ 0 \ -1 \ 0 \ \cdots \ 0] \quad (2.51)$$

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

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & -4 \end{bmatrix}. \quad (2.52)$$

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

$$\tilde{\mathbf{A}} = \begin{bmatrix} 1 & 3 & 0 & 0 & 3 \\ \color{blue}{0} & \color{blue}{-1} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & -4 \\ \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{-1} \end{bmatrix} \quad (2.53)$$

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

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

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

1013

Calculating the Inverse

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

$$[\mathbf{A} | \mathbf{I}_n] \rightsquigarrow \dots \rightsquigarrow [\mathbf{I}_n | \mathbf{A}^{-1}]. \quad (2.55)$$

1014

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.

1015

1016

1017

Example 2.9 (Calculating an Inverse Matrix by Gaussian Elimination)

To determine the inverse of

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (2.56)$$

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:

$$\mathbf{A}^{-1} = \begin{bmatrix} -1 & 2 & -2 & 2 \\ 1 & -1 & 2 & -2 \\ 1 & -1 & 1 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}. \quad (2.57)$$

1018 2.3.4 Algorithms for Solving a System of Linear Equations

1019 In the following, we briefly discuss approaches to solving a system of lin-
 1020 ear equations of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$.

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

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

Moore-Penrose
pseudo-inverse 1021 and use the *Moore-Penrose pseudo-inverse* $(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top$ to determine the
 1022 solution (2.58) that solves $\mathbf{A}\mathbf{x} = \mathbf{b}$, which also corresponds to the mini-
 1023 mum norm least-squares solution. A disadvantage of this approach is that
 1024 it requires many computations for the matrix-matrix product and comput-
 1025 ing the inverse of $\mathbf{A}^\top \mathbf{A}$. Moreover, for reasons of numerical precision it
 1026 is generally not recommended to compute the inverse or pseudo-inverse.
 1027 In the following, we therefore briefly discuss alternative approaches to
 1028 solving systems of linear equations.

1029 Gaussian elimination plays an important role when computing deter-
 1030 minants (Section 4.1), checking whether a set of vectors is linearly inde-
 1031 pendent (Section 2.5), computing the inverse of a matrix (Section 2.2.2),
 1032 computing the rank of a matrix (Section 2.6.2) and a basis of a vector
 1033 space (Section 2.6.1). We will discuss all these topics later on. Gaussian
 1034 elimination is an intuitive and constructive way to solve a system of linear
 1035 equations with thousands of variables. However, for systems with millions
 1036 of variables, it is impractical as the required number of arithmetic opera-
 1037 tions scales cubically in the number of simultaneous equations.

1038 In practice, systems of many linear equations are solved indirectly, by
 1039 either stationary iterative methods, such as the Richardson method, the
 1040 Jacobi method, the Gauß-Seidel method, or the successive over-relaxation
 1041 method, or Krylov subspace methods, such as conjugate gradients, gener-
 1042 alized minimal residual, or biconjugate gradients.

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

is to set up an iteration of the form

$$\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{x}^{(k)} \quad (2.59)$$

that reduces the residual error $\|\mathbf{x}^{(k+1)} - \mathbf{x}_*\|$ in every iteration and finally converges to \mathbf{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 $\otimes : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}$ defined on \mathcal{G} .

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

- 1 *Closure* of \mathcal{G} under \otimes : $\forall x, y \in \mathcal{G} : x \otimes y \in \mathcal{G}$
- 2 *Associativity*: $\forall x, y, z \in \mathcal{G} : (x \otimes y) \otimes z = x \otimes (y \otimes z)$
- 3 *Neutral element*: $\exists e \in \mathcal{G} \forall x \in \mathcal{G} : x \otimes e = x$ and $e \otimes x = x$
- 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 .

group
Closure
Associativity:
Neutral element:
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).

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.
- $(\mathbb{R} \setminus \{0\}, \cdot)$ is Abelian.
- $(\mathbb{R}^n, +), (\mathbb{Z}^n, +), n \in \mathbb{N}$ are Abelian if + is defined componentwise, i.e.,

$$(x_1, \dots, x_n) + (y_1, \dots, y_n) = (x_1 + y_1, \dots, x_n + y_n). \quad (2.60)$$

Then, $(x_1, \dots, x_n)^{-1} := (-x_1, \dots, -x_n)$ is the inverse element and $e = (0, \dots, 0)$ 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 I_n is the neutral element with respect to matrix multiplication “.” in $(\mathbb{R}^{n \times n}, \cdot)$.
 - Inverse element: If the inverse exists then A^{-1} is the inverse element of $A \in \mathbb{R}^{n \times n}$.

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

1071 1072

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

1073 1074
general linear group⁵
1076

Definition 2.7 (General Linear Group). The set of regular (invertible) matrices $A \in \mathbb{R}^{n \times n}$ 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.

1077

2.4.2 Vector Spaces

1078
1079
1080
1081
1082

When we discussed groups, we looked at sets \mathcal{G} and inner operations on \mathcal{G} , i.e., mappings $\mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}$ 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 $x \in \mathcal{V}$ 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

$$+ : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V} \quad (2.61)$$

$$\cdot : \mathbb{R} \times \mathcal{V} \rightarrow \mathcal{V} \quad (2.62)$$

1083 where

1084 1 $(\mathcal{V}, +)$ is an Abelian group

1085 2 Distributivity:

1086 1 $\forall \lambda \in \mathbb{R}, \mathbf{x}, \mathbf{y} \in \mathcal{V} : \lambda \cdot (\mathbf{x} + \mathbf{y}) = \lambda \cdot \mathbf{x} + \lambda \cdot \mathbf{y}$

1087 2 $\forall \lambda, \psi \in \mathbb{R}, \mathbf{x} \in \mathcal{V} : (\lambda + \psi) \cdot \mathbf{x} = \lambda \cdot \mathbf{x} + \psi \cdot \mathbf{x}$

1088 3 Associativity (outer operation): $\forall \lambda, \psi \in \mathbb{R}, \mathbf{x} \in \mathcal{V} : \lambda \cdot (\psi \cdot \mathbf{x}) = (\lambda \psi) \cdot \mathbf{x}$

1089 4 Neutral element with respect to the outer operation: $\forall \mathbf{x} \in \mathcal{V} : 1 \cdot \mathbf{x} = \mathbf{x}$

1090 The elements $\mathbf{x} \in \mathcal{V}$ are called *vectors*. The neutral element of $(\mathcal{V}, +)$ is
1091 the zero vector $\mathbf{0} = [0, \dots, 0]^\top$, and the inner operation $+$ is called *vector*
1092 *addition*. The elements $\lambda \in \mathbb{R}$ are called *scalars* and the outer operation
1093 \cdot is a *multiplication by scalars*. Note that a scalar product is something
1094 different, and we will get to this in Section 3.2.

vectors

vector addition

scalars

multiplication by
scalars

1095 Remark. A “vector multiplication” \mathbf{ab} , $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, is not defined. Theoretically, we could define an element-wise multiplication, such that $\mathbf{c} = \mathbf{ab}$ 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: $\mathbf{ab}^\top \in \mathbb{R}^{n \times n}$ (outer product), $\mathbf{a}^\top \mathbf{b} \in \mathbb{R}$ (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: $\mathbf{x} + \mathbf{y} = (x_1, \dots, x_n) + (y_1, \dots, y_n) = (x_1 + y_1, \dots, x_n + y_n)$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$
 - Multiplication by scalars: $\lambda \mathbf{x} = \lambda(x_1, \dots, x_n) = (\lambda x_1, \dots, \lambda x_n)$ for all $\lambda \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$
- $\mathcal{V} = \mathbb{R}^{m \times n}, m, n \in \mathbb{N}$ is a vector space with
 - Addition: $\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$ is defined elementwise for all $\mathbf{A}, \mathbf{B} \in \mathcal{V}$
 - Multiplication by scalars: $\lambda \mathbf{A} = \begin{bmatrix} \lambda a_{11} & \cdots & \lambda a_{1n} \\ \vdots & & \vdots \\ \lambda a_{m1} & \cdots & \lambda a_{mn} \end{bmatrix}$ 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.

1104 1105 1106 1107 *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 $\mathbf{x} \in V$ for vectors in \mathcal{V} to simplify notation. \diamond

column vectors 1108 *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*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \quad (2.63)$$

row vectors 1108 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 \mathbf{x} to denote a column vector, and a row vector is denoted by \mathbf{x}^\top , the *transpose* of \mathbf{x} . \diamond

1112 2.4.3 Vector Subspaces

1113 1114 1115 1116 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”.

vector subspace 1117 1118 1119 1120 1121 **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 .

1122 1123 1124 1125 1126 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 $\mathbf{x} \in \mathcal{V}$, and in particular for all $\mathbf{x} \in \mathcal{U} \subseteq \mathcal{V}$. 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

- 1127 1 $\mathcal{U} \neq \emptyset$, in particular: $\mathbf{0} \in \mathcal{U}$
- 1128 2 Closure of U :
 - 1129 1 With respect to the outer operation: $\forall \lambda \in \mathbb{R} \forall \mathbf{x} \in \mathcal{U} : \lambda \mathbf{x} \in \mathcal{U}$.
 - 1130 2 With respect to the inner operation: $\forall \mathbf{x}, \mathbf{y} \in \mathcal{U} : \mathbf{x} + \mathbf{y} \in \mathcal{U}$.

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 2.6 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 $A\mathbf{x} = \mathbf{0}$ with n unknowns $\mathbf{x} = [x_1, \dots, x_n]^\top$ is a subspace of \mathbb{R}^n .
- The solution of an inhomogeneous equation system $A\mathbf{x} = \mathbf{b}$, $\mathbf{b} \neq \mathbf{0}$ is not a subspace of \mathbb{R}^n .
- The intersection of arbitrarily many subspaces is a subspace itself.

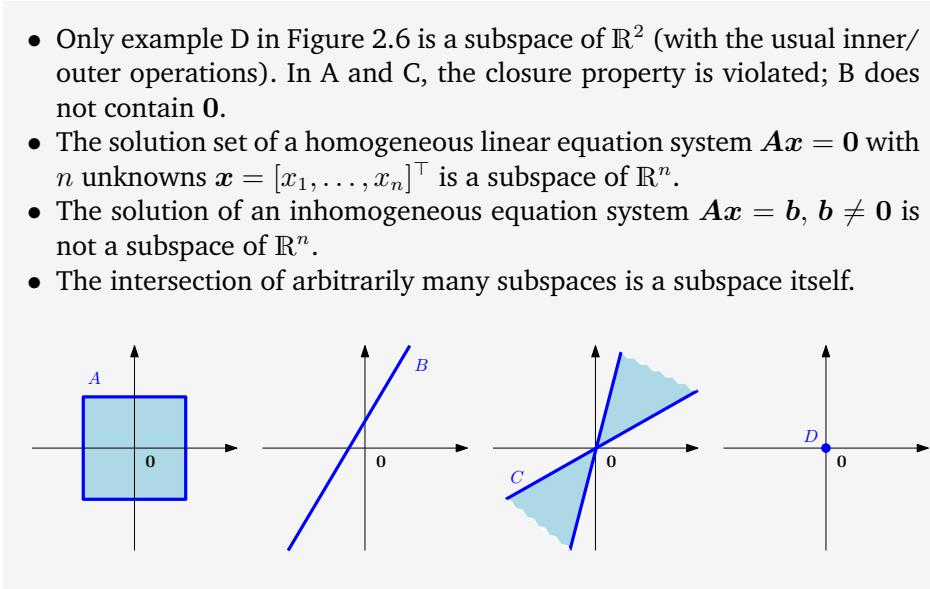


Figure 2.6 Not all subsets of \mathbb{R}^2 are subspaces. In A and C, the closure property is violated; B does not contain $\mathbf{0}$. Only D is a subspace.

1131 *Remark.* Every subspace $U \subseteq (\mathbb{R}^n, +, \cdot)$ is the solution space of a homogeneous linear equation system $A\mathbf{x} = \mathbf{0}$. 1132 ◇

1133 2.5 Linear Independence

1134 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 1135 the vector space). In particular, we can add vectors together and multiply them with scalars. The closure property guarantees that we end up 1136 with another vector in the same vector space. Let us formalize this: 1137

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

$$\mathbf{v} = \lambda_1 \mathbf{x}_1 + \dots + \lambda_k \mathbf{x}_k = \sum_{i=1}^k \lambda_i \mathbf{x}_i \in V \quad (2.64)$$

1138 with $\lambda_1, \dots, \lambda_k \in \mathbb{R}$ is a *linear combination* of the vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$.

linear combination

1139 The $\mathbf{0}$ -vector can always be written as the linear combination of k vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$ because $\mathbf{0} = \sum_{i=1}^k 0 \mathbf{x}_i$ is always true. In the following, 1140 we are interested in non-trivial linear combinations of a set of vectors to 1141 represent $\mathbf{0}$, i.e., linear combinations of vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$ where not all 1142 coefficients λ_i in (2.64) are 0. 1143

1144 **Definition 2.11** (Linear (In)dependence). Let us consider a vector space V with $k \in \mathbb{N}$ and $\mathbf{x}_1, \dots, \mathbf{x}_k \in V$. If there is a non-trivial linear combination, such that $\mathbf{0} = \sum_{i=1}^k \lambda_i \mathbf{x}_i$ with at least one $\lambda_i \neq 0$, the vectors

1148 $\mathbf{x}_1, \dots, \mathbf{x}_k$ are *linearly dependent*. If only the trivial solution exists, i.e.,
1149 $\lambda_1 = \dots = \lambda_k = 0$ the vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$ are *linearly independent*.

linearly dependent

linearly independent

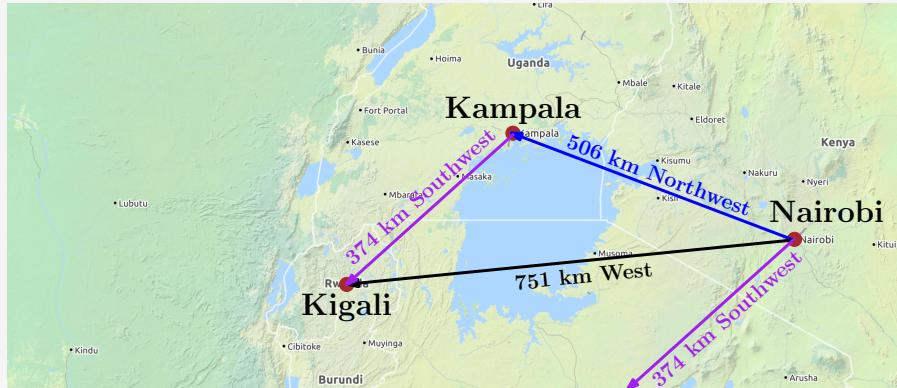
1150 1151 1152 1153 1154 Linear independence is one of the most important concepts in linear algebra. Intuitively, a set of linearly independent vectors are vectors that have no redundancy, i.e., if we remove any of those vectors from the set, we will lose something. Throughout the next sections, we 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 km Northwest to Kampala (Uganda) and then 374 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 km West of here.” Although this last statement is true, it is not necessary to find Kigali given the previous information (see Figure 2.7 for an illustration).

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

Figure 2.7
Geographic example
(with crude
approximations to
cardinal directions)
of linearly
dependent vectors
in a
two-dimensional
space (plane).



In this example, the “506 km Northwest” vector (blue) and the “374 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 km West” vector (black) is a linear combination of the other two vectors, and it makes the set of vectors linearly dependent.

1155 1156 *Remark.* The following properties are useful to find out whether vectors are linearly independent.

- 1157 1158 • k vectors are either linearly dependent or linearly independent. There is no third option.

- If at least one of the vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$ is $\mathbf{0}$ then they are linearly dependent. The same holds if two vectors are identical.
- The vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_k : \mathbf{x}_i \neq \mathbf{0}, i = 1, \dots, k\}$, $k \geq 2$, 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., $\mathbf{x}_i = \lambda \mathbf{x}_j$, $\lambda \in \mathbb{R}$ then the set $\{\mathbf{x}_1, \dots, \mathbf{x}_k : \mathbf{x}_i \neq \mathbf{0}, i = 1, \dots, k\}$ is linearly dependent.
- A practical way of checking whether vectors $\mathbf{x}_1, \dots, \mathbf{x}_k \in V$ are linearly independent is to use Gaussian elimination: Write all vectors as columns of a matrix \mathbf{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

$$\begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad (2.65)$$

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

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

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

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

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

$$\left[\begin{array}{ccc} 1 & 1 & -1 \\ 2 & 1 & -2 \\ -3 & 0 & 1 \\ 4 & 2 & 1 \end{array} \right] \rightsquigarrow \dots \rightsquigarrow \left[\begin{array}{ccc} 1 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array} \right] \quad (2.68)$$

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 $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are linearly independent.

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

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

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

$$\mathbf{x}_j = \mathbf{B} \boldsymbol{\lambda}_j, \quad \boldsymbol{\lambda}_j = \begin{bmatrix} \lambda_{1j} \\ \vdots \\ \lambda_{kj} \end{bmatrix}, \quad j = 1, \dots, m, \quad (2.70)$$

¹¹⁷⁹ in a more compact form.

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

$$\sum_{j=1}^m \psi_j \mathbf{x}_j = \sum_{j=1}^m \psi_j \mathbf{B} \boldsymbol{\lambda}_j = \mathbf{B} \sum_{j=1}^m \psi_j \boldsymbol{\lambda}_j. \quad (2.71)$$

¹¹⁸⁰ This means that $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ are linearly independent if and only if the
¹¹⁸¹ column vectors $\{\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_m\}$ are linearly independent.

◇

¹¹⁸³ *Remark.* In a vector space V , m linear combinations of k vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$
¹¹⁸⁴ are linearly dependent if $m > k$.

◇

Example 2.15

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

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{b}_1 - 2\mathbf{b}_2 + \mathbf{b}_3 - \mathbf{b}_4 \\ \mathbf{x}_2 &= -4\mathbf{b}_1 - 2\mathbf{b}_2 + 4\mathbf{b}_4 \\ \mathbf{x}_3 &= 2\mathbf{b}_1 + 3\mathbf{b}_2 - \mathbf{b}_3 - 3\mathbf{b}_4 \\ \mathbf{x}_4 &= 17\mathbf{b}_1 - 10\mathbf{b}_2 + 11\mathbf{b}_3 + \mathbf{b}_4\end{aligned}\quad (2.72)$$

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

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

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

$$\mathbf{A} = \begin{bmatrix} 1 & -4 & 2 & 17 \\ -2 & -2 & 3 & -10 \\ 1 & 0 & -1 & 11 \\ -1 & 4 & -3 & 1 \end{bmatrix} \quad (2.74)$$

is given as

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

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

1185

2.6 Basis and Rank

1186 In a vector space V , we are particularly interested in sets of vectors A that
 1187 possess the property that any vector $\mathbf{v} \in V$ can be obtained by a linear
 1188 combination of vectors in A . These vectors are special vectors, and in the
 1189 following, we will characterize them.

1190

2.6.1 Generating Set and Basis

1191 **Definition 2.12** (Generating Set and Span). Consider a vector space $V =$
 1192 $(\mathcal{V}, +, \cdot)$ and set of vectors $\mathcal{A} = \{\mathbf{x}_1, \dots, \mathbf{x}_k\} \subseteq \mathcal{V}$. If every vector $\mathbf{v} \in$

generating set
span 1193 \mathcal{V} can be expressed as a linear combination of $\mathbf{x}_1, \dots, \mathbf{x}_k$, \mathcal{A} is called a
1194 *generating set* of V . The set of all linear combinations of vectors in \mathcal{A} is
1195 called the *span* of \mathcal{A} . If \mathcal{A} spans the vector space V , we write $V = \text{span}[\mathcal{A}]$
1196 or $V = \text{span}[\mathbf{x}_1, \dots, \mathbf{x}_k]$.

1197 Generating sets are sets of vectors that span vector (sub)spaces, i.e.,
1198 every vector can be represented as a linear combination of the vectors
1199 in the generating set. Now, we will be more specific and characterize the
1200 smallest generating set that spans a vector (sub)space.

minimal 1201 **Definition 2.13 (Basis).** Consider a vector space $V = (\mathcal{V}, +, \cdot)$ and $\mathcal{A} \subseteq$
1202 \mathcal{V} . A generating set \mathcal{A} of V is called *minimal* if there exists no smaller set
1203 $\tilde{\mathcal{A}} \subseteq \mathcal{A} \subseteq \mathcal{V}$ that spans V . Every linearly independent generating set of V
1204 is minimal and is called a *basis* of V .

A basis is a minimal
generating set and¹²⁰⁵ maximal linearly
independent set of
vectors. 1204 Let $V = (\mathcal{V}, +, \cdot)$ be a vector space and $\mathcal{B} \subseteq \mathcal{V}, \mathcal{B} \neq \emptyset$. Then, the
1205 following statements are equivalent:

- \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
1206 other vector to this set will make it linearly dependent.
- Every vector $\mathbf{x} \in V$ is a linear combination of vectors from \mathcal{B} , and every
1207 linear combination is unique, i.e., with

$$\mathbf{x} = \sum_{i=1}^k \lambda_i \mathbf{b}_i = \sum_{i=1}^k \psi_i \mathbf{b}_i \quad (2.76)$$

1211 and $\lambda_i, \psi_i \in \mathbb{R}$, $\mathbf{b}_i \in \mathcal{B}$ it follows that $\lambda_i = \psi_i$, $i = 1, \dots, k$.

Example 2.16

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

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

• Different bases in \mathbb{R}^3 are

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

• The set

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

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} .

1212 *Remark.* Every vector space V possesses a basis \mathcal{B} . The examples above
1213 show that there can be many bases of a vector space V , i.e., there is no
1214 unique basis. However, all bases possess the same number of elements,
1215 the *basis vectors*. \diamond

1216 We only consider finite-dimensional vector spaces V . In this case, the
1217 *dimension* of V is the number of basis vectors, and we write $\dim(V)$. If
1218 $U \subseteq V$ is a subspace of V then $\dim(U) \leq \dim(V)$ and $\dim(U) = \dim(V)$
1219 if and only if $U = V$. Intuitively, the dimension of a vector space can be
1220 thought of as the number of independent directions in this vector space.

1221 *Remark.* A basis of a subspace $U = \text{span}[\mathbf{x}_1, \dots, \mathbf{x}_m] \subseteq \mathbb{R}^n$ can be found
1222 by executing the following steps:

- 1223 1 Write the spanning vectors as columns of a matrix \mathbf{A}
- 1224 2 Determine the row echelon form of \mathbf{A} .
- 1225 3 The spanning vectors associated with the pivot columns are a basis of
1226 U .

1227 \diamond

Example 2.17 (Determining a Basis)

For a vector subspace $U \subseteq \mathbb{R}^5$, spanned by the vectors

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

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

$$\sum_{i=1}^4 \lambda_i \mathbf{x}_i = \mathbf{0}, \quad (2.81)$$

which leads to a homogeneous system of equations with matrix

$$[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4] = \begin{bmatrix} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{bmatrix}. \quad (2.82)$$

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

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

$$\left[\begin{array}{cccc} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{array} \right] \rightsquigarrow \cdots \rightsquigarrow \left[\begin{array}{cccc} 1 & 2 & 3 & -1 \\ 0 & 1 & 2 & -2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right].$$

Since the pivot columns indicate which set of vectors is linearly independent, we see from the row echelon form that $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4$ are linearly independent (because the system of linear equations $\lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 + \lambda_4\mathbf{x}_4 = \mathbf{0}$ can only be solved with $\lambda_1 = \lambda_2 = \lambda_4 = 0$). Therefore, $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4\}$ is a basis of U .

2.6.2 Rank

- rank
- 1228 The number of linearly independent columns of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$
 1229 equals the number of linearly independent rows and is called the *rank*
 1230 of \mathbf{A} and is denoted by $\text{rk}(\mathbf{A})$.
 1231
- 1232 *Remark.* The rank of a matrix has some important properties:
- 1233 • $\text{rk}(\mathbf{A}) = \text{rk}(\mathbf{A}^\top)$, i.e., the column rank equals the row rank.
 1234 • The columns of $\mathbf{A} \in \mathbb{R}^{m \times n}$ span a subspace $U \subseteq \mathbb{R}^m$ with $\dim(U) =$
 1235 $\text{rk}(\mathbf{A})$. Later, we will call this subspace the *image* or *range*. A basis of
 1236 U can be found by applying Gaussian elimination to \mathbf{A} to identify the
 1237 pivot columns.
 1238 • The rows of $\mathbf{A} \in \mathbb{R}^{m \times n}$ span a subspace $W \subseteq \mathbb{R}^n$ with $\dim(W) =$
 1239 $\text{rk}(\mathbf{A})$. A basis of W can be found by applying Gaussian elimination to
 1240 \mathbf{A}^\top .
 1241 • For all $\mathbf{A} \in \mathbb{R}^{n \times n}$ holds: \mathbf{A} is regular (invertible) if and only if $\text{rk}(\mathbf{A}) =$
 1242 n .
 1243 • For all $\mathbf{A} \in \mathbb{R}^{m \times n}$ and all $\mathbf{b} \in \mathbb{R}^m$ it holds that the linear equation
 1244 system $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be solved if and only if $\text{rk}(\mathbf{A}) = \text{rk}(\mathbf{A}|\mathbf{b})$, where
 1245 $\mathbf{A}|\mathbf{b}$ denotes the augmented system.
 1246 • For $\mathbf{A} \in \mathbb{R}^{m \times n}$ the subspace of solutions for $\mathbf{A}\mathbf{x} = \mathbf{0}$ possesses dimen-
 1247 sion $n - \text{rk}(\mathbf{A})$. Later, we will call this subspace the *kernel* or the *null*
 1248 *space*.
 1249 • A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ has *full rank* if its rank equals the largest possible
 1250 rank for a matrix of the same dimensions. This means that the rank of
 1251 a full-rank matrix is the lesser of the number of rows and columns, i.e.,
 1252 $\text{rk}(\mathbf{A}) = \min(m, n)$. A matrix is said to be *rank deficient* if it does not
 1253 have full rank.

◇

Example 2.18 (Rank)

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

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \rightsquigarrow \dots \rightsquigarrow \begin{bmatrix} 1 & 2 & 1 \\ 0 & -1 & 3 \\ 0 & 0 & 0 \end{bmatrix}. \quad (2.83)$$

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

1255

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 $\Phi : V \rightarrow W$ preserves the structure of the vector space if

$$\Phi(\mathbf{x} + \mathbf{y}) = \Phi(\mathbf{x}) + \Phi(\mathbf{y}) \quad (2.84)$$

$$\Phi(\lambda\mathbf{x}) = \lambda\Phi(\mathbf{x}) \quad (2.85)$$

1256 for all $\mathbf{x}, \mathbf{y} \in V$ and $\lambda \in \mathbb{R}$. We can summarize this in the following
1257 definition:

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

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

1258 Before we continue, we will briefly introduce special mappings.

1259 **Definition 2.15** (Injective, Surjective, Bijective). Consider a mapping $\Phi : 1260 \mathcal{V} \rightarrow \mathcal{W}$, where \mathcal{V}, \mathcal{W} can be arbitrary sets. Then Φ is called

- 1261 • *injective* if $\forall \mathbf{x}, \mathbf{y} \in \mathcal{V} : \Phi(\mathbf{x}) = \Phi(\mathbf{y}) \implies \mathbf{x} = \mathbf{y}$.
- 1262 • *surjective* if $\Phi(\mathcal{V}) = \mathcal{W}$.
- 1263 • *bijective* if it is injective and surjective.

linear mapping
vector space
homomorphism
linear
transformation

injective
surjective
bijective

If Φ is injective then it can also be “undone”, i.e., there exists a mapping $\Psi : \mathcal{W} \rightarrow \mathcal{V}$ so that $\Psi \circ \Phi(\mathbf{x}) = \mathbf{x}$. 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*: $\Phi : V \rightarrow W$ linear and bijective

- *Endomorphism*: $\Phi : V \rightarrow V$ linear

- *Automorphism*: $\Phi : V \rightarrow V$ linear and bijective

- We define $\text{id}_V : V \rightarrow V$, $\mathbf{x} \mapsto \mathbf{x}$ as the *identity mapping* in V .

Example 2.19 (Homomorphism)

The mapping $\Phi : \mathbb{R}^2 \rightarrow \mathbb{C}$, $\Phi(\mathbf{x}) = x_1 + ix_2$, is a homomorphism:

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

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

1290



1291

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 $\{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ of an n -dimensional vector space V . In the following, the order of the basis vectors will be important. Therefore, we write

$$B = (\mathbf{b}_1, \dots, \mathbf{b}_n) \quad (2.88)$$

1292 and call this n -tuple an *ordered basis* of V .

ordered basis

1293 *Remark* (Notation). We are at the point where notation gets a bit tricky.
 1294 Therefore, we summarize some parts here. $B = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is an ordered
 1295 basis, $\mathcal{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ is an (unordered) basis, and $B = [\mathbf{b}_1, \dots, \mathbf{b}_n]$ is a
 1296 matrix whose columns are the vectors $\mathbf{b}_1, \dots, \mathbf{b}_n$. ◇

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

$$\mathbf{x} = \alpha_1 \mathbf{b}_1 + \dots + \alpha_n \mathbf{b}_n \quad (2.89)$$

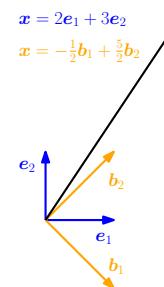
of \mathbf{x} with respect to B . Then $\alpha_1, \dots, \alpha_n$ are the *coordinates* of \mathbf{x} with respect to B , and the vector

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

1297 is the *coordinate vector/coordinate representation* of \mathbf{x} with respect to the
1298 ordered basis B .coordinate vector
coordinate
representation

1299 *Remark.* Intuitively, the basis vectors can be thought of as being equipped
1300 with units (including common units such as “kilograms” or “seconds”).
 1301 Let us have a look at a geometric vector $\mathbf{x} \in \mathbb{R}^2$ with coordinates $[2, 3]^\top$
 1302 with respect to the standard basis $(\mathbf{e}_1, \mathbf{e}_2)$ of \mathbb{R}^2 . This means, we can write
 1303 $\mathbf{x} = 2\mathbf{e}_1 + 3\mathbf{e}_2$. However, we do not have to choose the standard basis to
 1304 represent this vector. If we use the basis vectors $\mathbf{b}_1 = [1, -1]^\top$, $\mathbf{b}_2 = [1, 1]^\top$
 1305 we will obtain the coordinates $\frac{1}{2}[-1, 5]^\top$ to represent the same vector with
 1306 respect to $(\mathbf{b}_1, \mathbf{b}_2)$ (see Figure 2.8). ◇

Figure 2.8
 Different coordinate representations of a vector \mathbf{x} , depending on the choice of basis.



1307 *Remark.* For an n -dimensional vector space V and an ordered basis B
 1308 of V , the mapping $\Phi : \mathbb{R}^n \rightarrow V$, $\Phi(\mathbf{e}_i) = \mathbf{b}_i$, $i = 1, \dots, n$, is linear
 1309 (and because of Theorem 2.16 an isomorphism), where $(\mathbf{e}_1, \dots, \mathbf{e}_n)$ is
 1310 the standard basis of \mathbb{R}^n . ◇

1312 Now we are ready to make an explicit connection between matrices and
1313 linear mappings between finite-dimensional vector spaces.

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

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

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

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

transformation matrix ₁₃₁₄ the *transformation matrix* of Φ (with respect to the ordered bases B of V and C of W). ₁₃₁₅

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

$$\hat{\mathbf{y}} = \mathbf{A}_\Phi \hat{\mathbf{x}}. \quad (2.93)$$

₁₃₁₆ 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 ₁₃₁₇ ordered basis in W . ₁₃₁₈

Example 2.20 (Transformation Matrix)

Consider a homomorphism $\Phi : V \rightarrow W$ and ordered bases $B = (\mathbf{b}_1, \dots, \mathbf{b}_3)$ of V and $C = (\mathbf{c}_1, \dots, \mathbf{c}_4)$ of W . With

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

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

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

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

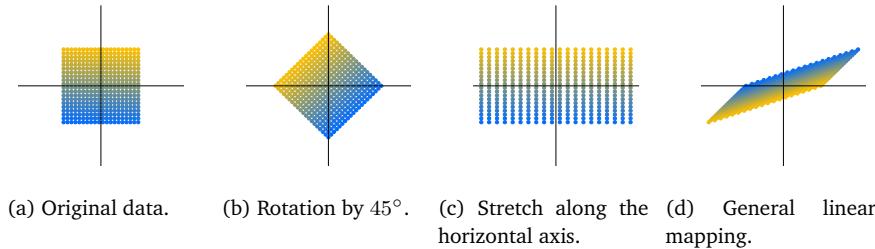


Figure 2.9 Three examples of linear transformations of the vectors shown as dots in (a). (b) Rotation by 45°; (c) Stretching of the horizontal coordinates by 2; (d) Combination of reflection, rotation and stretching.

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

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

Figure 2.9 gives three examples of linear transformations of a set of vectors. Figure 2.9(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 \mathbf{A}_1 in (2.96) to linearly transform each of these vectors, we obtain the rotated square in Figure 2.9(b). If we apply the linear mapping represented by \mathbf{A}_2 , we obtain the rectangle in Figure 2.9(c) where each x_1 -coordinate is stretched by 2. Figure 2.9(d) shows the original square from Figure 2.9(a) when linearly transformed using \mathbf{A}_3 , which is a combination of a reflection, a rotation and a stretch.

1319

2.7.2 Basis Change

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

$$B = (\mathbf{b}_1, \dots, \mathbf{b}_n), \quad \tilde{B} = (\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n) \quad (2.97)$$

of V and two ordered bases

$$C = (\mathbf{c}_1, \dots, \mathbf{c}_m), \quad \tilde{C} = (\tilde{\mathbf{c}}_1, \dots, \tilde{\mathbf{c}}_m) \quad (2.98)$$

1320 of W . Moreover, $\mathbf{A}_\Phi \in \mathbb{R}^{m \times n}$ is the transformation matrix of the linear
 1321 mapping $\Phi : V \rightarrow W$ with respect to the bases B and C , and $\tilde{\mathbf{A}}_\Phi \in \mathbb{R}^{m \times n}$
 1322 is the corresponding transformation mapping with respect to \tilde{B} and \tilde{C} .
 1323 In the following, we will investigate how \mathbf{A} and $\tilde{\mathbf{A}}$ are related, i.e., how/
 1324 whether we can transform \mathbf{A}_Φ into $\tilde{\mathbf{A}}_\Phi$ if we choose to perform a basis
 1325 change from B, C to \tilde{B}, \tilde{C} .

1326 *Remark.* We effectively get different coordinate representations of the
 1327 identity mapping id_V . In the context of Figure 2.8, this would mean to
 1328 map coordinates with respect to (e_1, e_2) onto coordinates with respect to
 1329 (b_1, b_2) without changing the vector x . By changing the basis and corre-
 1330 spondingly the representation of vectors, the transformation matrix with
 1331 respect to this new basis can have a particularly simple form that allows
 1332 for straightforward computation. \diamond

Example 2.22 (Basis Change)

Consider a transformation matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (2.99)$$

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

$$B = \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \quad (2.100)$$

we obtain a diagonal transformation matrix

$$\tilde{\mathbf{A}} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.101)$$

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

1333 In the following, we will look at mappings that transform coordinate
 1334 vectors with respect to one basis into coordinate vectors with respect to
 1335 a different basis. We will state our main result first and then provide an
 1336 explanation.

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

$$B = (b_1, \dots, b_n), \quad \tilde{B} = (\tilde{b}_1, \dots, \tilde{b}_n) \quad (2.102)$$

of V and

$$C = (c_1, \dots, c_m), \quad \tilde{C} = (\tilde{c}_1, \dots, \tilde{c}_m) \quad (2.103)$$

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

$$\tilde{\mathbf{A}}_\Phi = \mathbf{T}^{-1} \mathbf{A}_\Phi \mathbf{S}. \quad (2.104)$$

1337 Here, $\mathbf{S} \in \mathbb{R}^{n \times n}$ is the transformation matrix of id_V that maps coordinates
 1338 with respect to \tilde{B} onto coordinates with respect to B , and $\mathbf{T} \in \mathbb{R}^{m \times m}$ is the
 1339 transformation matrix of id_W that maps coordinates with respect to \tilde{C} onto
 1340 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

$$\tilde{\mathbf{b}}_j = s_{1j}\mathbf{b}_1 + \cdots + s_{nj}\mathbf{b}_n = \sum_{i=1}^n s_{ij}\mathbf{b}_i, \quad j = 1, \dots, n. \quad (2.105)$$

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

$$\tilde{\mathbf{c}}_k = t_{1k}\mathbf{c}_1 + \cdots + t_{mk}\mathbf{c}_m = \sum_{l=1}^m t_{lk}\mathbf{c}_l, \quad k = 1, \dots, m. \quad (2.106)$$

1341 We define $S = ((s_{ij})) \in \mathbb{R}^{n \times n}$ as the transformation matrix that maps
1342 coordinates with respect to \tilde{B} onto coordinates with respect to B and
1343 $T = ((t_{lk})) \in \mathbb{R}^{m \times m}$ as the transformation matrix that maps coordinates
1344 with respect to \tilde{C} onto coordinates with respect to C . In particular, the j th
1345 column of S is the coordinate representation of $\tilde{\mathbf{b}}_j$ with respect to B and
1346 the k th column of T is the coordinate representation of $\tilde{\mathbf{c}}_k$ with respect to
1347 C . Note that both S and T are regular.

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

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

1348 where we first expressed the new basis vectors $\tilde{\mathbf{c}}_k \in W$ as linear com-
1349 binations of the basis vectors $\mathbf{c}_l \in W$ and then swapped the order of
1350 summation.

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

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

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

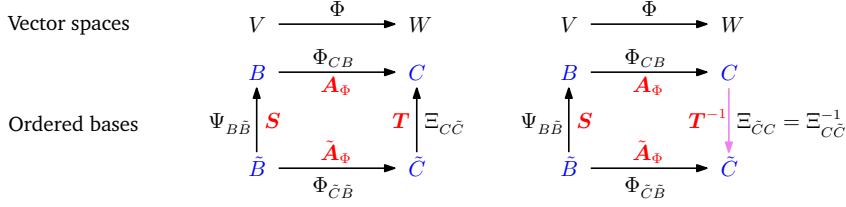
where we exploited the linearity of Φ . Comparing (2.107) and (2.108b), it follows for all $j = 1, \dots, n$ and $l = 1, \dots, m$ that

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

and, therefore,

$$T \tilde{A}_\Phi = A_\Phi S \in \mathbb{R}^{m \times n}, \quad (2.110)$$

Figure 2.10 For a homomorphism $\Phi : V \rightarrow W$ 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

$$\tilde{A}_\Phi = T^{-1} A_\Phi S, \quad (2.111)$$

which proves Theorem 2.19. \square

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 A_Φ of a linear mapping $\Phi : V \rightarrow W$ is replaced by an equivalent matrix \tilde{A}_Φ with

$$\tilde{A}_\Phi = T^{-1} A_\Phi S. \quad (2.112)$$

Figure 2.10 illustrates this relation: Consider a homomorphism $\Phi : V \rightarrow W$ 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 A_Φ 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{A}_Φ as follows: First, we find the matrix representation of the linear mapping $\Psi_{B\tilde{B}} : V \rightarrow V$ 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 A_Φ of $\Phi_{CB} : V \rightarrow W$ to map these coordinates onto the coordinates with respect to C in W . Finally, we use a linear mapping $\Xi_{C\tilde{C}} : W \rightarrow W$ 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:

$$\Phi_{\tilde{C}\tilde{B}} = \Xi_{\tilde{C}C} \circ \Phi_{CB} \circ \Psi_{B\tilde{B}} = \Xi_{C\tilde{C}}^{-1} \circ \Phi_{CB} \circ \Psi_{B\tilde{B}}. \quad (2.113)$$

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

equivalent 1354 **Definition 2.20** (Equivalence). Two matrices $A, \tilde{A} \in \mathbb{R}^{m \times n}$ are *equivalent* if there exist regular matrices $S \in \mathbb{R}^{n \times n}$ and $T \in \mathbb{R}^{m \times m}$, such that $\tilde{A} = T^{-1}AS$.

similar 1357 **Definition 2.21** (Similarity). Two matrices $A, \tilde{A} \in \mathbb{R}^{n \times n}$ are *similar* if there exists a regular matrix $S \in \mathbb{R}^{n \times n}$ with $\tilde{A} = S^{-1}AS$

1359 *Remark.* Similar matrices are always equivalent. However, equivalent matrices are not necessarily similar. \diamond

1361 1362 1363 1364 1365 *Remark.* Consider vector spaces V, W, X . From the remark on page 48 we already know that for linear mappings $\Phi : V \rightarrow W$ and $\Psi : W \rightarrow X$ the mapping $\Psi \circ \Phi : V \rightarrow X$ is also linear. With transformation matrices \mathbf{A}_Φ and \mathbf{A}_Ψ of the corresponding mappings, the overall transformation matrix is $\mathbf{A}_{\Psi \circ \Phi} = \mathbf{A}_\Psi \mathbf{A}_\Phi$. \diamond

1366 In light of this remark, we can look at basis changes from the perspective of composing linear mappings:

- 1368 • \mathbf{A}_Φ is the transformation matrix of a linear mapping $\Phi_{CB} : V \rightarrow W$ with respect to the bases B, C .
- 1370 • $\tilde{\mathbf{A}}_\Phi$ is the transformation matrix of the linear mapping $\Phi_{\tilde{C}\tilde{B}} : V \rightarrow W$ with respect to the bases \tilde{B}, \tilde{C} .
- 1372 • \mathbf{S} is the transformation matrix of a linear mapping $\Psi_{B\tilde{B}} : V \rightarrow V$ (automorphism) that represents \tilde{B} in terms of B . Normally, $\Psi = \text{id}_V$ is the identity mapping in V .
- 1375 • \mathbf{T} is the transformation matrix of a linear mapping $\Xi_{C\tilde{C}} : W \rightarrow W$ (automorphism) that represents \tilde{C} in terms of C . Normally, $\Xi = \text{id}_W$ is the identity mapping in W .

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

$$\tilde{B} \rightarrow \tilde{C} = \tilde{B} \rightarrow B \rightarrow C \rightarrow \tilde{C} \quad (2.114)$$

$$\tilde{\mathbf{A}}_\Phi = \mathbf{T}^{-1} \mathbf{A}_\Phi \mathbf{S}. \quad (2.115)$$

1378 Note that the execution order in (2.115) is from right to left because vectors are multiplied at the right-hand side so that $\mathbf{x} \mapsto \mathbf{S}\mathbf{x} \mapsto \mathbf{A}_\Phi(\mathbf{S}\mathbf{x}) \mapsto \mathbf{T}^{-1}(\mathbf{A}_\Phi(\mathbf{S}\mathbf{x})) = \tilde{\mathbf{A}}_\Phi \mathbf{x}$.

Example 2.23 (Basis Change)

Consider a linear mapping $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^4$ whose transformation matrix is

$$\mathbf{A}_\Phi = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 1 & 3 \\ 3 & 7 & 1 \\ -1 & 2 & 4 \end{bmatrix} \quad (2.116)$$

with respect to the standard bases

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

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

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

Then,

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

where the i th column of S is the coordinate representation of \tilde{b}_i in terms of the basis vectors of B . Similarly, the j th column of T is the coordinate representation of \tilde{c}_j in terms of the basis vectors of C .

Therefore, we obtain

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

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

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 $\sum_{i=1}^3 \lambda_i b_i = \tilde{b}_j$, $j = 1, \dots, 3$.

1381 In Chapter 4, we will be able to exploit the concept of a basis change
1382 to find a basis with respect to which the transformation matrix of an endomorphism has a particularly simple (diagonal) form. In Chapter 10, we
1383 will look at a data compression problem and find a convenient basis onto
1384 which we can project the data while minimizing the compression loss.
1385

2.7.3 Image and Kernel

1386 The image and kernel of a linear mapping are vector subspaces with certain important properties. In the following, we will characterize them more carefully.

1387 **Definition 2.22** (Image and Kernel).

1388 For $\Phi : V \rightarrow W$, we define the *kernel/null space*

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

1389 and the *image/range*

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

kernel
null space

image
range

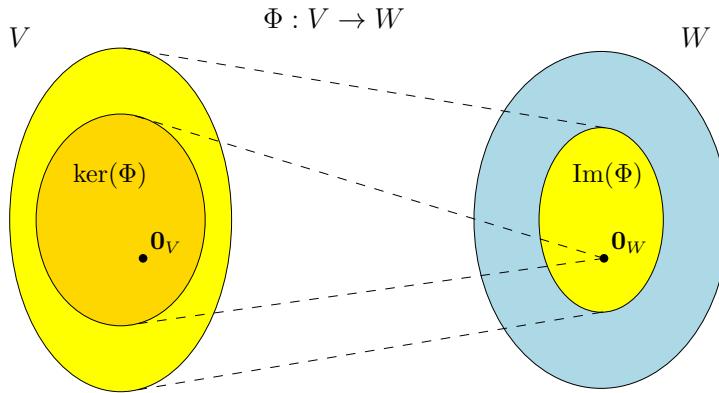


Figure 2.11 Kernel and Image of a linear mapping $\Phi : V \rightarrow W$.

1391 We also call V and W also the *domain* and *codomain* of Φ , respectively.

domain
codomain

1392 Intuitively, the kernel is the set of vectors in $v \in V$ that Φ maps onto
1393 the neutral element $0_W \in W$. The image is the set of vectors $w \in W$ that
1394 can be “reached” by Φ from any vector in V . An illustration is given in
1395 Figure 2.11.

1396 *Remark.* Consider a linear mapping $\Phi : V \rightarrow W$, where V, W are vector
1397 spaces.

- 1398 • It always holds that $\Phi(0_V) = 0_W$ and, therefore, $0_V \in \ker(\Phi)$. In
1399 particular, the null space is never empty.
- 1400 • $\text{Im}(\Phi) \subseteq W$ is a subspace of W , and $\ker(\Phi) \subseteq V$ is a subspace of V .
- 1401 • Φ is injective (one-to-one) if and only if $\ker(\Phi) = \{0\}$

◇

1402
1403 *Remark* (Null Space and Column Space). Let us consider $A \in \mathbb{R}^{m \times n}$ and
1404 a linear mapping $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $x \mapsto Ax$.

- For $A = [a_1, \dots, a_n]$, where a_i are the columns of A , we obtain

$$\text{Im}(\Phi) = \{Ax : x \in \mathbb{R}^n\} = \left\{ \sum_{i=1}^n x_i a_i : x_1, \dots, x_n \in \mathbb{R} \right\} \quad (2.123a)$$

$$= \text{span}[a_1, \dots, a_n] \subseteq \mathbb{R}^m, \quad (2.123b)$$

1405 i.e., the image is the span of the columns of A , also called the *column
1406 space*. Therefore, the column space (image) is a subspace of \mathbb{R}^m , where
1407 m is the “height” of the matrix.

column space

- 1408 • $\text{rk}(A) = \dim(\text{Im}(\Phi))$
- 1409 • The kernel/null space $\ker(\Phi)$ is the general solution to the linear ho-
1410 mogeneous equation system $Ax = 0$ and captures all possible linear
1411 combinations of the elements in \mathbb{R}^n that produce $0 \in \mathbb{R}^m$.
- 1412 • The kernel is a subspace of \mathbb{R}^n , where n is the “width” of the matrix.

- 1413 • The kernel focuses on the relationship among the columns, and we can
 1414 use it to determine whether/how we can express a column as a linear
 1415 combination of other columns.
 1416 • The purpose of the kernel is to determine whether a solution of the
 1417 system of linear equations is unique and, if not, to capture all possible
 1418 solutions.

1419


Example 2.24 (Image and Kernel of a Linear Mapping)

The mapping

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

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

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

$$\text{Im}(\Phi) = \text{span} \left[\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right]. \quad (2.126)$$

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

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

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 a_3 is equivalent to $-\frac{1}{2}$ times the second column a_2 . Therefore, $\mathbf{0} = a_3 + \frac{1}{2}a_2$. In the same way, we see that $a_4 = a_1 - \frac{1}{2}a_2$ and, therefore, $\mathbf{0} = a_1 - \frac{1}{2}a_2 - a_4$. Overall, this gives us the kernel (null space) as

$$\ker(\Phi) = \text{span} \left[\begin{bmatrix} 0 \\ \frac{1}{2} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ \frac{1}{2} \\ 0 \\ 1 \end{bmatrix} \right]. \quad (2.128)$$

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

$$\dim(\ker(\Phi)) + \dim(\text{Im}(\Phi)) = \dim(V). \quad (2.129)$$

1420 Direct consequences of Theorem 2.23 are

- 1421 • If $\dim(\text{Im}(\Phi)) < \dim(V)$ then $\ker(\Phi)$ is non-trivial, i.e., the kernel
1422 contains more than $\mathbf{0}_V$ and $\dim(\ker(\Phi)) \geq 1$.
- 1423 • If A_Φ is the transformation matrix of Φ with respect to an ordered basis
1424 and $\dim(\text{Im}(\Phi)) < \dim(V)$ then the system of linear equations $A_\Phi x =$
1425 $\mathbf{0}$ has infinitely many solutions.
- 1426 • If $\dim(V) = \dim(W)$ then Φ is injective if and only if it is surjective if
1427 and only if it is bijective since $\text{Im}(\Phi) \subseteq W$.

1428 2.8 Affine Spaces

1429 In the following, we will have a closer look at spaces that are offset from
1430 the origin, i.e., spaces that are no longer vector subspaces. Moreover, we
1431 will briefly discuss properties of mappings between these affine spaces,
1432 which resemble linear mappings.

1433 2.8.1 Affine Subspaces

Definition 2.24 (Affine Subspace). Let V be a vector space, $x_0 \in V$ and $U \subseteq V$ a subspace. Then the subset

$$L = x_0 + U := \{x_0 + u : u \in U\} \quad (2.130a)$$

$$= \{v \in V | \exists u \in U : v = x_0 + u\} \subseteq V \quad (2.130b)$$

1434 is called *affine subspace* or *linear manifold* of V . U is called *direction* or
1435 *direction space*, and x_0 is called *support point*. In Chapter 12, we refer to
1436 such a subspace as a *hyperplane*.

affine subspace
linear manifold
direction
direction space
support point
hyperplane

1437 Note that the definition of an affine subspace excludes $\mathbf{0}$ if $x_0 \notin U$.
1438 Therefore, an affine subspace is not a (linear) subspace (vector subspace)
1439 of V for $x_0 \notin U$.

1440 Examples of affine subspaces are points, lines and planes in \mathbb{R}^3 , which
1441 do not (necessarily) go through the origin.

parameters

1442 *Remark.* Consider two affine subspaces $L = x_0 + U$ and $\tilde{L} = \tilde{x}_0 + \tilde{U}$ of a
1443 vector space V . Then, $L \subseteq \tilde{L}$ if and only if $U \subseteq \tilde{U}$ and $x_0 - \tilde{x}_0 \in \tilde{U}$.

Affine subspaces are often described by *parameters*: Consider a k -dimensional affine space $L = x_0 + U$ of V . If (b_1, \dots, b_k) is an ordered basis of U , then every element $x \in L$ can be uniquely described as

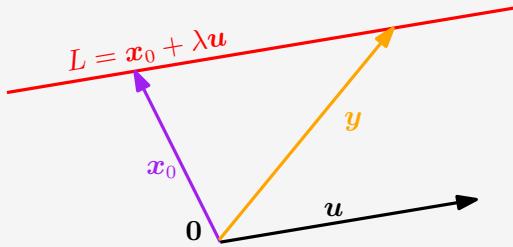
$$x = x_0 + \lambda_1 b_1 + \dots + \lambda_k b_k, \quad (2.131)$$

1444 where $\lambda_1, \dots, \lambda_k \in \mathbb{R}$. This representation is called *parametric equation*
1445 of L with directional vectors b_1, \dots, b_k and *parameters* $\lambda_1, \dots, \lambda_k$. \diamond

parametric equation
parameters

Example 2.25 (Affine Subspaces)

Figure 2.12 Vectors y on a line lie in an affine subspace L with support point x_0 and direction u .



lines

- One-dimensional affine subspaces are called *lines* and can be written as $y = x_0 + \lambda x_1$, where $\lambda \in \mathbb{R}$, where $U = \text{span}[x_1] \subseteq \mathbb{R}^n$ is a one-dimensional subspace of \mathbb{R}^n . This means, a line is defined by a support point x_0 and a vector x_1 that defines the direction. See Figure 2.12 for an illustration.

planes

- Two-dimensional affine subspaces of \mathbb{R}^n are called *planes*. The parametric equation for planes is $y = x_0 + \lambda_1 x_1 + \lambda_2 x_2$, where $\lambda_1, \lambda_2 \in \mathbb{R}$ and $U = [x_1, x_2] \subseteq \mathbb{R}^n$. This means, a plane is defined by a support point x_0 and two linearly independent vectors x_1, 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 $y = x_0 + \sum_{i=1}^{n-1} \lambda_i x_i$, where x_1, \dots, 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 x_0 and $(n - 1)$ linearly independent vectors x_1, \dots, 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.

1446 **Remark** (Inhomogeneous linear equation systems and affine subspaces).
1447 For $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ the solution of the linear equation system
1448 $Ax = b$ is either the empty set or an affine subspace of \mathbb{R}^n of dimension
1449 $n - \text{rk}(A)$. In particular, the solution of the linear equation $\lambda_1 x_1 + \dots +$
1450 $\lambda_n x_n = b$, where $(\lambda_1, \dots, \lambda_n) \neq (0, \dots, 0)$, is a hyperplane in \mathbb{R}^n .

1451 In \mathbb{R}^n , every k -dimensional affine subspace is the solution of a linear
1452 inhomogeneous equation system $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and
1453 $\text{rk}(A) = n - k$. Recall that for homogeneous equation systems $Ax = 0$
1454 the solution was a vector subspace, which we can also think of as a special
1455 affine space with support point $x_0 = 0$. ◇

1456

2.8.2 Affine Mappings

1457 Similar to linear mappings between vector spaces, which we discussed
 1458 in Section 2.7, we can define affine mappings between two affine spaces.
 1459 Linear and affine mappings are closely related. Therefore, many properties
 1460 that we already know from linear mappings, e.g., that the composition of
 1461 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 $\Phi : V \rightarrow W$ and $a \in W$ the mapping

$$\phi : V \rightarrow W \quad (2.132)$$

$$x \mapsto a + \Phi(x) \quad (2.133)$$

1462 is an *affine mapping* from V to W . The vector a is called the *translation*
 1463 *vector* of ϕ .

affine mapping
translation vector

- 1464 • Every affine mapping $\phi : V \rightarrow W$ is also the composition of a linear
 1465 mapping $\Phi : V \rightarrow W$ and a translation $\tau : W \rightarrow W$ in W , such that
 1466 $\phi = \tau \circ \Phi$. The mappings Φ and τ are uniquely determined.
- 1467 • The composition $\phi' \circ \phi$ of affine mappings $\phi : V \rightarrow W$, $\phi' : W \rightarrow X$ is
 1468 affine.
- 1469 • Affine mappings keep the geometric structure invariant. They also pre-
 1470 serve the dimension and parallelism.

1471

Exercises

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

$$a \star b := ab + a + b, \quad a, b \in \mathbb{R} \setminus \{-1\} \quad (2.134)$$

- 1472 1 Show that $(\mathbb{R} \setminus \{-1\}, \star)$ is an Abelian group
 2 Solve

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

1473 in the Abelian group $(\mathbb{R} \setminus \{-1\}, \star)$, 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

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

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:

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

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

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

1474 1 Show that (\mathbb{Z}_n, \oplus) is a group. Is it Abelian?

1475 2 We now define another operation \otimes for all \bar{a} and \bar{b} in \mathbb{Z}_n as

$$\bar{a} \otimes \bar{b} = \overline{a \times b} \quad (2.135)$$

1476 where $a \times b$ represents the usual multiplication in \mathbb{Z} .

1477 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 $\mathbb{Z}_5 \setminus \{\bar{0}\}$.

1478 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.

1481 3 Show that $(\mathbb{Z}_8 \setminus \{\bar{0}\}, \otimes)$ is not a group.

1482 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 $(\mathbb{Z}_n \setminus \{\bar{0}\}, \otimes)$ is a group if and only if $n \in \mathbb{N} \setminus \{0\}$ is prime.

1486 2.3 Consider the set \mathcal{G} of 3×3 matrices defined as:

$$\mathcal{G} = \left\{ \begin{bmatrix} 1 & x & z \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \mid x, y, z \in \mathbb{R} \right\} \quad (2.136)$$

1486 We define \cdot as the standard matrix multiplication.

1487 Is (\mathcal{G}, \cdot) a group? If yes, is it Abelian? Justify your answer.

1488 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

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

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

1

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

2

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

- 3 Using Gaussian elimination find all solutions of the inhomogeneous equation system $\mathbf{Ax} = \mathbf{b}$ with

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

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

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

¹⁴⁹¹ and $\sum_{i=1}^3 x_i = 1$.

- ¹⁴⁹² 2.7 Determine the inverse of the following matrices if possible:

1

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

2

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

- ¹⁴⁹³ 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 $D = \{(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3 \mid \xi_2 \in \mathbb{Z}\}$

- ¹⁴⁹⁹ 2.9 Are the following vectors linearly independent?

1

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

2

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

2.10 Write

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

as linear combination of

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

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

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

1500

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 $\mathbf{A}_1\mathbf{x} = \mathbf{0}$ and U_2 is the solution space of the homogeneous equation system $\mathbf{A}_2\mathbf{x} = \mathbf{0}$ with

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

1501

1 Determine the dimension of U_1, U_2

1502

2 Determine bases of U_1 and U_2

1503

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 \mathbf{A}_1 and U_2 is spanned by the columns of \mathbf{A}_2 with

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

1504

1 Determine the dimension of U_1, U_2

1505

2 Determine bases of U_1 and U_2

1506 3 Determine a basis of $U_1 \cap U_2$

1507 2.13 Let $F = \{(x, y, z) \in \mathbb{R}^3 \mid x+y-z=0\}$ and $G = \{(a-b, a+b, a-3b) \mid a, b \in \mathbb{R}\}$.

1508 1 Show that F and G are subspaces of \mathbb{R}^3 .

1509 2 Calculate $F \cap G$ without resorting to any basis vector.

1510 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.

1511 2.14 Are the following mappings linear?

1512 1 Let $a, b \in \mathbb{R}$.

$$\Phi : L^1([a, b]) \rightarrow \mathbb{R}$$

$$f \mapsto \Phi(f) = \int_a^b f(x) dx,$$

1513 where $L^1([a, b])$ denotes the set of integrable function on $[a, b]$.

1514 2

$$\Phi : C^1 \rightarrow C^0$$

$$f \mapsto \Phi(f) = f'.$$

1515 where for $k \geq 1$, C^k denotes the set of k times continuously differentiable functions, and C^0 denotes the set of continuous functions.

1516 3

$$\Phi : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \Phi(x) = \cos(x)$$

1517 4

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$$

$$\mathbf{x} \mapsto \begin{bmatrix} 1 & 2 & 3 \\ 1 & 4 & 3 \end{bmatrix} \mathbf{x}$$

1518 5 Let θ be in $[0, 2\pi[$.

$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

$$\mathbf{x} \mapsto \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \mathbf{x}$$

2.15 Consider the linear mapping

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^4$$

$$\Phi \left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) = \begin{bmatrix} 3x_1 + 2x_2 + x_3 \\ x_1 + x_2 + x_3 \\ x_1 - 3x_2 \\ 2x_1 + 3x_2 + x_3 \end{bmatrix}$$

1519 • Find the transformation matrix A_Φ

1520 • Determine $\text{rk}(A_\Phi)$

1521 • Compute kernel and image of Φ . What are $\dim(\ker(\Phi))$ and $\dim(\text{Im}(\Phi))$?

- 1519 2.16 Let E be a vector space. Let f and g be two endomorphisms on E such that
 1520 $f \circ g = \text{id}_E$ (i.e. $f \circ g$ is the identity isomorphism). Show that $\ker(f) =$
 1521 $\ker(g \circ f)$, $\text{Im}(g) = \text{Im}(g \circ f)$ and that $\ker(f) \cap \text{Im}(g) = \{\mathbf{0}_E\}$.

- 1521 2.17 Consider an endomorphism $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ whose transformation matrix
 (with respect to the standard basis in \mathbb{R}^3) is

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

1522 1 Determine $\ker(\Phi)$ and $\text{Im}(\Phi)$.

2 Determine the transformation matrix $\tilde{\mathbf{A}}_\Phi$ with respect to the basis

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

1523 i.e., perform a basis change toward the new basis B .

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

$$\mathbf{b}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \mathbf{b}'_1 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad \mathbf{b}'_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (2.137)$$

1524 and let us define two ordered bases $B = (\mathbf{b}_1, \mathbf{b}_2)$ and $B' = (\mathbf{b}'_1, \mathbf{b}'_2)$ of \mathbb{R}^2 .

1525 1 Show that B and B' are two bases of \mathbb{R}^2 and draw those basis vectors.

1526 2 Compute the matrix \mathbf{P}_1 that performs a basis change from B' to B .

1526 3 We consider $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$, 3 vectors of \mathbb{R}^3 defined in the standard basis of \mathbb{R}
 as

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad (2.138)$$

1527 and we define $C = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$.

1528 1 Show that C is a basis of \mathbb{R}^3 , e.g., by using determinants (see Sec-
 1529 tion 4.1)

1530 2 Let us call $C' = (\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3)$ the standard basis of \mathbb{R}^3 . Determine the
 1531 matrix \mathbf{P}_2 that performs the basis change from C to C' .

1531 4 We consider a homomorphism $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, such that

$$\begin{aligned} \Phi(\mathbf{b}_1 + \mathbf{b}_2) &= \mathbf{c}_2 + \mathbf{c}_3 \\ \Phi(\mathbf{b}_1 - \mathbf{b}_2) &= 2\mathbf{c}_1 - \mathbf{c}_2 + 3\mathbf{c}_3 \end{aligned} \quad (2.139)$$

1532 where $B = (\mathbf{b}_1, \mathbf{b}_2)$ and $C = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ are ordered bases of \mathbb{R}^2 and \mathbb{R}^3 ,
 1533 respectively.

1534 Determine the transformation matrix \mathbf{A}_Φ of Φ with respect to the ordered
 1535 bases B and C .

1536 5 Determine \mathbf{A}' , the transformation matrix of Φ with respect to the bases
 1537 B' and C' .

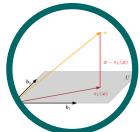
1538 6 Let us consider the vector $\mathbf{x} \in \mathbb{R}^2$ whose coordinates in B' are $[2, 3]^\top$. In
 1539 other words, $\mathbf{x} = 2\mathbf{b}'_1 + 3\mathbf{b}'_2$.

- 1540 1 Calculate the coordinates of x in B .
1541 2 Based on that, compute the coordinates of $\Phi(x)$ expressed in C .
1542 3 Then, write $\Phi(x)$ in terms of c'_1, c'_2, c'_3 .
1543 4 Use the representation of x in B' and the matrix A' to find this result
1544 directly.

3

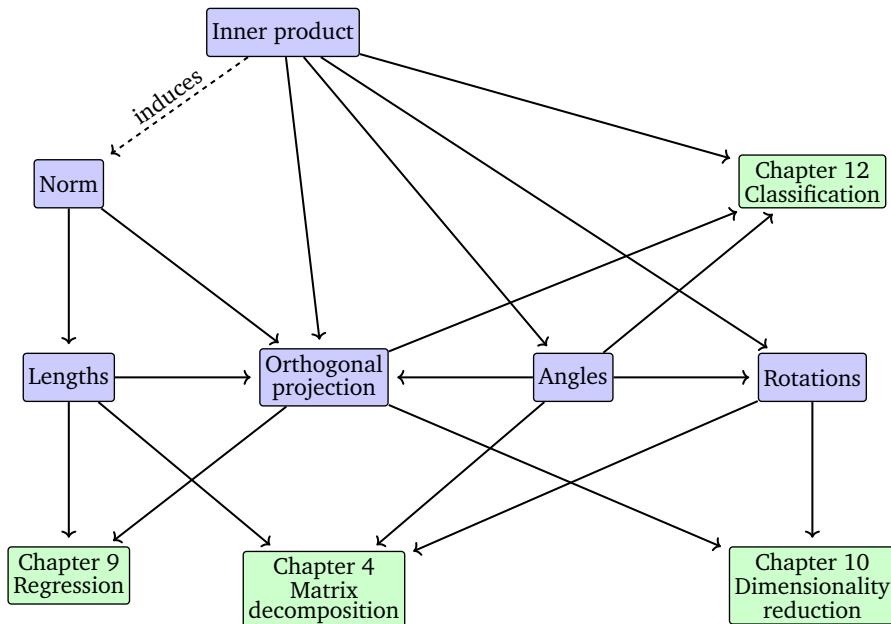
1541

Analytic Geometry



1542 In Chapter 2, we studied vectors, vector spaces and linear mappings at
1543 a general but abstract level. In this chapter, we will add some geometric
1544 interpretation and intuition to all of these concepts. In particular, we will
1545 look at geometric vectors, compute their lengths and distances or angles
1546 between two vectors. To be able to do this, we equip the vector space with
1547 an inner product that induces the geometry of the vector space. Inner
1548 products and their corresponding norms and metrics capture the intuitive
1549 notions of similarity and distances, which we use to develop the Support
1550 Vector Machine in Chapter 12. We will then use the concepts of lengths
1551 and angles between vectors to discuss orthogonal projections, which will
1552 play a central role when we discuss principal component analysis in Chap-
1553 ter 10 and regression via maximum likelihood estimation in Chapter 9.
1554 Figure 3.1 gives an overview of how concepts in this chapter are related
1555 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.



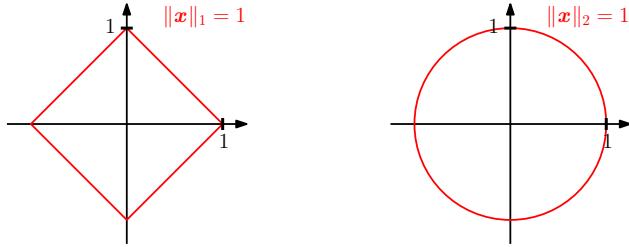


Figure 3.3 For different norms, the red lines indicate the set of vectors with norm 1. Left: Manhattan norm; Right: Euclidean distance.

1556

3.1 Norms

1557 When we think of geometric vectors, i.e., directed line segments that start
 1558 at the origin, then intuitively the length of a vector is the distance of the
 1559 “end” of this directed line segment from the origin. In the following, we
 1560 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

$$\|\cdot\| : V \rightarrow \mathbb{R}, \quad (3.1)$$

$$\mathbf{x} \mapsto \|\mathbf{x}\|, \quad (3.2)$$

1561 which assigns each vector \mathbf{x} its *length* $\|\mathbf{x}\| \in \mathbb{R}$, such that for all $\lambda \in \mathbb{R}$ length
 1562 and $\mathbf{x}, \mathbf{y} \in V$ the following hold:

- 1563 • *Absolutely homogeneous*: $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$
- 1564 • *Triangle inequality*: $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$
- 1565 • *Positive definite*: $\|\mathbf{x}\| \geq 0$ and $\|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0}$.

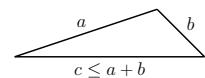
Triangle inequality

Positive definite

1566 In geometric terms, the triangle inequality states that for any triangle,
 1567 the sum of the lengths of any two sides must be greater than or equal to
 1568 the length of the remaining side; see Figure 3.2 for an illustration.

1569 Recall that for a vector $\mathbf{x} \in \mathbb{R}^n$ we denote the elements of the vector
 1570 using a subscript, that is x_i is the i^{th} element of the vector \mathbf{x} .

Figure 3.2 Triangle inequality.



Example 3.1 (Manhattan Norm)

The *Manhattan norm* on \mathbb{R}^n is defined for $\mathbf{x} \in \mathbb{R}^n$ as

Manhattan norm

$$\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|, \quad (3.3)$$

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

 ℓ_1 norm

Example 3.2 (Euclidean Norm)

The length of a vector $\mathbf{x} \in \mathbb{R}^n$ is given by

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{\mathbf{x}^\top \mathbf{x}}, \quad (3.4)$$

Euclidean distance

Euclidean norm

ℓ_2 norm

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

Cauchy-Schwarz
inequality

1571 *Remark.* Throughout this book, we will use the Euclidean norm (3.4) by default if not stated otherwise. \diamond

1572 *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*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\|. \quad (3.5)$$

1573

\diamond

1574

3.2 Inner Products

1575 1576 1577 1578 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.

1579

3.2.1 Dot Product

scalar product
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

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^n x_i y_i. \quad (3.6)$$

1580

1581 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. 1582

1583

3.2.2 General Inner Products

bilinear mapping

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*

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 $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V, \lambda \in \mathbb{R}$

$$\Omega(\lambda\mathbf{x} + \mathbf{y}, \mathbf{z}) = \lambda\Omega(\mathbf{x}, \mathbf{z}) + \Omega(\mathbf{y}, \mathbf{z}) \quad (3.7)$$

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

1584 Here, (3.7) asserts that Ω is linear in the first argument, and (3.8) asserts
1585 that Ω is linear in the second argument.

1586 **Definition 3.2.** Let V be a vector space and $\Omega : V \times V \rightarrow \mathbb{R}$ be a bilinear
1587 mapping that takes two vectors and maps them onto a real number. Then

- 1588 • Ω is called *symmetric* if $\Omega(\mathbf{x}, \mathbf{y}) = \Omega(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in V$, i.e., the
1589 order of the arguments does not matter.
- Ω is called *positive definite* if

symmetric

positive definite

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

1590 **Definition 3.3.** Let V be a vector space and $\Omega : V \times V \rightarrow \mathbb{R}$ be a bilinear
1591 mapping that takes two vectors and maps them onto a real number. Then

- 1592 • A positive definite, symmetric bilinear mapping $\Omega : V \times V \rightarrow \mathbb{R}$ is called
1593 an *inner product* on V . We typically write $\langle \mathbf{x}, \mathbf{y} \rangle$ instead of $\Omega(\mathbf{x}, \mathbf{y})$.
- 1594 • The pair $(V, \langle \cdot, \cdot \rangle)$ is called an *inner product space* or (real) *vector space*
1595 with *inner product*. If we use the dot product defined in (3.6), we call
1596 $(V, \langle \cdot, \cdot \rangle)$ a *Euclidean vector space*.

inner product

inner product space

vector space with
inner productEuclidean vector
space

1597 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

$$\langle \mathbf{x}, \mathbf{y} \rangle := x_1y_1 - (x_1y_2 + x_2y_1) + 2x_2y_2 \quad (3.10)$$

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

1599 Symmetric, positive definite matrices play an important role in machine
1600 learning, and they are defined via the inner product.

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

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

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

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

$$\forall \mathbf{x} \in V \setminus \{\mathbf{0}\} : \mathbf{x}^\top \mathbf{A} \mathbf{x} > 0. \quad (3.12)$$

1601 symmetric, positive definite
1602 positive definite
1603 positive definite
1604 symmetric, positive semi-definite

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

Example 3.4 (Symmetric, Positive Definite Matrices)

Consider the following matrices:

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

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

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

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

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

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

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

1605 defines an inner product with respect to an ordered basis B where $\hat{\mathbf{x}}$ and
1606 $\hat{\mathbf{y}}$ are the coordinate representations of $\mathbf{x}, \mathbf{y} \in V$ 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 $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{R}$ is an inner product if and only if there exists a symmetric, positive definite matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with*

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

1607 The following properties hold if $\mathbf{A} \in \mathbb{R}^{n \times n}$ is symmetric and positive
1608 definite:

- The null space (kernel) of A consists only of $\mathbf{0}$ because $\mathbf{x}^\top A\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$. This implies that $A\mathbf{x} \neq \mathbf{0}$ if $\mathbf{x} \neq \mathbf{0}$.
- The diagonal elements a_{ii} of A are positive because $a_{ii} = \mathbf{e}_i^\top A \mathbf{e}_i > 0$, where \mathbf{e}_i is the i th 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

$$\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} \quad (3.17)$$

Inner products induce norms.

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 $\mathbf{x} = [1, 1]^\top \in \mathbb{R}^2$. If we use the dot product as the inner product, with (3.17) we obtain

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}} = \sqrt{1^2 + 1^2} = \sqrt{2} \quad (3.18)$$

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

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

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

$$\langle \mathbf{x}, \mathbf{x} \rangle = x_1^2 - x_1 x_2 + x_2^2 = 1 - 1 + 1 = 1 \implies \|\mathbf{x}\| = \sqrt{1} = 1, \quad (3.20)$$

such that \mathbf{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

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

distance
Euclidean distance

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

$$d : V \times V \rightarrow \mathbb{R} \quad (3.22)$$

$$(\mathbf{x}, \mathbf{y}) \mapsto d(\mathbf{x}, \mathbf{y}) \quad (3.23)$$

metric

¹⁶²² is called *metric*.

¹⁶²³ *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. \diamond

¹⁶²⁷ A metric d satisfies:

positive definite

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

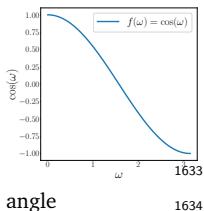
symmetric

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

Triangular
inequality

¹⁶³¹ 3. *Triangular inequality*: $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

¹⁶³² **Figure 3.4** When restricted to $[0, \pi]$ then $f(\omega) = \cos(\omega)$ returns a unique number in the interval $[-1, 1]$.



angle

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

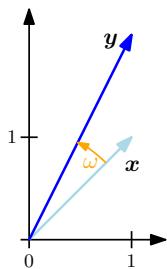
$$-1 \leq \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} \leq 1. \quad (3.24)$$

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

$$\cos \omega = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}, \quad (3.25)$$

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

¹⁶³⁷ **Figure 3.5** The angle ω between two vectors \mathbf{x}, \mathbf{y} is computed using the inner product.



Example 3.6 (Angle between Vectors)

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

$$\cos \omega = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\sqrt{\langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle}} = \frac{\mathbf{x}^\top \mathbf{y}}{\sqrt{\mathbf{x}^\top \mathbf{x} \mathbf{y}^\top \mathbf{y}}} = \frac{3}{\sqrt{10}}, \quad (3.26)$$

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

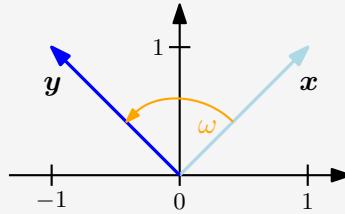
1638 The inner product also allows us to characterize vectors that are orthogonal.
1639

1640 **Definition 3.7** (Orthogonality). Two vectors \mathbf{x} and \mathbf{y} are *orthogonal* if and only if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, and we write $\mathbf{x} \perp \mathbf{y}$. If additionally $\|\mathbf{x}\| = 1 = \|\mathbf{y}\|$, i.e., the vectors are unit vectors, then \mathbf{x} and \mathbf{y} are *orthonormal*.

1643 An implication of this definition is that the 0-vector is orthogonal to every vector in the vector space.
1644

1645 *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. ◇
1646
1647
1648

Example 3.7 (Orthogonal Vectors)



orthogonal
orthonormal

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

Consider two vectors $\mathbf{x} = [1, 1]^\top, \mathbf{y} = [-1, 1]^\top \in \mathbb{R}^2$, 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 \mathbf{x} and \mathbf{y} of 90° , such that $\mathbf{x} \perp \mathbf{y}$. However, if we choose the inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{y}, \quad (3.27)$$

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

$$\cos \omega = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} = -\frac{1}{3} \implies \omega \approx 1.91 \text{ rad} \approx 109.5^\circ, \quad (3.28)$$

and \mathbf{x} and \mathbf{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 $\mathbf{A} \in \mathbb{R}^{n \times n}$ is an *orthogonal matrix* if and only if its columns are orthonormal so that

orthogonal matrix

$$\mathbf{A}\mathbf{A}^\top = \mathbf{I} = \mathbf{A}^\top \mathbf{A}, \quad (3.29)$$

which implies that

$$\mathbf{A}^{-1} = \mathbf{A}^\top, \quad (3.30)$$

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

Transformations with orthogonal matrices preserve distances and angles.

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

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

$$\|\mathbf{Ax}\|^2 = (\mathbf{Ax})^\top (\mathbf{Ax}) = \mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax} = \mathbf{x}^\top \mathbf{I}\mathbf{x} = \mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|^2. \quad (3.31)$$

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

$$\cos \omega = \frac{(\mathbf{Ax})^\top (\mathbf{Ay})}{\|\mathbf{Ax}\| \|\mathbf{Ay}\|} = \frac{\mathbf{x}^\top \mathbf{A}^\top \mathbf{Ay}}{\sqrt{\mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{x} \mathbf{y}^\top \mathbf{A}^\top \mathbf{A} \mathbf{y}}} = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}, \quad (3.32)$$

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

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 $\{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ of V . If

$$\langle \mathbf{b}_i, \mathbf{b}_j \rangle = 0 \quad \text{for } i \neq j \quad (3.33)$$

$$\langle \mathbf{b}_i, \mathbf{b}_i \rangle = 1 \quad (3.34)$$

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 $\{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ given a set $\{\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n\}$ 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

$$\mathbf{b}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{b}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (3.35)$$

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

1667 We will exploit the concept of an orthonormal basis in Chapter 12 and
1668 Chapter 10 when we discuss Support Vector Machines and Principal Com-
1669 ponent Analysis.

3.6 Inner Product of Functions

1670 Thus far, we looked at properties of inner products to compute lengths,
1671 angles and distances. We focused on inner products of finite-dimensional
1672 vectors.

1673 In the following, we will look at an example of inner products of a
1674 different type of vectors: inner products of functions.

1675 The inner products we discussed so far were defined for vectors with a
1676 finite number of entries. We can think of a vector $\mathbf{x} \in \mathbb{R}^n$ as function with
1677 n function values. The concept of an inner product can be generalized to
1678 vectors with an infinite number of entries (countably infinite) and also
1679 continuous-valued functions (uncountably infinite). Then, the sum over
1680 individual components of vectors, see (3.6) for example, turns into an
1681 integral.

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

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

1682 for lower and upper limits $a, b < \infty$, respectively. As with our usual in-
1683 ner product, we can define norms and orthogonality by looking at the
1684 inner product. If (3.36) evaluates to 0, the functions u and v are orthogo-
1685 nal. To make the above inner product mathematically precise, we need to
1686 take care of measures, and the definition of integrals. Furthermore, unlike
1687 inner product on finite-dimensional vectors, inner products on functions
1688 may diverge (have infinite value). Some careful definitions need to be ob-
1689 served, which requires a foray into real and functional analysis which we
1690 do not cover in this book.

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

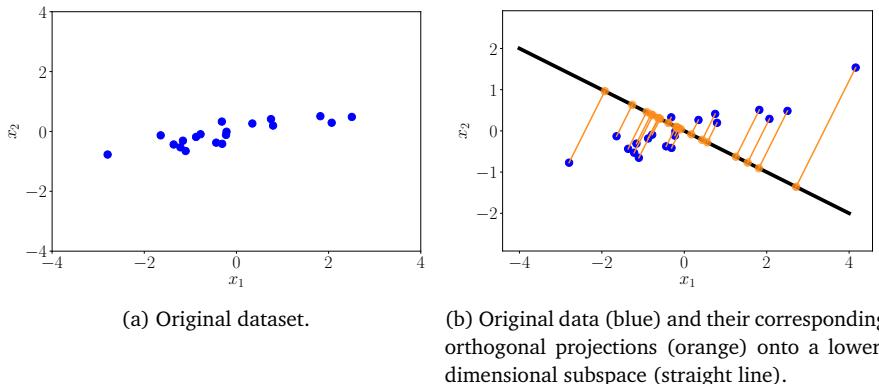
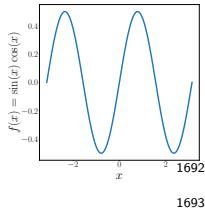


Figure 3.7 $f(x) = \sin(x)\cos(x)$.



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.

Remark. It also holds that the collection of functions

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

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

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

1696

3.7 Orthogonal Projections

1697 Projections are an important class of linear transformations (besides rotations and reflections). Projections play an important role in graphics,
1698 coding theory, statistics and machine learning. In machine learning, we
1699 often deal with data that is high-dimensional. High-dimensional data is
1700 often hard to analyze or visualize. However, high-dimensional data quite
1701 often possesses the property that only a few dimensions contain most in-
1702 formation, and most other dimensions are not essential to describe key
1703 properties of the data. When we compress or visualize high-dimensional
1704 data we will lose information. To minimize this compression loss, we
1705 ideally find the most informative dimensions in the data. Then, we can
1706 project the original high-dimensional data onto a lower-dimensional fea-
1707 ture space and work in this lower-dimensional space to learn more about
1708 the dataset and extract patterns. For example, machine learning algo-
1709 rithms, such as Principal Component Analysis (PCA) by Pearson (1901b);
1710

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

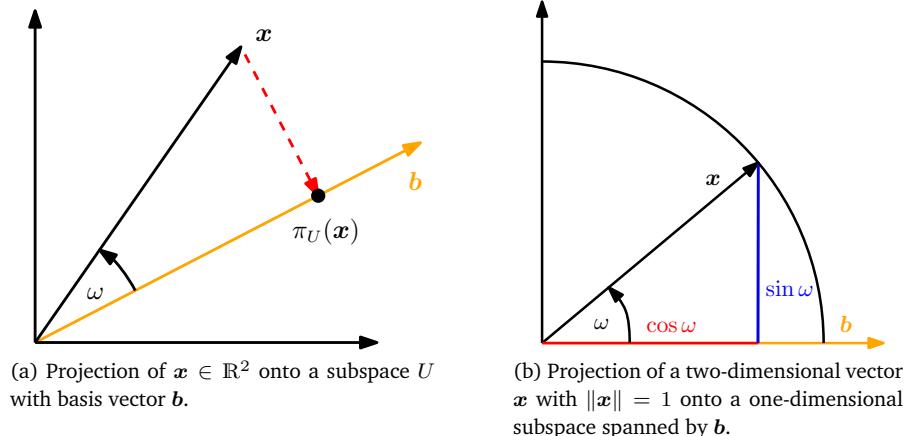


Figure 3.9
Examples of
projections onto
one-dimensional
subspaces.

1711 Hotelling (1933) and Deep Neural Networks (e.g., deep auto-encoders Deng
 1712 et al. (2010)), heavily exploit the idea of dimensionality reduction. In the
 1713 following, we will focus on orthogonal projections, which we will use in
 1714 Chapter 10 for linear dimensionality reduction and in Chapter 12 for clas-
 1715 sification. Even linear regression, which we discuss in Chapter 9, can be
 1716 interpreted using orthogonal projections. For a given lower-dimensional
 1717 subspace, orthogonal projections of high-dimensional data retain as much
 1718 information as possible and minimize the difference/error between the
 1719 original data and the corresponding projection. An illustration of such an
 1720 orthogonal projection is given in Figure 3.8.

1721 Before we detail how to obtain these projections, let us define what a
 1722 projection actually is.

1723 **Definition 3.10** (Projection). Let V be a vector space and $W \subseteq V$ a
 1724 subspace of V . A linear mapping $\pi : V \rightarrow W$ is called a *projection* if
 1725 $\pi^2 = \pi \circ \pi = \pi$.

1726 **Remark** (Projection matrix). Since linear mappings can be expressed by
 1727 transformation matrices (see Section 2.7), the definition above applies
 1728 equally to a special kind of transformation matrices, the *projection matrices*
 1729 P_π , which exhibit the property that $P_\pi^2 = P_\pi$. ◊

1730 In the following, we will derive orthogonal projections of vectors in the
 1731 inner product space $(\mathbb{R}^n, \langle \cdot, \cdot \rangle)$ onto subspaces. We will start with one-
 1732 dimensional subspaces, which are also called *lines*. If not mentioned oth-
 1733 erwise, we assume the dot product $\langle x, y \rangle = x^\top y$ as the inner product.
 1734 lines

3.7.1 Projection onto 1-Dimensional Subspaces (Lines)

1735 Assume we are given a line (1-dimensional subspace) through the origin
 1736 with basis vector $b \in \mathbb{R}^n$. The line is a one-dimensional subspace $U \subseteq \mathbb{R}^n$

1738 spanned by \mathbf{b} . When we project $\mathbf{x} \in \mathbb{R}^n$ onto U , we want to find the
 1739 point $\pi_U(\mathbf{x}) \in U$ that is closest to \mathbf{x} . Using geometric arguments, let us
 1740 characterize some properties of the projection $\pi_U(\mathbf{x})$ (Fig. 3.9 serves as
 1741 an illustration):

- 1742 • The projection $\pi_U(\mathbf{x})$ is closest to \mathbf{x} , where “closest” implies that the
 1743 distance $\|\mathbf{x} - \pi_U(\mathbf{x})\|$ is minimal. It follows that the segment $\pi_U(\mathbf{x}) - \mathbf{x}$
 1744 from $\pi_U(\mathbf{x})$ to \mathbf{x} is orthogonal to U and, therefore, the basis \mathbf{b} of U . The
 1745 orthogonality condition yields $\langle \pi_U(\mathbf{x}) - \mathbf{x}, \mathbf{b} \rangle = 0$ since angles between
 1746 vectors are defined by means of the inner product.
 1747 • The projection $\pi_U(\mathbf{x})$ of \mathbf{x} onto U must be an element of U and, there-
 1748 fore, a multiple of the basis vector \mathbf{b} that spans U . Hence, $\pi_U(\mathbf{x}) = \lambda \mathbf{b}$,
 1749 for some $\lambda \in \mathbb{R}$.

λ is then the
 1749 coordinate of $\pi_U(\mathbf{x})$
 with respect to \mathbf{b} .
 1750

In the following three steps, we determine the coordinate λ , the projection
 1751 $\pi_U(\mathbf{x}) \in U$ and the projection matrix \mathbf{P}_π that maps arbitrary $\mathbf{x} \in \mathbb{R}^n$ onto
 1752 U .

1. Finding the coordinate λ . The orthogonality condition yields

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

$$\stackrel{\pi_U(\mathbf{x}) = \lambda \mathbf{b}}{\iff} \langle \mathbf{x} - \lambda \mathbf{b}, \mathbf{b} \rangle = 0. \quad (3.39)$$

With a general inner
 product, we get
 $\lambda = \langle \mathbf{x}, \mathbf{b} \rangle$ if
 $\|\mathbf{b}\| = 1$.
 1750

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

$$\langle \mathbf{x}, \mathbf{b} \rangle - \lambda \langle \mathbf{b}, \mathbf{b} \rangle = 0 \quad (3.40)$$

$$\iff \lambda = \frac{\langle \mathbf{x}, \mathbf{b} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} = \frac{\langle \mathbf{x}, \mathbf{b} \rangle}{\|\mathbf{b}\|^2} \quad (3.41)$$

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

$$\lambda = \frac{\mathbf{b}^\top \mathbf{x}}{\mathbf{b}^\top \mathbf{b}} = \frac{\mathbf{b}^\top \mathbf{x}}{\|\mathbf{b}\|^2} \quad (3.42)$$

1753 If $\|\mathbf{b}\| = 1$, then the coordinate λ of the projection is given by $\mathbf{b}^\top \mathbf{x}$.

2. Finding the projection point $\pi_U(\mathbf{x}) \in U$. Since $\pi_U(\mathbf{x}) = \lambda \mathbf{b}$ we imme-
 1754 diately obtain with (3.42) that

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

where the last equality holds for the dot product only. We can also
 1755 compute the length of $\pi_U(\mathbf{x})$ by means of Definition 3.1 as

$$\|\pi_U(\mathbf{x})\| = \|\lambda \mathbf{b}\| = |\lambda| \|\mathbf{b}\|. \quad (3.44)$$

1756 This means that our projection is of length $|\lambda|$ times the length of \mathbf{b} .
 1757 This also adds the intuition that λ is the coordinate of $\pi_U(\mathbf{x})$ with
 respect to the basis vector \mathbf{b} that spans our one-dimensional subspace
 U .

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

$$\|\pi_U(\mathbf{x})\| \stackrel{(3.43)}{=} \frac{|\mathbf{b}^\top \mathbf{x}|}{\|\mathbf{b}\|^2} \|\mathbf{b}\| \stackrel{(3.25)}{=} |\cos \omega| \|\mathbf{x}\| \|\mathbf{b}\| \frac{\|\mathbf{b}\|}{\|\mathbf{b}\|^2} = |\cos \omega| \|\mathbf{x}\|. \quad (3.45)$$

1758 Here, ω is the angle between \mathbf{x} and \mathbf{b} . This equation should be familiar
 1759 from trigonometry: If $\|\mathbf{x}\| = 1$ then \mathbf{x} lies on the unit circle. It follows
 1760 that the projection onto the horizontal axis spanned by \mathbf{b} is exactly
 1761 $\cos \omega$, and the length of the corresponding vector $\pi_U(\mathbf{x}) = |\cos \omega|$. An
 1762 illustration is given in Figure 3.9.

The horizontal axis
is a one-dimensional
subspace.

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

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

we immediately see that

$$\mathbf{P}_\pi = \frac{\mathbf{b} \mathbf{b}^\top}{\|\mathbf{b}\|^2}. \quad (3.47)$$

1763 Note that $\mathbf{b} \mathbf{b}^\top$ is a symmetric matrix (with rank 1) and $\|\mathbf{b}\|^2 = \langle \mathbf{b}, \mathbf{b} \rangle$
 1764 is a scalar.

Projection matrices
are always
symmetric.

1765 The projection matrix \mathbf{P}_π projects any vector $\mathbf{x} \in \mathbb{R}^n$ onto the line through
 1766 the origin with direction \mathbf{b} (equivalently, the subspace U spanned by \mathbf{b}).

1767 *Remark.* The projection $\pi_U(\mathbf{x}) \in \mathbb{R}^n$ is still an n -dimensional vector and
 1768 not a scalar. However, we no longer require n coordinates to represent the
 1769 projection, but only a single one if we want to express it with respect to
 1770 the basis vector \mathbf{b} that spans the subspace U : λ . \diamond

Example 3.10 (Projection onto a Line)

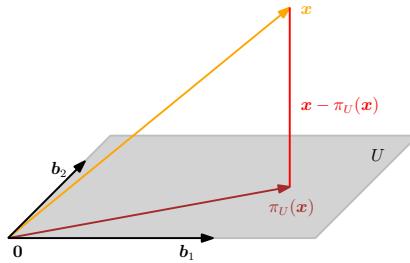
Find the projection matrix \mathbf{P}_π onto the line through the origin spanned by $\mathbf{b} = [1 \ 2 \ 2]^\top$. \mathbf{b} is a direction and a basis of the one-dimensional subspace (line through origin).

With (3.47), we obtain

$$\mathbf{P}_\pi = \frac{\mathbf{b} \mathbf{b}^\top}{\mathbf{b}^\top \mathbf{b}} = \frac{1}{9} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 2 & 2 \\ 2 & 4 & 4 \\ 2 & 4 & 4 \end{bmatrix}. \quad (3.48)$$

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

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



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

If U is given by a set of spanning vectors¹⁷⁷¹ which are not a basis, make sure you determine a basis $\mathbf{b}_1, \dots, \mathbf{b}_m$ ¹⁷⁷² before proceeding¹⁷⁷³

The basis vectors¹⁷⁷⁴ form the columns of $\mathbf{B} \in \mathbb{R}^{n \times m}$, where $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_m]$.¹⁷⁷⁵

spanned by \mathbf{b} . For $\mathbf{x} = [1 \ 1 \ 1]^\top$, the projection is

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

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

3.7.2 Projection onto General Subspaces

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

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

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

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

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

$$\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_m] \in \mathbb{R}^{n \times m}, \quad \boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_m]^\top \in \mathbb{R}^m, \quad (3.51)$$

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

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

$$\vdots \quad (3.53)$$

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

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

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

$$\vdots \quad (3.56)$$

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

such that we obtain a homogeneous linear equation system

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

$$\iff \mathbf{B}^\top \mathbf{B}\lambda = \mathbf{B}^\top \mathbf{x}. \quad (3.59)$$

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

$$\lambda = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{x}. \quad (3.60)$$

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

- 1781 2. Find the projection $\pi_U(\mathbf{x}) \in U$. We already established that $\pi_U(\mathbf{x}) =$
1782 $\mathbf{B}\lambda$. Therefore, with (3.60)

$$\pi_U(\mathbf{x}) = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{x}. \quad (3.61)$$

- 1783 3. Find the projection matrix \mathbf{P}_π . From (3.61) we can immediately see
1784 that the projection matrix that solves $\mathbf{P}_\pi \mathbf{x} = \pi_U(\mathbf{x})$ must be

$$\mathbf{P}_\pi = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top. \quad (3.62)$$

- 1785 4. *Remark.* Comparing the solutions for projecting onto a one-dimensional
1786 subspace and the general case, we see that the general case includes the
1787 1D case as a special case: If $\dim(U) = 1$ then $\mathbf{B}^\top \mathbf{B} \in \mathbb{R}$ is a scalar and
1788 we can rewrite the projection matrix in (3.62) $\mathbf{P}_\pi = \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top$ as
 $\mathbf{P}_\pi = \frac{\mathbf{B}\mathbf{B}^\top}{\mathbf{B}^\top \mathbf{B}}$, which is exactly the projection matrix in (3.47). \diamond

normal equation

pseudo-inverse

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

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 $\mathbf{x} = \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^3$ find the

coordinates λ of x in terms of the subspace U , the projection point $\pi_U(x)$ and the projection matrix P_π .

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

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

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

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

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

Fourth, the projection $\pi_U(x)$ of x onto U , i.e., into the column space of B , can be directly computed via

$$\pi_U(x) = B\lambda = \begin{bmatrix} 5 \\ 2 \\ -1 \end{bmatrix}. \quad (3.65)$$

projection error

The corresponding *projection error* is the norm of the difference vector between the original vector and its projection onto U , i.e.,

$$\|x - \pi_U(x)\| = \left\| \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}^\top \right\| = \sqrt{6}. \quad (3.66)$$

The projection error
is also called the
reconstruction error.

Fifth, the projection matrix (for any $x \in \mathbb{R}^3$) is given by

$$P_\pi = B(B^\top B)^{-1}B^\top = \frac{1}{6} \begin{bmatrix} 5 & 2 & -1 \\ 2 & 2 & 2 \\ -1 & 2 & 5 \end{bmatrix}. \quad (3.67)$$

To verify the results, we can (a) check whether the displacement vector $\pi_U(x) - x$ is orthogonal to all basis vectors of U , (b) verify that $P_\pi = P_\pi^2$ (see Definition 3.10).

1789 1790 1791 1792 *Remark.* The projections $\pi_U(x)$ 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 $\lambda_1, \dots, \lambda_m$ with respect to the basis vectors b_1, \dots, b_m of U . \diamond

1793 1794 1795 *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. \diamond

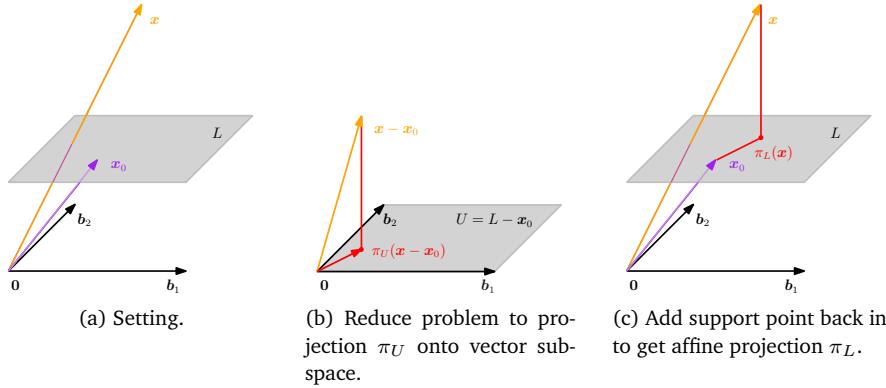


Figure 3.11
 Projection onto an affine space. (a) The original setting; (b) The setting is shifted by $-x_0$, so that $x - x_0$ can be projected onto the direction space U ; (c) The projection is translated back to $x_0 + \pi_U(x - x_0)$, which gives the final orthogonal projection $\pi_L(x)$.

1796 Projections allow us to look at situations where we have a linear system
 1797 $\mathbf{A}\mathbf{x} = \mathbf{b}$ without a solution. Recall that this means that \mathbf{b} does not lie in
 1798 the span of \mathbf{A} , i.e., the vector \mathbf{b} does not lie in the subspace spanned by
 1799 the columns of \mathbf{A} . Given that the linear equation cannot be solved exactly,
 1800 we can find an *approximate solution*. The idea is to find the vector in the
 1801 subspace spanned by the columns of \mathbf{A} that is closest to \mathbf{b} , i.e., we compute
 1802 the orthogonal projection of \mathbf{b} onto the subspace spanned by the columns
 1803 of \mathbf{A} . This problem arises often in practice, and the solution is called the
 1804 *least squares solution* (assuming the dot product as the inner product) of
 1805 an overdetermined system. This is discussed further in Chapter 9.

We can find approximate solutions to unsolvable linear equation systems using projections.

least squares solution

Remark. We just looked at projections of vectors \mathbf{x} onto a subspace U with basis vectors $\{\mathbf{b}_1, \dots, \mathbf{b}_k\}$. If this basis is an ONB, i.e., (3.33)–(3.34) are satisfied, the projection equation (3.61) simplifies greatly to

$$\pi_U(\mathbf{x}) = \mathbf{B}\mathbf{B}^\top \mathbf{x} \quad (3.68)$$

since $\mathbf{B}^\top \mathbf{B} = \mathbf{I}$ with coordinates

$$\lambda = \mathbf{B}^\top \mathbf{x}. \quad (3.69)$$

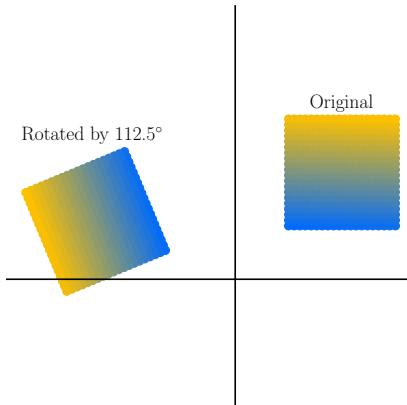
1806 This means that we no longer have to compute the tedious inverse from (3.61),
 1807 which saves us much computation time. \diamond

3.7.3 Projection onto Affine Subspaces

1809 Thus far, we discussed how to project a vector onto a lower-dimensional
 1810 subspace U . In the following, we provide a solution to projecting a vector
 1811 onto an affine subspace.

Consider the setting in Figure 3.11(a). We are given an affine space $L = x_0 + U$ where $\mathbf{b}_1, \mathbf{b}_2$ are basis vectors of U . To determine the orthogonal projection $\pi_L(\mathbf{x})$ of \mathbf{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 \mathbf{x}_0 from \mathbf{x} and from L , so that $L - \mathbf{x}_0 = U$ 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 $\pi_U(\mathbf{x} - \mathbf{x}_0)$, which is illustrated in Figure 3.11(b). This projection can now be translated back into L by adding \mathbf{x}_0 , such that we obtain the orthogonal projection onto an affine space L as

$$\pi_L(\mathbf{x}) = \mathbf{x}_0 + \pi_U(\mathbf{x} - \mathbf{x}_0), \quad (3.70)$$

1812 where $\pi_U(\cdot)$ is the orthogonal projection onto the subspace U , i.e., the
1813 direction space of L , see Figure 3.11(c).

From Figure 3.11 it is also evident that the distance of \mathbf{x} from the affine space L is identical to the distance of $\mathbf{x} - \mathbf{x}_0$ from U , i.e.,

$$d(\mathbf{x}, L) = \|\mathbf{x} - \pi_L(\mathbf{x})\| = \|\mathbf{x} - (\mathbf{x}_0 + \pi_U(\mathbf{x} - \mathbf{x}_0))\| \quad (3.71)$$

$$= d(\mathbf{x} - \mathbf{x}_0, \pi_U(\mathbf{x} - \mathbf{x}_0)). \quad (3.72)$$

1814

3.8 Rotations

1815 Length and angle preservation, as discussed in Section 3.4, are the two
1816 characteristics of linear mappings with orthogonal transformation matrices.
1817 In the following, we will have a closer look at specific orthogonal
1818 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

$$\mathbf{R} = \begin{bmatrix} -0.38 & -0.92 \\ 0.92 & -0.38 \end{bmatrix}. \quad (3.73)$$

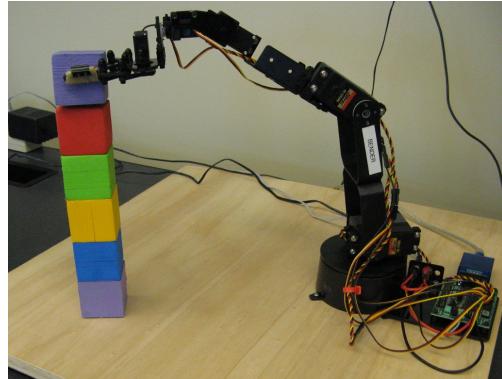


Figure 3.13 The robotic arm needs to rotate its joints in order to pick up objects or to place them correctly. Figure taken from (Deisenroth et al., 2015).

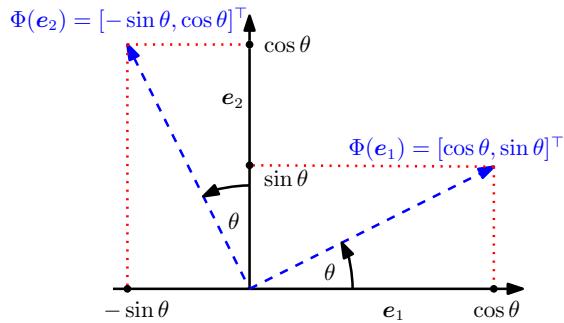


Figure 3.14 Rotation of the standard basis in \mathbb{R}^2 by an angle θ .

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 $\left\{e_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\}$ 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.

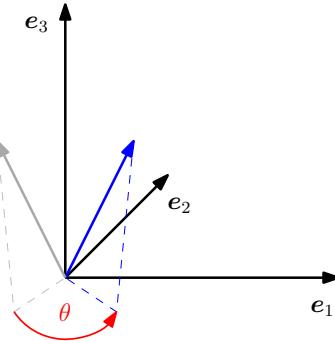
Rotations Φ are linear mappings so that we can express them by a *rotation matrix* $R(\theta)$. Trigonometry (see Figure 3.14) allows us to determine the coordinates of the rotated axes (the image of Φ) with respect to the standard basis in \mathbb{R}^2 . We obtain

$$\Phi(e_1) = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}, \quad \Phi(e_2) = \begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix}. \quad (3.74)$$

Therefore, the rotation matrix that performs the basis change into the

rotation matrix

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 $R(\theta)$ is given as

$$\mathbf{R}(\theta) = [\Phi(\mathbf{e}_1) \quad \Phi(\mathbf{e}_2)] = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}. \quad (3.75)$$

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 $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are supposed to be rotated, and making sure these images $\mathbf{R}\mathbf{e}_1, \mathbf{R}\mathbf{e}_2, \mathbf{R}\mathbf{e}_3$ are orthonormal to each other. We can then obtain a general rotation matrix \mathbf{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 \mathbf{e}_1 -axis

$$\mathbf{R}_1(\theta) = [\Phi(\mathbf{e}_1) \quad \Phi(\mathbf{e}_2) \quad \Phi(\mathbf{e}_3)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}. \quad (3.76)$$

Here, the \mathbf{e}_1 coordinate is fixed, and the counterclockwise rotation is performed in the $\mathbf{e}_2\mathbf{e}_3$ plane.

- Rotation about the \mathbf{e}_2 -axis

$$\mathbf{R}_2(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}. \quad (3.77)$$

If we rotate the $\mathbf{e}_1\mathbf{e}_3$ plane about the \mathbf{e}_2 axis, we need to look at the \mathbf{e}_2 axis from its “tip” toward the origin.

- Rotation about the e_3 -axis

$$\mathbf{R}_3(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.78)$$

1846 Figure 3.15 illustrates this.

1847 3.8.3 Rotations in n Dimensions

1848 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 V be an n -dimensional Euclidean vector space and $\Phi : V \rightarrow V$ an automorphism with transformation matrix

$$\mathbf{R}_{ij}(\theta) := \begin{bmatrix} \mathbf{I}_{i-1} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \cos \theta & \mathbf{0} & -\sin \theta & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{j-i-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sin \theta & \mathbf{0} & \cos \theta & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{I}_{n-j} \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad (3.79)$$

for $1 \leq i < j \leq n$ and $\theta \in \mathbb{R}$. Then $\mathbf{R}_{ij}(\theta)$ is called a *Givens rotation*. Givens rotation Essentially, $\mathbf{R}_{ij}(\theta)$ is the identity matrix \mathbf{I}_n with

$$r_{ii} = \cos \theta, \quad r_{ij} = -\sin \theta, \quad r_{ji} = \sin \theta, \quad r_{jj} = \cos \theta. \quad (3.80)$$

1853 In two dimensions (i.e., $n = 2$), we obtain (3.75) as a special case.

1854 3.8.4 Properties of Rotations

1855 Rotations exhibit a number useful properties:

- Rotations preserve distances, i.e., $\|\mathbf{x} - \mathbf{y}\| = \|\mathbf{R}_\theta(\mathbf{x}) - \mathbf{R}_\theta(\mathbf{y})\|$. In other words, rotations leave the distance between any two points unchanged after the transformation.
- Rotations preserve angles, i.e., the angle between $\mathbf{R}_\theta \mathbf{x}$ and $\mathbf{R}_\theta \mathbf{y}$ equals the angle between \mathbf{x} and \mathbf{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 $\mathbf{R}(\phi)\mathbf{R}(\theta) = \mathbf{R}(\theta)\mathbf{R}(\phi)$ 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).

1867

3.9 Further Reading

1868 In this chapter, we gave a brief overview of some of the important concepts
 1869 of analytic geometry, which we will use in later chapters of the book. For
 1870 a broader and more in-depth overview of some the concepts we presented
 1871 we refer to the following excellent books: Axler (2015) and Boyd and
 1872 Vandenberghe (2018).

1873 Inner products allow us to determine specific bases of vector (sub)spaces,
 1874 where each vector is orthogonal to all others (orthogonal bases) using the
 1875 Gram-Schmidt method. These bases are important in optimization and
 1876 numerical algorithms for solving linear equation systems. For instance,
 1877 Krylov subspace methods, such as Conjugate Gradients or GMRES, mini-
 1878 mize residual errors that are orthogonal to each other (Stoer and Burlirsch,
 1879 2002).

1880 In machine learning, inner products are important in the context of
 1881 kernel methods (Schölkopf and Smola, 2002). Kernel methods exploit the
 1882 fact that many linear algorithms can be expressed purely by inner prod-
 1883 uct computations. Then, the “kernel trick” allows us to compute these
 1884 inner products implicitly in a (potentially infinite-dimensional) feature
 1885 space, without even knowing this feature space explicitly. This allowed the
 1886 “non-linearization” of many algorithms used in machine learning, such as
 1887 kernel-PCA (Schölkopf et al., 1997) for dimensionality reduction. Gaus-
 1888 sian processes (Rasmussen and Williams, 2006) also fall into the category
 1889 of kernel methods and are the current state-of-the-art in probabilistic re-
 1890 gression (fitting curves to data points). The idea of kernels is explored
 1891 further in Chapter 12.

1892 Projections are often used in computer graphics, e.g., to generate shad-
 1893 ows. In optimization, orthogonal projections are often used to (iteratively)
 1894 minimize residual errors. This also has applications in machine learning,
 1895 e.g., in linear regression where we want to find a (linear) function that
 1896 minimizes the residual errors, i.e., the lengths of the orthogonal projec-
 1897 tions of the data onto the linear function (Bishop, 2006). We will investi-
 1898 giate this further in Chapter 9. PCA (Hotelling, 1933; Pearson, 1901b) also
 1899 uses projections to reduce the dimensionality of high-dimensional data.
 1900 We will discuss this in more detail in Chapter 10.

1901

Exercises

3.1 Show that $\langle \cdot, \cdot \rangle$ defined for all $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$ in \mathbb{R}^2 by:

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

1902

is an inner product.

3.2 Consider \mathbb{R}^2 with $\langle \cdot, \cdot \rangle$ defined for all x and y in \mathbb{R}^2 as:

$$\langle x, y \rangle := x^\top \underbrace{\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix}}_{=:A} y$$

1903 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 $x \in \mathbb{R}^5$ are given by

$$U = \text{span} \left[\begin{bmatrix} 0 \\ -1 \\ 2 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -3 \\ 1 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} -3 \\ 4 \\ 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -3 \\ 5 \\ 0 \\ 7 \end{bmatrix} \right], \quad x = \begin{bmatrix} -1 \\ -9 \\ -1 \\ 4 \\ 1 \end{bmatrix}$$

1904 1. Determine the orthogonal projection $\pi_U(x)$ of x onto U

1905 2. Determine the distance $d(x, U)$

3.4 Consider \mathbb{R}^3 with the inner product

$$\langle x, y \rangle := x^\top \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} y.$$

1906 Furthermore, we define e_1, e_2, e_3 as the standard/canonical basis in \mathbb{R}^3 .

1. Determine the orthogonal projection $\pi_U(e_2)$ of e_2 onto

$$U = \text{span}[e_1, e_3].$$

1907 Hint: Orthogonality is defined through the inner product.

1908 2. Compute the distance $d(e_2, U)$.

1909 3. Draw the scenario: standard basis vectors and $\pi_U(e_2)$

1910 3.5 Prove the Cauchy-Schwarz inequality $|\langle x, y \rangle| \leq \|x\| \|y\|$ for $x, y \in V$, where
1911 V is a vector space.

4

1921

Matrix Decompositions

1922 In Chapters 2 and 3, we studied ways to manipulate and measure vectors,
1923 projections of vectors and linear mappings. Mappings and transformations
1924 of vectors can be conveniently described as operations performed on ma-
1925 trices. Moreover, data is often represented in matrix form as well, for ex-
1926 ample where the rows of the matrix represent different instances of the
1927 data (for example people) and the columns describe different features of
1928 the data (for example weight, height and socio-economic status). In this
1929 chapter we present three aspects of matrices: how to summarize matrices,
1930 how matrices can be decomposed, and how these decompositions can be
1931 used to consider matrix approximations.

1932 We first consider methods that allow us to describe matrices with just
1933 a few numbers that characterize the overall properties of matrices. We
1934 will do this in the sections on determinants (Section 4.1) and eigenvalues
1935 (Section 4.2 for the important special case of square matrices. These char-
1936 acteristic numbers have important mathematical consequences and allow
1937 us to quickly grasp what useful properties a matrix has. From here we will
1938 proceed to matrix decomposition methods: An analogy for matrix decom-
1939 position is the factoring of numbers, such as the factoring of 21 into prime
1940 numbers 7×3 . For this reason matrix decomposition is also often referred
matrix factorization 1941 to as *matrix factorization*. Matrix decompositions are used to interpret a
1942 matrix using a different representation using factors of interpretable ma-
1943 trices.

1944 We will first cover a square-root-like operation for matrices called Cholesky
1945 decomposition (Section 4.3) for symmetric, positive definite matrices. From
1946 here we will look at two related methods for factorizing matrices into
1947 canonical forms. The first one is known as matrix diagonalization (Sec-
1948 tion 4.4), which allows us to represent the linear mapping using a diag-
1949 onal transformation matrix if we choose an appropriate basis. The second
1950 method, singular value decomposition (Section 4.5), extends this factor-
1951 ization to non-square matrices, and it is considered one of the fundamen-
1952 tal concepts in linear algebra. These decomposition are helpful as matrices
1953 representing numerical data are often very large and hard to analyze. We
1954 conclude the chapter with a systematic overview of the types of matrices
1955 and the characteristic properties that distinguish them in form of a matrix
1956 taxonomy (Section 4.7).

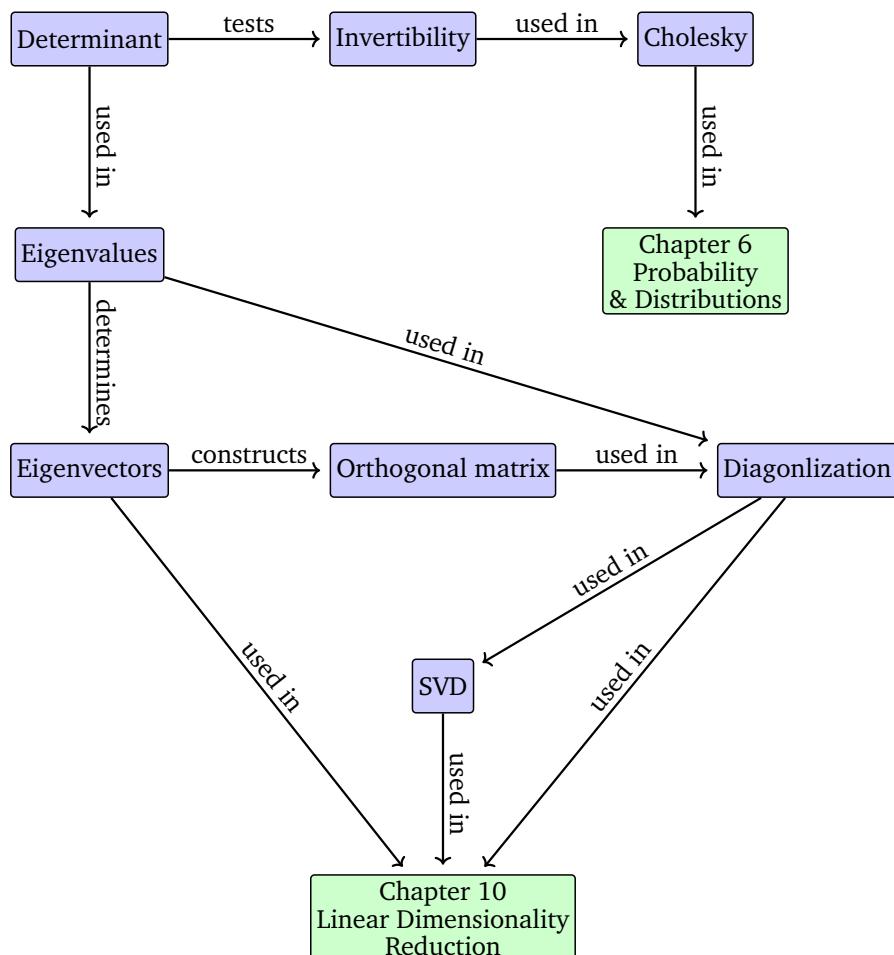


Figure 4.1 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

1957 The methods that we cover in this chapter will become important in
 1958 both subsequent mathematical chapters, such as Chapter 6 but also in ap-
 1959 plied chapters, such as dimensionality reduction in Chapters 10 or density
 1960 estimation in Chapter 11. This chapter's overall structure is depicted in
 1961 the mind map of Figure 4.1.

1962

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 $\mathbf{A} \in \mathbb{R}^{n \times n}$, that is matrices with the same number of rows and columns. In this book we write this as $\det(\mathbf{A})$ (some textbooks may use $|\mathbf{A}|$, 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 \mathbf{A} . Then we write

$$\det(\mathbf{A}) = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}. \quad (4.1)$$

determinant 1963 The *determinant* of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a function that maps
 1964 \mathbf{A} onto a real number. Before provide a definition of the determinant for
 1965 general $n \times n$ matrices let us look at some motivating examples, and define
 1966 determinants for some special matrices.

Example 4.1 (Testing for Matrix Invertibility)

Let us begin with exploring if a square matrix \mathbf{A} is invertible (see Section 2.2.2). For the smallest cases, we already know when a matrix is invertible. If \mathbf{A} is a 1×1 matrix, i.e., it is a scalar number, then $\mathbf{A} = a \implies \mathbf{A}^{-1} = \frac{1}{a}$. 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 $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ and thus we can write that the inverse of \mathbf{A}^{-1} is (from Equation 2.24)

$$\mathbf{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}. \quad (4.2)$$

Thus, \mathbf{A} is invertible if and only if

$$a_{11}a_{22} - a_{12}a_{21} \neq 0. \quad (4.3)$$

This quantity is the determinant of $\mathbf{A} \in \mathbb{R}^{2 \times 2}$, that is

$$\det(\mathbf{A}) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \quad (4.4)$$

1967 The example above points already at the relationship between determinants
 1968 and the existence of inverse matrices. The next theorem states the
 1969 same result for $n \times n$ matrices.

1970 **Theorem 4.1.** *For any square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ it holds that \mathbf{A} is invertible
 1971 if and only if $\det(\mathbf{A}) \neq 0$.*

We have explicit (closed form) expressions for determinants of small matrices in terms of the elements of the matrix. For $n = 1$,

$$\det(\mathbf{A}) = \det(a_{11}) = a_{11}. \quad (4.5)$$

For $n = 2$,

$$\det(\mathbf{A}) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}, \quad (4.6)$$

which we have observed in the example above. For $n = 3$ (known as Sarrus' rule),

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23} - a_{31}a_{22}a_{13} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33}. \quad (4.7)$$

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 \mathbf{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 \mathbf{A} , the determinant is the product of the diagonal elements:

$$\det(\mathbf{A}) = \prod_{i=1}^n a_{ii}. \quad (4.8)$$

upper triangular matrix
lower triangular matrix

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 \mathbf{A} .

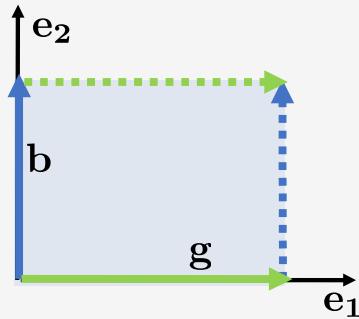


Figure 4.2
Determinants can measure areas spanned by vectors. The area A of the parallelogram (shaded region) spanned by the vectors \mathbf{b} and \mathbf{g} is given by the determinant $\det([\mathbf{b}, \mathbf{g}])$.

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 \mathbf{b}, \mathbf{g} that form the columns of a matrix $\mathbf{A} = [\mathbf{b}, \mathbf{g}]$. Then, the absolute value of the determinant of \mathbf{A} is the area of the parallelogram with vertices $0, \mathbf{b}, \mathbf{g}, \mathbf{b} + \mathbf{g}$. In particular, if the two vectors \mathbf{b}, \mathbf{g} were linearly dependent so that $\mathbf{b} = \lambda\mathbf{g}$ for some $\lambda \in \mathbb{R}$ they no longer form a two-dimensional parallelogram. Therefore, the corresponding area is 0. On the contrary, if \mathbf{b}, \mathbf{g} were al and lie along the canonical coordinate axes e_1, e_2 then they

would reduce to $\mathbf{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}$ and $\mathbf{g} = \begin{bmatrix} 0 \\ g \end{bmatrix}$ and the determinant

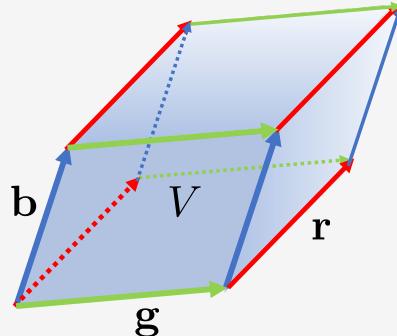
$$\begin{vmatrix} b & 0 \\ 0 & g \end{vmatrix} = bg - 0 = bg \quad (4.9)$$

becomes the familiar formula: area = height \times length.

The sign of the determinant measures the orientation of the spanning vectors \mathbf{b}, \mathbf{g} with respect to the standard coordinate system e_1, e_2 . In our figure, flipping the spanning order to \mathbf{g}, \mathbf{b} swaps the columns of A and reverses the orientation of the shaded surface A .

This intuition extends to higher dimensions. In \mathbb{R}^3 , we consider three vectors $\mathbf{r}, \mathbf{b}, \mathbf{g} \in \mathbb{R}^3$ 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 $[\mathbf{r}, \mathbf{b}, \mathbf{g}]$ 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 $\mathbf{r}, \mathbf{b}, \mathbf{g}$ is given by the determinant $\det([\mathbf{r}, \mathbf{b}, \mathbf{g}])$.



Consider the three linearly independent vectors $\mathbf{r}, \mathbf{g}, \mathbf{b} \in \mathbb{R}^3$ given as

$$\mathbf{r} = \begin{bmatrix} 2 \\ 0 \\ -8 \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} 6 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 4 \\ -1 \end{bmatrix}. \quad (4.10)$$

$$\mathbf{A} = [\mathbf{r}, \mathbf{g}, \mathbf{b}] = \begin{bmatrix} 2 & 6 & 1 \\ 0 & 1 & 4 \\ -8 & 0 & -1 \end{bmatrix}. \quad (4.11)$$

Therefore, the volume is given as

$$V = |\det(\mathbf{A})| = 186. \quad (4.12)$$

1972

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-

1973

1974

1975 minant of an $n \times n$ matrix to computing the determinant of $(n-1) \times (n-1)$
 1976 matrices. By recursively applying the Laplace expansion we can therefore
 1977 compute determinants of $n \times n$ matrices by ultimately computing deter-
 1978 minants of 2×2 matrices.

1979 **Theorem 4.2** (Laplace Expansion). *Consider a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. Then,
 1980 for all $j = 1, \dots, n$:*

1 *Expansion along column j*

$$\det(\mathbf{A}) = \sum_{k=1}^n (-1)^{k+j} a_{kj} \det(\mathbf{A}_{k,j}). \quad (4.13)$$

$\det(\mathbf{A}_{k,j})$ is called
 a *minor* and
 $(-1)^{k+j} \det(\mathbf{A}_{k,j})$
 a *cofactor*.

2 *Expansion along row j*

$$\det(\mathbf{A}) = \sum_{k=1}^n (-1)^{k+j} a_{jk} \det(\mathbf{A}_{j,k}). \quad (4.14)$$

1981 Here $\mathbf{A}_{k,j} \in \mathbb{R}^{(n-1) \times (n-1)}$ is the submatrix of \mathbf{A} that we obtain when delet-
 1982 ing row k and column j .

Example 4.3 (Laplace Expansion)

Let us compute the determinant of

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.15)$$

using the Laplace expansion along the first row. By applying (4.14) we obtain

$$\begin{vmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 0 & 0 & 1 \end{vmatrix} = (-1)^{1+1} \cdot 1 \begin{vmatrix} 1 & 2 \\ 0 & 1 \end{vmatrix} + (-1)^{1+2} \cdot 2 \begin{vmatrix} 3 & 2 \\ 0 & 1 \end{vmatrix} + (-1)^{1+3} \cdot 3 \begin{vmatrix} 3 & 1 \\ 0 & 0 \end{vmatrix}. \quad (4.16)$$

Then we can use (4.6) to compute the determinants of all 2×2 matrices and obtain.

$$\det(\mathbf{A}) = 1(1 - 0) - 2(3 - 0) + 3(0 - 0) = -5.$$

For completeness we can compare this result to computing the determinant using Sarrus' rule (4.7):

$$\det(\mathbf{A}) = 1 \cdot 1 \cdot 1 + 3 \cdot 0 \cdot 3 + 0 \cdot 2 \cdot 2 - 0 \cdot 1 \cdot 3 - 1 \cdot 0 \cdot 2 - 3 \cdot 2 \cdot 1 = 1 - 6 = -5. \quad (4.17)$$

1983 For $\mathbf{A} \in \mathbb{R}^{n \times n}$ the determinant exhibits the following properties:

- 1984 • The determinant of a product is the product of the determinant, $\det(\mathbf{AB}) =$
 1985 $\det(\mathbf{A})\det(\mathbf{B})$.

- 1986 • Determinants are invariant to transposition $\det(\mathbf{A}) = \det(\mathbf{A}^\top)$.
- 1987 • If \mathbf{A} is regular (Section 2.2.2) then $\det(\mathbf{A}^{-1}) = \frac{1}{\det(\mathbf{A})}$
- 1988 • Similar matrices (Defintion 2.21) possess the same determinant. There-
1989 fore, for a linear mapping $\Phi : V \rightarrow V$ all transformation matrices \mathbf{A}_Φ
1990 of Φ have the same determinant. Thus, the determinant is invariant to
1991 the choice of basis of a linear mapping.
- 1992 • Adding a multiple of a column/row to another one does not change
1993 $\det(\mathbf{A})$.
- 1994 • Multiplication of a column/row with $\lambda \in \mathbb{R}$ scales $\det(\mathbf{A})$ by λ . In
1995 particular, $\det(\lambda\mathbf{A}) = \lambda^n \det(\mathbf{A})$.
- 1996 • Swapping two rows/columns changes the sign of $\det(\mathbf{A})$.

1997 Because of the last three properties, we can use Gaussian elimination (see
1998 Section 2.1) to compute $\det(\mathbf{A})$ by bringing \mathbf{A} into row-echelon form. We
1999 can stop Gaussian elimination when we have \mathbf{A} in a triangular form where
2000 the elements below the diagonal are all 0. Recall from Equation (4.8) that
2001 the determinant is then the product of the diagonal elements.

2002 **Theorem 4.3.** A square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ has $\det(\mathbf{A}) \neq 0$ if and only if
2003 $\text{rk } \mathbf{A} = n$. In other words a square matrix is invertible if and only if it is full
2004 rank.

2005 When mathematics was mainly performed by hand, the determinant
2006 calculation was considered an essential way to analyze matrix invertibil-
2007 ity. However, contemporary approaches in machine learning use direct
2008 numerical methods that superseded the explicit calculation of the deter-
2009 minant. For example, in Chapter 2 we learned that inverse matrices can
2010 be computed by Gaussian elimination. Gaussian elimination can thus be
2011 used to compute the determinant of a matrix.

2012 Determinants will play an important theoretical role for the following
2013 sections, especially when we learn about eigenvalues and eigenvectors
2014 (Section 4.2) through the characteristic polynomial of a matrix.

trace

Definition 4.4. The trace of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a linear function
denoted by $\text{tr}(\mathbf{A})$ and defined as

$$\text{tr}(\mathbf{A}) := \sum_{i=1}^n a_{ii}, \quad (4.18)$$

2015 in other words, the trace is the sum of the diagonal elements of \mathbf{A} .

2016 *Remark.* For $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ the trace satisfies the following properties:

- 2017 1 $\text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B})$
- 2018 2 $\text{tr}(\alpha\mathbf{A}) = \alpha\text{tr}(\mathbf{A}), \quad \alpha \in \mathbb{R}$
- 2019 3 $\text{tr}(\mathbf{I}_n) = n$
- 2020 4 $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$

2021 It can be shown that only one function satisfies these four properties together – the trace (Gohberg et al., 2012). \diamond

2023 *Remark.* The properties of the trace of matrix products are more general:

- The trace is invariant under *cyclic permutations*, i.e.,

cyclic permutations

$$\text{tr}(\mathbf{A}\mathbf{K}\mathbf{L}) = \text{tr}(\mathbf{K}\mathbf{L}\mathbf{A}) \quad (4.19)$$

2024 for matrices $\mathbf{A} \in \mathbb{R}^{a \times k}$, $\mathbf{K} \in \mathbb{R}^{l \times l}$, $\mathbf{L} \in \mathbb{R}^{l \times a}$. 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 $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$:

$$\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}). \quad (4.20)$$

In particular, this means that for two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$\text{tr}(\mathbf{xy}^\top) = \text{tr}(\mathbf{y}^\top \mathbf{x}) = \mathbf{y}^\top \mathbf{x} \in \mathbb{R}. \quad (4.21)$$

\diamond

Remark. Given some linear map $\Phi : V \rightarrow V$, 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 \mathbf{A} relative to this basis, and taking the trace of this square matrix. Assume that \mathbf{B} is transformation matrix between bases of V . Then, we can write

$$\text{tr}(\mathbf{BAB}^{-1}) = \text{tr}(\mathbf{B}^{-1}\mathbf{BA}) = \text{tr}(\mathbf{IA}) = \text{tr}(\mathbf{A}). \quad (4.22)$$

2027 Thus, while matrix representations of linear mappings are basis dependent 2028 its trace is independent of the basis. \diamond

2029 The trace is useful in certain classes of machine learning models where 2030 data is fitted using linear regression models. The trace captures model 2031 complexity in these models and can be used to compare between models 2032 (a more principled foundation for model comparison is discussed in detail 2033 in Section 8.5).

2034 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 \mathbf{A} in terms of a polynomial, which we will use extensively in the 2035 following sections.

Definition 4.5 (Characteristic Polynomial). For $\lambda \in \mathbb{R}$ and a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$

$$p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) \quad (4.23)$$

$$= c_0 + c_1\lambda + c_2\lambda^2 + \cdots + c_{n-1}\lambda^{n-1} + (-1)^n\lambda^n, \quad (4.24)$$

$c_0, \dots, c_{n-1} \in \mathbb{R}$, is the *characteristic polynomial* of \mathbf{A} . In particular,

characteristic polynomial

$$c_0 = \det(\mathbf{A}), \quad (4.25)$$

$$c_{n-1} = (-1)^{n-1}\text{tr}(\mathbf{A}). \quad (4.26)$$

2039 The characteristic polynomial will allow us to compute eigenvalues and
 2040 eigenvectors, covered in the next section.

2041 **4.2 Eigenvalues and Eigenvectors**

2042 We will now get to know a new way to characterize a matrix and, its as-
 2043 sociated linear mapping. Let us recall from Section 2.7.1 that every linear
 2044 mapping has a unique transformation matrix given an ordered basis. We
 2045 can interpret linear mappings and their associated transformation matri-
 2046 ces by performing an “Eigen” analysis. *Eigen* is a German word meaning
 2047 “characteristic”, “self” or “own”. As we will see the eigenvalues of a lin-
 2048 ear mapping will tell us how a special set of vectors, the eigenvectors, are
 2049 transformed by the linear mapping.

Definition 4.6. Let $A \in \mathbb{R}^{n \times n}$ be a square matrix. Then $\lambda \in \mathbb{R}$ is an *eigenvalue* of A and a nonzero $x \in \mathbb{R}^n$ is the corresponding *eigenvector* of A if

$$Ax = \lambda x. \quad (4.27)$$

eigenvalue
eigenvector

eigenvalue equation²⁰⁵⁰ We call this the *eigenvalue equation*.

2051 *Remark.* In linear algebra literature and software, it is often a conven-
 2052 tion that eigenvalues are sorted in descending order, so that the largest
 2053 eigenvalue and associated eigenvector are called the first eigenvalue and
 2054 its associated eigenvector, and the second largest called the second eigen-
 2055 value and its associated eigenvector, and so on. However textbooks and
 2056 publications may have different or no notion of orderings. We do not want
 2057 to presume an ordering in our book. ◇

codirected
collinear

2058 **Definition 4.7** (Collinearity & Codirection). Two vectors that point in the
 2059 same direction are called *codirected*. Two vectors are *collinear* if they point
 2060 in the same or the opposite direction.

Remark (Non-uniqueness of Eigenvectors). If x is an eigenvector of A associated with eigenvalue λ then for any $c \in \mathbb{R} \setminus \{0\}$ it holds that cx is an eigenvector of A with the same eigenvalue since

$$A(cx) = cAx = c\lambda x = \lambda(cx). \quad (4.28)$$

2061 Thus, all vectors that are collinear to x are also eigenvectors of A . ◇

eigenspace

2063 **Theorem 4.8.** $\lambda \in \mathbb{R}$ is eigenvalue of $A \in \mathbb{R}^{n \times n}$ if and only if λ is a root of
 2064 the characteristic polynomial $p_A(\lambda)$ of A .

2065 **Definition 4.9** (Eigenspace and Eigenspectrum). For $A \in \mathbb{R}^{n \times n}$ the set
 2066 of all eigenvectors of A associated with an eigenvalue λ spans a subspace
 2067 of \mathbb{R}^n , which is called *eigenspace* of A with respect to λ and is denoted

by E_λ . The set of all eigenvalues of \mathbf{A} is called the *eigenspectrum*, or just spectrum, of \mathbf{A} .

There are a number of ways to think about these characteristics

- The eigenvector is a special vector that, left multiplying with the matrix \mathbf{A} merely stretches the vector by a factor – the eigenvalue.
- Recall the definition of the kernel from Section 2.7.3, it follows that $E_\lambda = \ker(\mathbf{A} - \lambda\mathbf{I})$ since

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \iff \mathbf{A}\mathbf{x} - \lambda\mathbf{x} = \mathbf{0} \quad (4.29)$$

$$\iff (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0} \iff \mathbf{x} \in \ker(\mathbf{A} - \lambda\mathbf{I}). \quad (4.30)$$

- Similar matrices (see Definition 2.21) possess the same eigenvalues. Therefore, a linear mapping Φ has eigenvalues that are independent from the choice of basis of its transformation matrix. This makes eigenvalues, together with the determinant and the trace, the key characteristic parameters of a linear mapping as they are all invariant under basis 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.

$$\mathbf{A} = \begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix}. \quad (4.31)$$

Step 1: Characteristic Polynomial

From our definition of the eigenvector \mathbf{x} and eigenvalue λ for \mathbf{A} there will be a vector such that $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, i.e., $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$. Since $\mathbf{x} \neq \mathbf{0}$ by definition of the eigenvectors, this condition requires that the kernel (nullspace) of $\mathbf{A} - \lambda\mathbf{I}$ contains more elements than just $\mathbf{0}$. This means that $\mathbf{A} - \lambda\mathbf{I}$ is not invertible and therefore $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. Hence we need to compute the roots of the characteristic polynomial (Equation (4.23)).

Step 2: Eigenvalues

The characteristic polynomial is given as

$$p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) = \det \left(\begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right) = \begin{vmatrix} 4 - \lambda & 2 \\ 1 & 3 - \lambda \end{vmatrix} \quad (4.32)$$

$$= (4 - \lambda)(3 - \lambda) - 2 \cdot 1. \quad (4.33)$$

We factorize the characteristic polynomial

$$p(\lambda) = (4 - \lambda)(3 - \lambda) - 2 \cdot 1 = 10 - 7\lambda + \lambda^2 = (2 - \lambda)(5 - \lambda) \quad (4.34)$$

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 \mathbf{x} such that

$$\begin{bmatrix} 4 - \lambda & 2 \\ 1 & 3 - \lambda \end{bmatrix} \mathbf{x} = \mathbf{0}. \quad (4.35)$$

For $\lambda = 5$ we obtain

$$\begin{bmatrix} 4 - 5 & 2 \\ 1 & 3 - 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{0}. \quad (4.36)$$

We now solve this homogeneous equation system and obtain a solution space

$$E_5 = \text{span} \left[\begin{bmatrix} 2 \\ 1 \end{bmatrix} \right], \quad (4.37)$$

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

$$\begin{bmatrix} 4 - 2 & 2 \\ 1 & 3 - 2 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \mathbf{x} = \mathbf{0}. \quad (4.38)$$

This means any vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 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

$$E_2 = \text{span} \left[\begin{bmatrix} 1 \\ -1 \end{bmatrix} \right]. \quad (4.39)$$

2079 2080 2081 2082 2083 2084 2085 2086

Remark (Eigenvalues and Eigenspaces). If λ is an eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$ then the corresponding eigenspace E_λ is the solution space of the homogeneous linear equation system $(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = \mathbf{0}$. 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 \mathbf{A} . \diamond

2087 *Remark.* The following statements are equivalent:

- 2088 • λ is eigenvalue of $\mathbf{A} \in \mathbb{R}^{n \times n}$
- 2089 • There exists an $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ with $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ or equivalently, $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{x} = \mathbf{0}$ can be solved non-trivially, i.e., $\mathbf{x} \neq \mathbf{0}$.

- 2091 • $\text{rk}(\mathbf{A} - \lambda \mathbf{I}_n) < n$
 2092 • $\det(\mathbf{A} - \lambda \mathbf{I}_n) = 0$

◊

2093
 2094 Useful properties regarding eigenvalues and eigenvectors of various ma-
 2095 trix types include

- 2096 • A matrix \mathbf{A} and its transpose \mathbf{A}^\top possess the same eigenvalues, but not
 2097 necessarily the same eigenvectors.
 2098 • Symmetric matrices always have real-valued eigenvalues.
 2099 • Symmetric positive definite matrices always have positive, real eigen-
 2100 values.
 2101 • The eigenvectors of symmetric matrices are always orthogonal to each
 2102 other.

Theorem 4.10. *Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ we can always obtain a \mathbf{S} that is a symmetric positive semi-definite matrix by computing*

$$\mathbf{S} = \mathbf{A}^\top \mathbf{A}. \quad (4.40)$$

2103 Understanding why this theorem holds is insightful for how we can
 2104 use symmetrised matrices: Symmetry requires $\mathbf{S} = \mathbf{S}^\top$ and by inserting
 2105 (4.40) we obtain $\mathbf{S} = \mathbf{A}^\top \mathbf{A} = \mathbf{A}^\top (\mathbf{A}^\top)^\top = (\mathbf{A}^\top \mathbf{A})^\top = \mathbf{S}^\top$. More-
 2106 over, positive semi-definiteness (Section 3.2.3) requires that $\mathbf{x}^\top \mathbf{S} \mathbf{x} \geq 0$
 2107 and inserting (4.40) we obtain $\mathbf{x}^\top \mathbf{S} \mathbf{x} = \mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{x} = (\mathbf{x}^\top \mathbf{A}^\top)(\mathbf{A} \mathbf{x}) =$
 2108 $(\mathbf{A} \mathbf{x})^\top (\mathbf{A} \mathbf{x}) \geq 0$, because the scalar product computes a sum of squares
 2109 (which are themselves always positive or zero).

2110 **Theorem 4.11** (Hogben (2006)). *Consider a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with distinct eigenvalues $\lambda_1, \dots, \lambda_n$ and corresponding eigenvectors $\mathbf{x}_1, \dots, \mathbf{x}_n$. Then the eigenvectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ are linearly independent.*

2113 The theorem states that eigenvectors belonging to different eigenvalues
 2114 form a linearly independent set. For symmetric matrices we can state a
 2115 stronger version of Theorem 4.11.

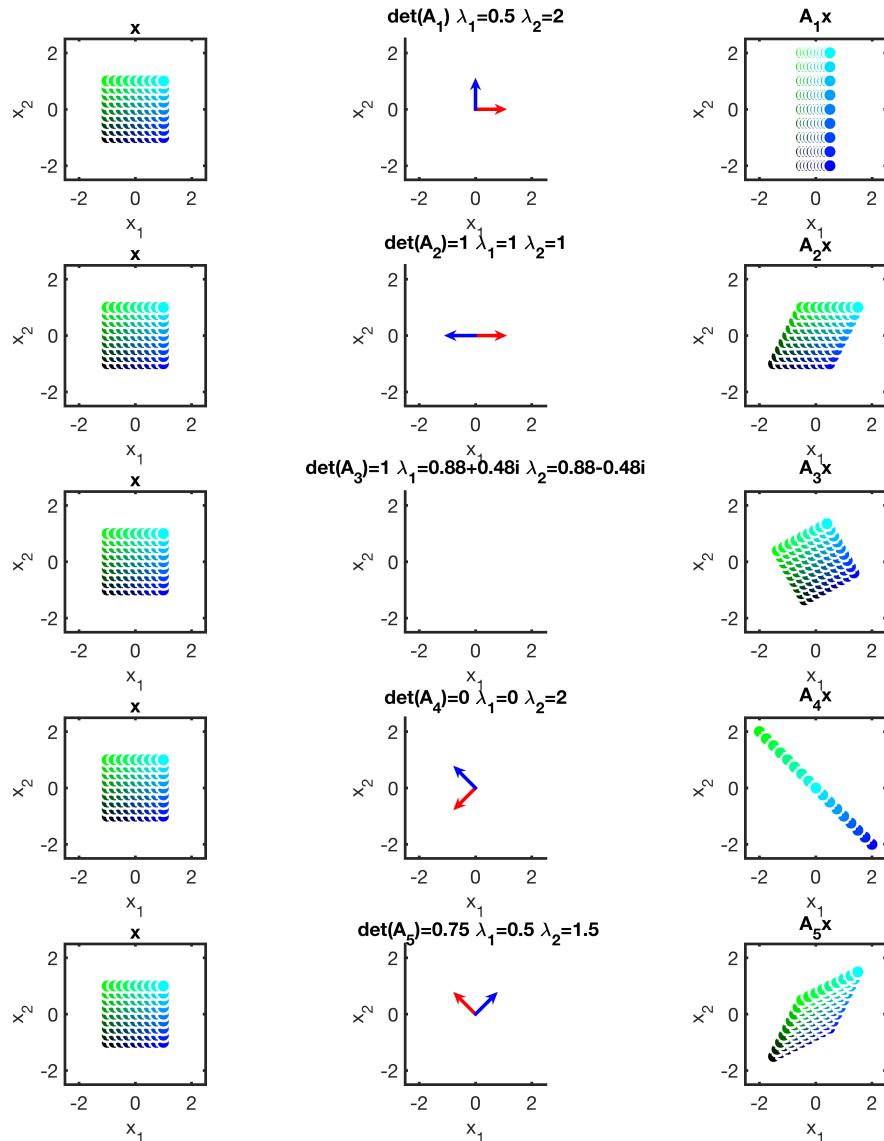
2116 **Theorem 4.12** (Meyer (2000)). *Any symmetric matrix $\mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^{n \times n}$ has n independent eigenvectors that form an orthogonal basis for \mathbb{R}^n .*

2118 Graphical Intuition in Two Dimensions

2119 Let us gain some intuition for determinants, eigenvectors, eigenvalues and
 2120 how linear maps affect space. Figure 4.4 depicts five transformation matri-
 2121 ces and their impact on a square grid of points. The square grid of points
 2122 are contained within a box of dimensions 2×2 with its centre at the origin.

- 2123 • $\mathbf{A}_1 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 2 \end{bmatrix}$. The direction of the two eigenvectors correspond to the
 2124 canonical basis vectors in \mathbb{R}^2 , i.e. to two cardinal axes. The horizontal
 2125 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
 $A_i \in \mathbb{R}^{2 \times 2}$ project 81 color-coded points $x \in \mathbb{R}^2$ (left column of plots) to target points $A_i x$ (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
 $A_i, i = \{1, \dots, 5\}$.



is extended by a factor of 2 (eigenvalue $\lambda_2 = 2$). The mapping is area preserving ($\det(A_1) = 1 = 2 \times \frac{1}{2}$). Note, that while the area covered by the box of points remained the same, the circumference around the box has increased by 20%.

- $A_2 = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$ 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 ($\det(A_2) = 1$). 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 while the mapping is area preserving the circumference around the box of points has increased.

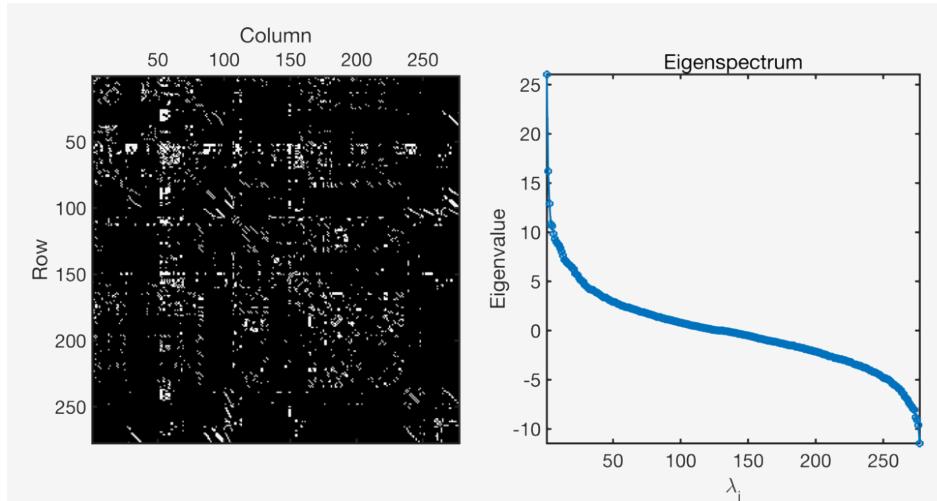
- $A_3 = \begin{bmatrix} \cos(\frac{\pi}{6}) & -\sin(\frac{\pi}{6}) \\ \sin(\frac{\pi}{6}) & \cos(\frac{\pi}{6}) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sqrt{3} & -1 \\ 1 & \sqrt{3} \end{bmatrix}$ The rotation matrix A_3 rotates the points by $\frac{\pi}{6}$ (or 30° degrees) anti-clockwise, and 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.

- $A_4 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ 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%.

- $A_5 = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}$ is a shear-and-stretch mapping that shrinks space by 75% ($|\det(A_5)| = \frac{3}{4}$), 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 \mathbf{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 $\mathbf{A}_{sym} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^\top)$. This new matrix \mathbf{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 \mathbf{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

2162 **Definition 4.13.** Let a square matrix \mathbf{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

2165 **Definition 4.14.** Let a square matrix \mathbf{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 .

2169 2170 2171 2172 *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. \diamond

Example 4.6

The matrix $A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$ has two repeated eigenvalues $\lambda_1 = \lambda_2 = 2$ and an algebraic multiplicity of 2. The eigenvalue has however only one distinct eigenvector $x_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and thus geometric multiplicity 1.

2173 2174 2175 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 $A \in \mathbb{R}^{n \times n}$ is the product of its eigenvalues, i.e.,*

$$\det(A) = \prod_{i=1}^n \lambda_i, \quad (4.41)$$

2176 where λ_i are (possibly repeated) eigenvalues of A .

Theorem 4.16. *The trace of a matrix $A \in \mathbb{R}^{n \times n}$ is the sum of its eigenvalues, i.e.,*

$$\text{tr}(A) = \sum_{i=1}^n \lambda_i, \quad (4.42)$$

2177 where λ_i are (possibly repeated) eigenvalues of A .

2178 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.

2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 *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 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

2193 length of the sides of the box. The trace of \mathbf{A} captures therefore a notion
2194 of how the matrix acts on the circumference of a volume. \diamond

Example 4.7 (Google's PageRank – Webpages as Eigenvectors)

Google uses the eigenvector corresponding to the maximal eigenvalue of a matrix \mathbf{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 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) $x_i \geq 0$ of a website a_i by counting the number of pages pointing to a_i . PageRank also takes 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 \mathbf{A} of this graph that tells us with what (click) probability somebody will end up on a different website. The matrix \mathbf{A} has the property that for any initial rank/importance vector \mathbf{x} of a website the sequence $\mathbf{x}, \mathbf{Ax}, \mathbf{A}^2\mathbf{x}, \dots$ converges to a vector \mathbf{x}^* . This vector is called the *PageRank* and satisfies $\mathbf{Ax}^* = \mathbf{x}^*$, i.e., it is an eigenvector (with corresponding eigenvalue 1) of \mathbf{A} . After normalizing by \mathbf{x}^* , such that $\|\mathbf{x}^*\| = 1$ 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

2195

4.3 Cholesky Decomposition

2196 There are many ways to factorize special types of matrices that we encounter often in machine learning. In the positive real numbers we have 2197 the square-root operation that yields us a decomposition of the number 2198 into components, for example, $9 = 3 \cdot 3$. For matrices, we need to be 2199 careful that we compute a square-root like operation on positive quantities. For symmetric, positive definite matrices (see Section 3.2.3) we can 2200 choose from a number of square-root equivalent operations. The *Cholesky* 2201 decomposition or *Cholesky factorization* provides a square-root equivalent 2202 operations that is very useful.

2203

2204 **Theorem 4.17.** *Cholesky Decomposition:* A symmetric positive definite matrix \mathbf{A} can be factorized into a product $\mathbf{A} = \mathbf{LL}^\top$, where \mathbf{L} is a lower triangular matrix with positive diagonal elements:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & \cdots & l_{n1} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & l_{nn} \end{bmatrix}. \quad (4.43)$$

Cholesky decomposition
Cholesky factorization

Cholesky factor 2205 \mathbf{L} is called the *Cholesky factor* of \mathbf{A} .

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.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \equiv \mathbf{L}\mathbf{L}^\top = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} \quad (4.44)$$

Expanding the right hand side yields

$$\mathbf{A} = \begin{bmatrix} l_{11}^2 & l_{21}l_{11} & l_{31}l_{11} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{31}l_{21} + l_{32}l_{22} \\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix}.$$

Comparing the left hand side and the right hand side shows that there is a simple pattern in the diagonal elements (l_{ii}):

$$l_{11} = \sqrt{a_{11}}, \quad l_{22} = \sqrt{a_{22} - l_{21}^2}, \quad l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)}. \quad (4.45)$$

Similarly for the elements below the diagonal (l_{ij} , where $i > j$) there is also a repeating pattern:

$$l_{21} = \frac{1}{l_{11}}a_{21}, \quad l_{31} = \frac{1}{l_{11}}a_{31}, \quad l_{32} = \frac{1}{l_{22}}(a_{32} - l_{31}l_{21}). \quad (4.46)$$

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 \mathbf{L} should be, given the values a_{ij} for \mathbf{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 6.5, 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 \mathbf{L}^{-1} of a triangular matrix by back substitution. Then the original matrix inverse is computed simply by multiplying the two inverses as $\mathbf{A}^{-1} = (\mathbf{L}\mathbf{L}^\top)^{-1} = (\mathbf{L}^{-1})^\top(\mathbf{L}^{-1})$. As bonus, the determinant is also much easier to compute, because $\det(\mathbf{A}) = \det(\mathbf{L})^2$, and the determinant of the triangular

2220 Cholesky factor \mathbf{L} is the product of its diagonal elements so that $\det(\mathbf{A}) =$
2221 $\prod_i l_{ii}^2$.

2222 4.4 Eigendecomposition and Diagonalization

Diagonal matrices are of the form

$$\mathbf{D} = \begin{bmatrix} c_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & c_n \end{bmatrix} \quad (4.47)$$

2223 and possess a very simple structure. Therefore, they allow fast computation
2224 of determinants, powers and inverses. The determinant is the product
2225 of its diagonal entries, a matrix power \mathbf{D}^k is given by each diagonal ele-
2226 ment raised to the power k , and the inverse \mathbf{D}^{-1} is the reciprocal of its
2227 diagonal elements if all of them are non-zero.

2228 In this section, we will look at how to transform matrices into diagonal
2229 form. This is an important application of the basis change we discussed in
2230 Section 2.7.2 and eigenvalues from Section 4.2.

2231 Let us recall that two matrices \mathbf{A}, \mathbf{D} are similar (Definition 2.21) if
2232 there exists an invertible matrix \mathbf{P} , such that $\mathbf{D} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$. More specif-
2233 ically, we will look at matrices \mathbf{A} that are similar to a diagonal matrix \mathbf{D}
2234 that contains the eigenvalues of \mathbf{A} on its diagonal.

2235 **diagonalizable** 2236 **Definition 4.18** (Diagonalizable). A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is *diagonalizable*
2237 if it is similar to a diagonal matrix, in other words there exists a matrix
 $\mathbf{P} \in \mathbb{R}^{n \times n}$ so that $\mathbf{D} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$.

2238 In the following, we will see that diagonalizing a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a
2239 way of expressing the same linear mapping but in another basis (see Sec-
2240 tion 2.6.1). Specifically we will try to diagonalize a matrix \mathbf{A} by finding
2241 a new basis that consists of the eigenvectors of \mathbf{A} . We present two theo-
2242 rems, first for square matrices (Theorem 4.19) then for symmetric matri-
2243 ces (Theorem 4.21). The following results parallels the discussion we had
2244 about eigenvalues and eigenvectors (Theorem 4.11 and Theorem 4.12).

We first explore how to compute \mathbf{P} so as to diagonalize \mathbf{A} . Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, let $\lambda_1, \dots, \lambda_n$ be a set of scalars, and let $\mathbf{p}_1, \dots, \mathbf{p}_n$ be a set of vectors in \mathbb{R}^n . Then we set $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_n]$ and let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be a diagonal matrix with diagonal entries $\lambda_1, \dots, \lambda_n$. Then we can show that

$$\mathbf{AP} = \mathbf{PD} \quad (4.48)$$

2245 if and only if $\lambda_1, \dots, \lambda_n$ are the eigenvalues of \mathbf{A} and the \mathbf{p}_i are the cor-
2246 responding eigenvectors of \mathbf{A} .

We can see that this statement holds because

$$\mathbf{AP} = \mathbf{A}[\mathbf{p}_1, \dots, \mathbf{p}_n] = [\mathbf{Ap}_1, \dots, \mathbf{Ap}_n] \quad (4.49)$$

$$\mathbf{P}\mathbf{D} = [\mathbf{p}_1, \dots, \mathbf{p}_n] \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} = [\lambda_1 \mathbf{p}_1, \dots, \lambda_n \mathbf{p}_n] . \quad (4.50)$$

Thus, (4.48) implies that

$$\mathbf{A}\mathbf{p}_1 = \lambda_1 \mathbf{p}_1 \quad (4.51)$$

⋮

$$\mathbf{A}\mathbf{p}_n = \lambda_n \mathbf{p}_n \quad (4.52)$$

2247 and vice versa.

2248 Thus, the matrix \mathbf{P} must be composed of columns of eigenvectors. But
2249 this is not sufficient to know if we can diagonalize \mathbf{A} , as our definition
2250 of diagonalization requires that \mathbf{P} is invertible. From Theorem 4.3 we
2251 know that our square matrix \mathbf{P} is only invertible (has determinant $\neq 0$)
2252 if it has full rank. This implies that the eigenvectors $\mathbf{p}_1, \dots, \mathbf{p}_n$ must be
2253 linearly independent. Moreover, consider that Theorem 4.11 tells us when
2254 \mathbf{A} is diagonalizable by having n independent eigenvectors, namely in only
2255 those cases where \mathbf{A} has n distinct eigenvalues. Taking together these
2256 arguments we can now combine them to formulate a key theorem of this
2257 chapter.

Theorem 4.19. *Eigendecomposition/Diagonalization theorem.* A square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ can be factored as

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1} \quad (4.53)$$

2258 where \mathbf{P} is an invertible matrix of eigenvectors and \mathbf{D} is a diagonal matrix
2259 which diagonal entries are the eigenvalues of \mathbf{A} , if and only if \mathbf{A} has n
2260 independent eigenvectors (i.e. $\text{rk}(\mathbf{P}) = n$).

Diagonalization

2261 **Definition 4.20.** A defective matrix is a square matrix if it does not have a
2262 complete set of eigenvectors (i.e. n linearly independent eigenvectors or
2263 the sum of the dimensions of the eigenspaces is n) and is therefore not
2264 diagonalizable (see also Theorem 4.11).

defective matrix

2265 **Remark.** • Any defective matrix must have 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 .

2270 • 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). \diamond

Jordan Normal Form

2273 For symmetric matrices we can obtain even stronger outcomes for the
2274 eigenvalue decomposition.

Theorem 4.21. A symmetric matrix $S = S^\top \in \mathbb{R}^{n \times n}$ can always be diagonalized into

$$S = PDP^\top \quad (4.54)$$

2275 where P is matrix of n orthogonal eigenvectors and D is a diagonal matrix
2276 of its n eigenvalues.

Proof By Theorem 4.12 we know that $P = [\mathbf{p}_1, \dots, \mathbf{p}_n]$ has n orthogonal eigenvectors of S with eigenvalues $\lambda_1, \dots, \lambda_n$. We can then write

$$(P^\top P)_{ij} = \mathbf{p}_i^\top \mathbf{p}_j \quad (4.55)$$

where

$$\mathbf{p}_i^\top \mathbf{p}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (4.56)$$

2277 and therefore $P^\top P = I$ and $P^{-1} = P^\top$.

We observe the following product

$$\lambda_i P \mathbf{p}_i = \lambda_i [\mathbf{p}_1, \dots, \mathbf{p}_n] \mathbf{p}_i = \lambda_i \mathbf{e}_i, \quad (4.57)$$

which we will use in the following derivation.

$$P^\top S P = P^\top S [\mathbf{p}_1, \dots, \mathbf{p}_n] \quad (4.58)$$

$$= P^\top [S \mathbf{p}_1, \dots, S \mathbf{p}_n] \quad (4.59)$$

$$= P^\top [\lambda_1 \mathbf{p}_1, \dots, \lambda_n \mathbf{p}_n] \quad (4.60)$$

$$= [\mathbf{p}_1, \dots, \mathbf{p}_n]^\top [\lambda_1 \mathbf{p}_1, \dots, \lambda_n \mathbf{p}_n] \quad (4.61)$$

$$= [\lambda_1 \mathbf{e}_1, \dots, \lambda_n \mathbf{e}_n] = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} = D \quad (4.62)$$

2278

□

Geometric intuition for the eigendecomposition

2279 We can interpret the eigendecomposition of a matrix as follows (see also
2280 Figure 4.6): Let A be the transformation matrix of a linear mapping with
2281 respect to the standard basis. P^{-1} performs a basis change from the stan-
2282 dard basis into the eigenbasis. This maps the eigenvectors \mathbf{p}_i (red and
2283 green arrows in Figure 4.6) onto the standard axes \mathbf{e}_i . Then, the diagonal
2284 D scales the vectors along these axes by the eigenvalues $\lambda_i \mathbf{e}_i$ and, finally,
2285 P transforms these scaled vectors back into the standard/canonical coor-
2286 dinates (yielding $\lambda_i \mathbf{p}_i$).
2287

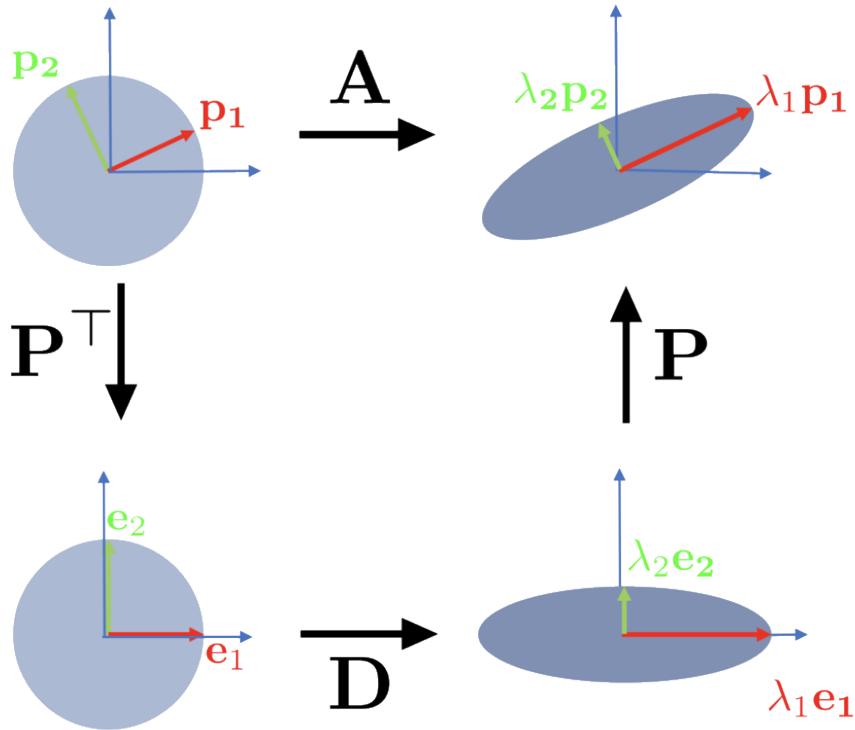


Figure 4.6 Intuition behind the eigendecomposition of a $A \in \mathbb{R}^{2 \times 2}$ in the standard basis as sequential transformations. Top-left to bottom-left: P^\top performs a basis change (here drawn in \mathbb{R}^2 and depicted as a rotation-like operation) mapping the eigenvectors into the standard basis. Bottom-left-to-bottom-right D performs a scaling along the remapped orthogonal eigenvectors, depicted here by a circle being stretched to an ellipse. Bottom-left to top-left: 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 $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

Step 1: Compute the eigenvalues and eigenvectors

The matrix has eigenvalues

$$\det(A - \lambda I) = \det \left(\begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} \right) \quad (4.63)$$

$$= (2 - \lambda)^2 - 1 = \lambda^2 - 2\lambda + 3$$

$$= (\lambda - 3)(\lambda - 1) = 0. \quad (4.64)$$

So the eigenvalues of A are $\lambda_1 = 1$ and $\lambda_2 = 3$ and the associated normalized eigenvectors are obtained via

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} p_1 = 1p_1 \quad (4.65)$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} p_2 = 3p_2. \quad (4.66)$$

This yields

$$\mathbf{p}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{p}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \quad (4.67)$$

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 \mathbf{P}

To compute the diagonalizing matrix we collect these normalized eigenvectors together

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad (4.68)$$

so that we obtain

$$\begin{aligned} \mathbf{AP} &= \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 3 \\ -1 & 3 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} = \mathbf{PD}. \end{aligned} \quad (4.69)$$

We can now obtain the matrices of the eigendecomposition by right multiplying with \mathbf{P}^{-1} . Alternatively as the matrix \mathbf{A} is symmetric we can use the orthogonality property of its eigenvectors with $\mathbf{P}^\top = \mathbf{P}^{-1}$ and solve for \mathbf{A} directly to obtain the eigendecomposition:

$$\mathbf{A} = \mathbf{PAP}^\top \quad (4.70)$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}. \quad (4.71)$$

2288 The eigenvalue decomposition of a matrix has a number of convenient
2289 properties

- Diagonal matrices \mathbf{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 \mathbf{A} via the eigenvalue decomposition

$$\mathbf{A}^k = (\mathbf{PDP}^{-1})^k = \mathbf{P}\mathbf{D}^k\mathbf{P}^{-1}. \quad (4.72)$$

2290 Computing \mathbf{D}^k is efficient because we apply this operation individually
2291 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-

2294 terpret random variables, e.g., for the Gaussian distributions we will
2295 encounter in Section 6.5 and in applications such as dimensionality re-
2296 duction Chapter 10.

2297 The eigenvalue decomposition requires square matrices, and for non-
2298 symmetric square matrices it is not guaranteed that we can transform
2299 them into diagonal form. It would be useful to be able to perform a de-
2300 composition on general matrices. In the next section, we introduce a more
2301 general matrix decomposition technique, the Singular Value Decomposi-
2302 tion.

2303 4.5 Singular Value Decomposition

2304 The Singular Value Decomposition (SVD) of a matrix is a central matrix
2305 decomposition method in linear algebra. It has been referred to as the
2306 “fundamental theorem of linear algebra” (Strang, 1993) because it can be
2307 applied to all matrices, not only to square matrices, and it always exists.
2308 Moreover, as we will explore in the following, the SVD of a linear map-
2309 ping $\Phi : V \rightarrow W$ quantifies the resulting change between the underlying
2310 geometry of these two vector spaces. We recommend Kalman (1996); Roy
2311 and Banerjee (2014) for a deeper overview of the mathematics of the SVD.

Theorem 4.22 (SVD theorem). *Let $A^{m \times n}$ be a rectangular matrix of rank r , with $r \in [0, \min(m, n)]$. The Singular Value Decomposition or SVD of A is a decomposition of A of the form*

$$\begin{matrix} & n \\ \begin{matrix} \text{---} \\ m \end{matrix} & \boxed{A} \end{matrix} = \begin{matrix} & m \\ \begin{matrix} \text{---} \\ n \end{matrix} & \boxed{U} \end{matrix} \begin{matrix} & n \\ \begin{matrix} \text{---} \\ n \end{matrix} & \boxed{\Sigma} \end{matrix} \begin{matrix} & n \\ \begin{matrix} \text{---} \\ n \end{matrix} & \boxed{V^\top} \end{matrix} \quad (4.73)$$

Singular Value
Decomposition
SVD

2312 where $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix composed of column vectors u_i ,
2313 $i = 1, \dots, m$, and $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix of column vectors
2314 v_j , $j = 1, \dots, n$, and Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geq 0$ and
2315 $\Sigma_{ij} = 0$, $i \neq j$. The SVD is always possible for any matrix A .

2316 The σ_i are called the singular values, u_i are called the left-singular vec-
2317 tors and v_j are called the right-singular vectors. By convention the singular
2318 vectors are ordered, i.e., $\sigma_1 \geq \sigma_2 \geq \sigma_r \geq 0$.

singular values
left-singular vectors
right-singular
vectors

2319 We will see a proof of this theorem later in this section. The SVD al-
2320 lows us to decompose general matrices, and the existence of the unique
2321 singular value matrix Σ requires attention. Observe that the $\Sigma \in \mathbb{R}^{m \times n}$ is
2322 rectangular, that is it is non-square. In particular note that Σ is the same
2323 size as A . This means that Σ has a diagonal submatrix that contains the
2324 singular values and needs additional zero vectors that increase the dimen-
2325 sion.

singular value
matrix

Specifically, if $m > n$ then the matrix Σ has diagonal structure up to row n and then consists of $\mathbf{0}^\top$ row vectors from $n + 1$ to m below

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix}. \quad (4.74)$$

Conversely, if $m < n$ the matrix Σ has a diagonal structure up to column m and columns that consist of $\mathbf{0}$ from $m + 1$ to n .

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & & 0 \\ 0 & 0 & \sigma_n & 0 & \dots & 0 \end{bmatrix} \quad (4.75)$$

2326 4.5.1 Geometric Intuitions for the SVD

2327 The SVD has a number of interesting geometric intuitions to offer to de-
 2328 scribe a transformation matrix. Broadly there are two intuitive views we
 2329 can have. First we consider the SVD as sequential operations performed
 2330 on the bases (discussed in the following), and second we consider the
 2331 SVD as operations performed on sets of (data) points as described in Ex-
 2332 ample 4.10.

2333 The SVD can be interpreted as a decomposition of a linear mapping
 2334 (recall Section 2.7.1) $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ into three operations (see Figure 4.7
 2335 for the following). The SVD intuition follows superficially a similar struc-
 2336 ture to our eigendecomposition intuition (confront Figure 4.7 for the SVD
 2337 with Figure 4.6 for the eigendecomposition: Broadly speaking the SVD
 2338 performs a basis change (\mathbf{V}^\top) followed by a scaling and augmentation
 2339 (or reduction) in dimensionality (Σ) and then performs a second basis
 2340 change (\mathbf{U}). The SVD entails a number of important details and caveats
 2341 which is why we will review our intuition in more detail and precision,
 2342 than we have had for the eigendecomposition.

2343 Assume we are given a transformation matrix of Φ with respect to the
 2344 standard bases B and C of \mathbb{R}^n and \mathbb{R}^m , respectively. Moreover, assume a
 2345 second basis \tilde{B} of \mathbb{R}^n and \tilde{C} of \mathbb{R}^m . Then

- 2346 1 \mathbf{V} performs a basis change in the domain \mathbb{R}^n from \tilde{B} (represented by
 2347 the red and green vectors v_1 and v_2 in Figure 4.7 top left) to the canon-
 2348 ical basis B . It is useful here to recall our discussion of basis changes
 2349 Section 2.7.2 and orthogonal matrices and orthonormal bases in Sec-
 2350 tion ??), as $\mathbf{V}^\top = \mathbf{V}^{-1}$ performs a basis change from B to \tilde{B} (the red
 2351 and green vectors are now aligned with the canonical basis in Figure 4.7
 2352 bottom left).

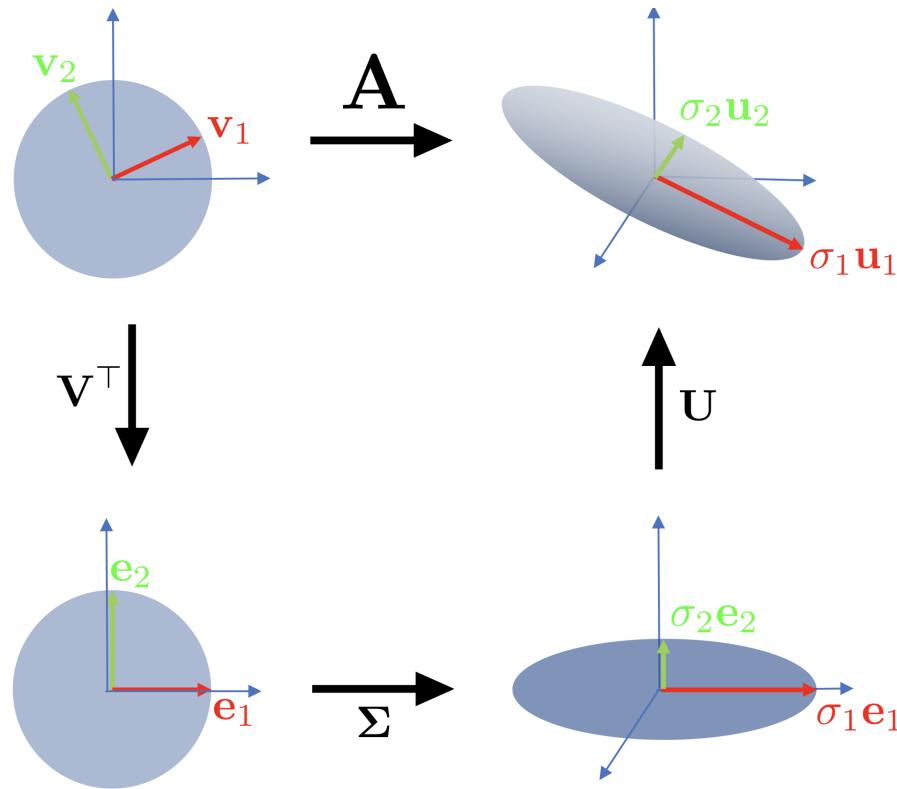
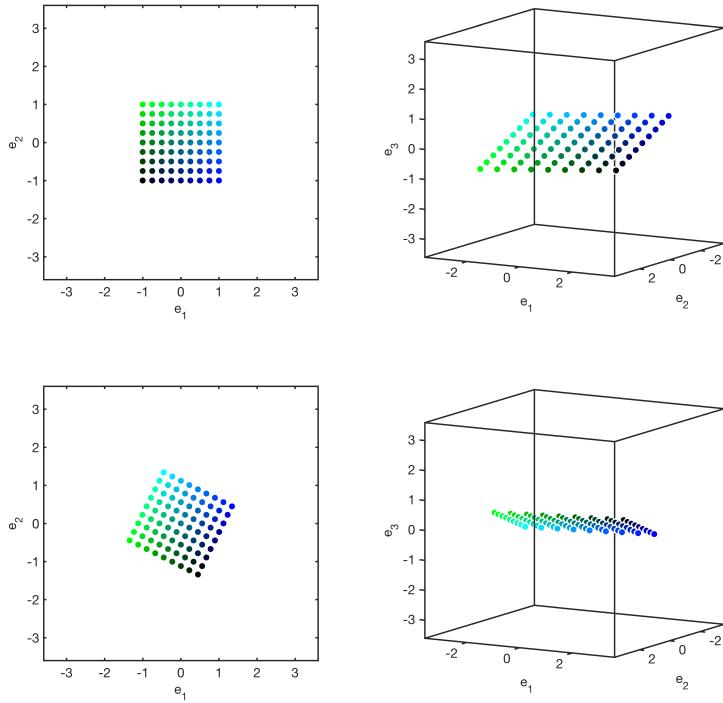


Figure 4.7 Intuition behind SVD of a $A \in \mathbb{R}^{3 \times 2}$ in the standard basis as sequential transformations. Top-left to bottom-left: V^\top performs a basis change in \mathbb{R}^2 . Bottom-left-to-bottom right Σ performs a scaling and increases the dimensionality from \mathbb{R}^2 to \mathbb{R}^3 . 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: U performs a second basis change within \mathbb{R}^3 .

- 2 Having changed the coordinate system to \tilde{B} , Σ scales the new coordinates by the singular values σ_i (and adding or deleting dimensions), i.e., Σ 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 e_1 - e_2 plane which is now embedded in a third dimension in Figure 4.7 bottom right) .
- 3 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 e_1 - 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 U and 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 Σ . We refer to Section 2.7.2 and Figure 2.11 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

$$\mathbf{A} = \begin{bmatrix} 1 & -2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (4.76)$$

$$= \mathbf{U} \Sigma \mathbf{V}^\top \quad (4.77)$$

$$= \begin{bmatrix} 0.913 & 0 & -0.408 \\ -0.365 & 0.4472 & -0.816 \\ 0.182 & 0.894 & 0.4082 \end{bmatrix} \begin{bmatrix} 2.449 & 0 \\ 0 & 1.0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.4472 & -0.894 \\ 0.8941 & 0.4472 \end{bmatrix} \quad (4.78)$$

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 $\mathbf{V}^\top \in \mathbb{R}^{2 \times 2}$ are shown in the bottom-left panel of Figure 4.8. After a mapping Σ to the codomain \mathbb{R}^3

(see bottom right panel in Figure 4.8) we can see how all the points lie on the e_1 - 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 \mathbf{A} to the codomain \mathbb{R}^3 equals the transformation of \mathcal{X} by $\mathbf{U}\Sigma\mathbf{V}^\top$, where \mathbf{U} performs a rotation within the codomain \mathbb{R}^3 so that the mapped points are no longer restricted to the e_1 - 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

$$\mathbf{S} = \mathbf{S}^\top = \mathbf{P}\mathbf{D}\mathbf{P}^\top \quad (4.79)$$

(which always exists) to the structure of the SVD of

$$\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^\top. \quad (4.80)$$

We identify

$$\mathbf{U} = \mathbf{P} = \mathbf{V}, \quad (4.81)$$

$$\mathbf{D} = \Sigma, \quad (4.82)$$

so that the SVD of symmetric matrices is their eigenvalue decomposition. \diamond

In the following we will explore why Theorem 4.22 should hold and how it is constructed. Computing the SVD of $\mathbf{A} \in \mathbb{R}^{m \times n}$ its existence is equivalent to finding two sets of orthonormal bases $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_m)$ and $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_n)$ of the domain \mathbb{R}^m and the codomain \mathbb{R}^n , respectively. From these ordered bases we will construct the matrices \mathbf{U} and \mathbf{V} , respectively.

Our plan is to start with constructing the orthonormal set of right-singular vectors $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^n$. We then construct the orthonormal set of left-singular vectors $\mathbf{u}_1, \dots, \mathbf{u}_m \in \mathbb{R}^m$. Thereafter, we will link the two and require that the orthogonality of the \mathbf{v}_i is preserved under the transformation of \mathbf{A} . This is important because we know the images \mathbf{Av}_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 $\mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{n \times n}$ from any rectangular matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. Thus, we can always diagonalize $\mathbf{A}^\top \mathbf{A}$ and obtain

$$\mathbf{A}^\top \mathbf{A} = \mathbf{P} \mathbf{D} \mathbf{P}^\top = \mathbf{P} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix} \mathbf{P}^\top. \quad (4.83)$$

Take note that the $\lambda_i \geq 0$ are the eigenvalues of $\mathbf{A}^\top \mathbf{A}$. Let us assume the SVD of \mathbf{A} exists and inject (4.73) into (4.83).

$$\mathbf{A}^\top \mathbf{A} = (\mathbf{U} \Sigma \mathbf{V}^\top)^\top (\mathbf{U} \Sigma \mathbf{V}^\top) = \mathbf{V} \Sigma^\top \mathbf{U}^\top \mathbf{U} \Sigma \mathbf{V}^\top. \quad (4.84)$$

where \mathbf{U}, \mathbf{V} are orthogonal matrices. Therefore, with $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$ we obtain

$$\mathbf{A}^\top \mathbf{A} = \mathbf{V} \Sigma^\top \Sigma \mathbf{V}^\top = \mathbf{V} \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix} \mathbf{V}^\top. \quad (4.85)$$

Comparing now (4.83) and (4.85) we identify

$$\mathbf{V} = \mathbf{P}, \quad (4.86)$$

$$\sigma_i^2 = \lambda_i. \quad (4.87)$$

2390 Therefore, the eigenvectors \mathbf{P} of $\mathbf{A}^\top \mathbf{A}$ are the right-singular vectors \mathbf{V} of 2391 \mathbf{A} (see (4.86)). They form an orthonormal basis because of Theorem 4.21, 2392 for the domain of the SVD. Moreover, the eigenvalues of $\mathbf{A}^\top \mathbf{A}$ are the 2393 squared singular values of Σ (see (4.87)).

Let us now repeat this derivation but this time we will focus on obtaining the left singular vectors \mathbf{U} instead of \mathbf{V} . Therefore we start again by computing the SVD of a symmetric matrix, this time $\mathbf{A} \mathbf{A}^\top \in \mathbb{R}^{m \times m}$ (instead of the above $\mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{n \times n}$). We inject again (4.73) and obtain:

$$\mathbf{A} \mathbf{A}^\top = (\mathbf{U} \Sigma \mathbf{V}^\top) (\mathbf{U} \Sigma \mathbf{V}^\top)^\top = \mathbf{U} \Sigma^\top \mathbf{V}^\top \mathbf{V} \Sigma \mathbf{U}^\top \quad (4.88)$$

$$= \mathbf{U} \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_m^2 \end{bmatrix} \mathbf{U}^\top. \quad (4.89)$$

2394 We can now obtain from the same arguments about symmetric matrices 2395 and their diagonalization, now applied to $\mathbf{A} \mathbf{A}^\top$, the orthonormal eigen- 2396 vectors of $\mathbf{A}^\top \mathbf{A}$. These are the left-singular vectors \mathbf{U} and form an or- 2397 thonormal basis set in the codomain of the SVD.

This leaves the question of the structure of the matrix Σ . We need to show that regardless of $n > m$ or $n < m$, that $\mathbf{A} \mathbf{A}^\top$ and $\mathbf{A}^\top \mathbf{A}$ have the

same non-zero eigenvalues: Let us assume that λ is a non-zero eigenvalue of $\mathbf{A}\mathbf{A}^\top$ and \mathbf{x} is an eigenvector belonging to λ_i . Then

$$(\mathbf{A}\mathbf{A}^\top)\mathbf{x} = \lambda\mathbf{x} \quad (4.90)$$

left multiplying by \mathbf{A} yields and pulling on the right-hand side the scalar factor λ forward

$$\mathbf{A}(\mathbf{A}\mathbf{A}^\top)\mathbf{x} = \mathbf{A}(\lambda\mathbf{x}) = \lambda(\mathbf{A}\mathbf{x}) \quad (4.91)$$

and we can use (2.31) to reorder the left-hand side factors

$$(\mathbf{A}^\top\mathbf{A})(\mathbf{A}^\top\mathbf{x}) = \lambda(\mathbf{A}\mathbf{x}). \quad (4.92)$$

2398 This is the eigenvalue equation for $\mathbf{A}\mathbf{A}^\top$. Therefore, λ is the same eigen-
2399 value for $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}^\top\mathbf{A}$, and $\mathbf{A}\mathbf{x}$ is its eigenvector. Thus, both matrices
2400 have the same non-zero eigenvalues. Thus, the Σ matrices in the SVD for
2401 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 \mathbf{V} . But, to finish construction of the SVD we link them to the orthonormal vectors \mathbf{U} . To reach this goal we use the fact the images of the \mathbf{v}_i under \mathbf{A} have to be orthonormal, too. Using the results from Section 3.4, we require that the inner product between \mathbf{Av}_i and \mathbf{Av}_j must be 0 for $i \neq j$. For any two orthogonal eigenvectors $\mathbf{v}_i, \mathbf{v}_j, i \neq j$ it holds that

$$(\mathbf{Av}_i)^\top(\mathbf{Av}_j) = \mathbf{v}_i^\top(\mathbf{A}^\top\mathbf{A})\mathbf{v}_j = \mathbf{v}_i^\top(\lambda_j\mathbf{v}_j) = \lambda_j\mathbf{v}_i^\top\mathbf{v}_j = 0. \quad (4.93)$$

2402 For the case $m > r$ this holds for all pairs $\mathbf{Av}_1, \dots, \mathbf{Av}_r$ the images are
2403 a basis of \mathbb{R}^m , while if any further vectors $\mathbf{Av}_i, i > r$ exist, they must be
2404 in the nullspace of \mathbf{A} (see remark after proof for the converse case).

To complete the SVD construction we need left-singular vectors that are orthonormal: we normalize the images of the right-singular vectors \mathbf{Av}_i and call them \mathbf{u}_i ,

$$\mathbf{u}_i = \frac{\mathbf{Av}_i}{\|\mathbf{Av}_i\|} = \frac{1}{\sqrt{\lambda_i}}\mathbf{Av}_i = \frac{1}{\sigma_i}\mathbf{Av}_i \quad (4.94)$$

2405 where the last equality was obtained from (4.87) and from equation (4.89)
2406 showing us that the eigenvalues of $\mathbf{A}\mathbf{A}^\top$ are such that $\sigma_i^2 = \lambda_i$.

2407 Therefore, the eigenvectors of $\mathbf{A}^\top\mathbf{A}$, which we know are the right-
2408 singular vectors \mathbf{v}_i and their normalized images under \mathbf{A} , the left singular
2409 vectors \mathbf{u}_i , form two self-consistent sets of orthonormal bases that are coupled by the singular value matrix Σ .

Remark. Let us rearrange (4.94) to obtain the *singular value equation*

$$\mathbf{Av}_i = \sigma_i\mathbf{u}_i, \quad i = 1, \dots, r. \quad (4.95)$$

singular value
equation

2411 This equation closely resembles the eigenvalue equation (4.27), but the
2412 vectors on the left and the right-hand sides are not the same.

For $n > m$ (4.95) holds only for $i \leq m$ and (4.95) say nothing about the \mathbf{u}_i for $i > m$, but we know by construction that they are orthonormal. Conversely for $m > n$, then (4.95) holds only for $i \leq n$. For $i > n$ we have $\mathbf{A}\mathbf{v}_i = 0$ and we still know that the \mathbf{v}_i form an orthonormal set. This means that the SVD also supplies an orthonormal basis of the kernel (or null space) or \mathbf{A} , the set of vectors \mathbf{x} with $\mathbf{Ax} = 0$ (see Section 2.7.3).

Moreover, collecting the \mathbf{v}_i as the columns of \mathbf{V} and \mathbf{u}_i as the columns of \mathbf{U} yields

$$\mathbf{AV} = \mathbf{U}\Sigma. \quad (4.96)$$

where Σ has the same dimensions as \mathbf{A} and a diagonal structure for rows $1, \dots, r$. Hence, right-multiplying with $\mathbf{V}^\top = \mathbf{V}^{-1}$ yields $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$, which is again our singular value decomposition of \mathbf{A} . \diamond

Example 4.11

Let us find the singular value decomposition of

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}. \quad (4.97)$$

Step 1: Compute the symmetrized matrix $\mathbf{A}^\top \mathbf{A}$

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & -2 & 1 \\ -2 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}. \quad (4.98)$$

Step 2: Compute the eigenvalue decomposition of $\mathbf{A}^\top \mathbf{A}$

We compute the singular values and right-singular vectors through the eigenvalue decomposition of $\mathbf{A}^\top \mathbf{A}$

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} 5 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (4.99)$$

$$= \begin{bmatrix} \frac{5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{2}} \\ \frac{1}{\sqrt{30}} & \frac{\sqrt{5}}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{5}{\sqrt{30}} & \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{30}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{-1}{\sqrt{2}} & \frac{-2}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \mathbf{P}\mathbf{D}\mathbf{P}^\top. \quad (4.100)$$

Note, that due to our orthonormality requirement implies that we chose the 3rd column of \mathbf{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 $\mathbf{A}^\top \mathbf{A}$ we obtain them straight from \mathbf{D} . Note that because $\text{rk}(\mathbf{A}) = 2$ 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 \mathbf{A} , hence,

$$\Sigma = \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (4.101)$$

We also have obtained already the right-singular vectors because

$$\mathbf{V} = \mathbf{P} = \begin{bmatrix} \frac{5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{\sqrt{2}}{2} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{2}} \end{bmatrix}. \quad (4.102)$$

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 \mathbf{A} and normalizing them by dividing them by their corresponding singular value.

$$\mathbf{u}_1 = \frac{1}{\sigma_1} \mathbf{Av}_1 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{5}{\sqrt{30}} \\ \frac{-2}{\sqrt{30}} \\ \frac{1}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}, \quad (4.103)$$

$$\mathbf{u}_2 = \frac{1}{\sigma_2} \mathbf{Av}_2 = \frac{1}{1} \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} \quad (4.104)$$

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2] = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}. \quad (4.105)$$

Note that in practice the approach illustrated here has poor numerical behaviour, and the SVD of \mathbf{A} is computed without resorting to the eigenvalue decomposition of $\mathbf{A}^\top \mathbf{A}$.

4.5.3 Eigenvalue Decomposition vs Singular Value Decomposition

Let us consider the eigendecomposition $\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$ and SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ 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 \mathbf{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 \mathbf{U} and \mathbf{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.

$$\begin{array}{c}
 \begin{array}{ccccc}
 & \text{Ali} & \text{Beatrix} & \text{Chandra} & \\
 \begin{matrix}
 \text{Star Wars} \\ \text{Blade Runner} \\ \text{Amelie} \\ \text{Delicatessen}
 \end{matrix} &
 \left[\begin{array}{cccc}
 5 & 4 & 1 & \\
 5 & 5 & 0 & \\
 0 & 0 & 5 & \\
 1 & 0 & 4 &
 \end{array} \right] = &
 \left[\begin{array}{cccc}
 -0.6710 & 0.0236 & 0.4647 & -0.5774 \\
 -0.7197 & 0.2054 & -0.4759 & 0.4619 \\
 -0.0939 & -0.7705 & -0.5268 & -0.3464 \\
 -0.1515 & -0.6030 & 0.5293 & 0.5774
 \end{array} \right] &
 \left[\begin{array}{ccc}
 9.6438 & 0 & 0 \\
 0 & 6.3639 & 0 \\
 0 & 0 & 0.7056 \\
 0 & 0 & 0
 \end{array} \right] &
 \left[\begin{array}{ccc}
 -0.7367 & -0.6515 & -0.1811 \\
 0.0852 & 0.1762 & -0.9807 \\
 0.6708 & -0.7379 & -0.0743
 \end{array} \right]
 \end{array}
 \end{array}$$

2438 A key difference between the eigendecomposition and the SVD is that
 2439 in the SVD, domain and co-domain can be vector spaces of different
 2440 dimensions.

- 2441 • In the SVD, the left and right singular vector matrices \mathbf{U} and \mathbf{V} are generally not inverse of each other. In the eigendecomposition the eigen-
 2442 vector matrices \mathbf{P} and \mathbf{P}^{-1} are inverses of each other.
- 2443 • In the SVD, the entries in the diagonal matrix Σ are all real and nonneg-
 2444 ative, which is not generally true for the diagonal matrix in the eigen-
 2445 decomposition.
- 2446 • The SVD and the eigendecomposition are closely related through their
 2447 projections
 - 2448 – The left-singular vectors of \mathbf{A} are eigenvectors of $\mathbf{A}\mathbf{A}^\top$
 - 2449 – The right-singular vectors of \mathbf{A} are eigenvectors of $\mathbf{A}^\top\mathbf{A}$.
 - 2450 – The non-zero singular values of \mathbf{A} are the square roots of the non-
 2451 zero eigenvalues of $\mathbf{A}\mathbf{A}^\top$, and equal the non-zero eigenvalues of
 2452 $\mathbf{A}^\top\mathbf{A}$.
- 2453 • For symmetric matrices the eigenvalue decomposition and the SVD are
 2454 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 $\mathbf{A} \in \mathbb{R}^{4 \times 3}$ (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 \mathbf{x}_{Ali} , $\mathbf{x}_{\text{Beatrix}}$, $\mathbf{x}_{\text{Chandra}}$.

Factoring \mathbf{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 u_i as stereotypical movies and the right-singular vectors 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 v_j . Similarly, any movie's like-ability can be expressed as a linear combination of the u_i .

Let us look at the specific outcome of performing SVD: The first left-singular vector 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 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 v_1 may reflect an idealized notion of a science fiction lover.

Similarly, u_2 , seems to capture a French art house film theme, and 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 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.

2456 2457 2458 2459 *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:

- 2460 2461 2462 2463
- 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*.
 - Some authors define the SVD a bit differently, for $A \in \mathbb{R}^{m \times n}$ and $m \geq n$

$$\underset{m \times n}{A} = \underset{m \times n, n \times n, n \times n}{U} \Sigma \underset{n \times n}{V^T} \quad (4.106)$$

2464 2465 2466 2467 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

full SVD

reduced SVD

2468 alternative notation is that Σ is diagonal, as in the eigenvalue decom-
 2469 position. However, it loses the interpretation of Σ as a transformation
 2470 matrix.

- 2471 • In Section 4.6, we will learn about matrix approximation techniques
 2472 using the SVD, which is also called the *truncated SVD*.
- 2473 • One can also define the SVD of a rank- r matrix A so that U is an
 2474 $m \times r$ matrix, Σ as a diagonal matrix $r \times r$, and V as $r \times n$ matrix.
 2475 This construction is very similar to our definition, and ensures that the
 2476 diagonal matrix Σ has only non-zero entries along the diagonal. The
 2477 main convenience of this alternative notation is that Σ is diagonal, as
 2478 in the eigenvalue decomposition.
- 2479 • One could also introduce the restriction that the SVD for A only applies
 2480 to $m \times n$ matrices with $m > n$. However, this restriction is practically
 2481 unnecessary. When $m < n$ the SVD decomposition will yield Σ with
 2482 more zero columns than rows and, consequently, the singular values
 2483 $\sigma_{m+1}, \dots, \sigma_n$ are implicitly 0.

◊

2484 The SVD is used in a variety of applications in machine learning from
 2485 least squares problems in curve fitting to solving systems of linear equa-
 2486 tions. These applications harness various important properties of the SVD,
 2487 its relation to the rank of a matrix and its ability to approximate matrices
 2488 of a given rank with lower rank matrices. Substituting the SVD form of a
 2489 matrix in computations rather use the original matrix has often the advan-
 2490 tage of making the calculation more robust to numerical rounding errors.
 2491 As we will explore in the next section the SVD's ability to approximate
 2492 matrices with "simpler" matrices in a principled manner opens up ma-
 2493 chine learning applications ranging from dimensionality reduction, topic
 2494 modeling to data compression and clustering.

2496 4.6 Matrix Approximation

2497 We will now investigate how the SVD allows us to represent a matrix A
 2498 as a sum of simpler matrices A_i .

Let us construct a rank-1 $m \times n$ matrix A_i as

$$2499 A_i = u_i v_i^\top \quad (4.107)$$

which is formed by the outer product of i th orthogonal column vector of
 U and V , respectively (see Figure 4.10 for a visual example). For a matrix A of rank r the matrix can be decomposed into a sum of rank-1
 matrices as follows A_i :

$$2500 A = \sum_{i=1}^r \sigma_i u_i v_i^\top = \sum_{i=1}^r \sigma_i A_i \quad (4.108)$$

2501 where the outer product matrices A_i are weighed by the size of the i th

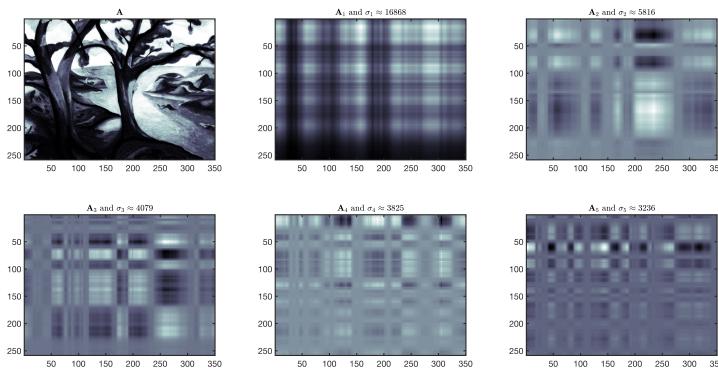


Figure 4.10 (Top left) A grayscale image is a 280×350 matrix of values between 0 (black) and 1 (white). (Middle left to Bottom right) rank-1 matrices $\mathbf{A}_1 \dots \mathbf{A}_5$ and their corresponding singular values $\sigma_1, \dots, \sigma_5$. Note, that the grid like structure of each rank-1 matrix is imposed by the outer-product of the left and right singular vectors.

2500 singular value σ_i . Thus, the sum of the outer products of matching left
 2501 and right singular vectors (weighted by their singular value) is equal to
 2502 \mathbf{A} . Note, that any terms $i > r$ are zero, as the singular values will be
 2503 0. We can see why (4.107) holds: the diagonal structure of the singular
 2504 value matrix Σ multiplies only matching left- and right-singular vectors
 2505 $(\mathbf{u}_i, \mathbf{v}_i^\top)$ and adds them up, while setting non-matching left- and right-
 2506 singular vectors $(\mathbf{u}_i, \mathbf{v}_j^\top, i \neq j)$ to zero.

In the previous paragraph we introduced a low-rank matrix \mathbf{A}_i (of rank 1). We summed up the r individual rank-1 matrices to obtain a rank r matrix \mathbf{A} . What happens if the sum does not over all matrices \mathbf{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 \mathbf{A} that we call the *rank-k approximation* $\widehat{\mathbf{A}}(k)$

$$\widehat{\mathbf{A}}(k) = \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \quad (4.109)$$

2507 of \mathbf{A} with $\text{rk}(\widehat{\mathbf{A}}) = k$.

2508 It would be useful if we could measure how large the difference be-
 2509 tween \mathbf{A} and its approximation $\widehat{\mathbf{A}}(k)$ is in terms of a single number – we
 2510 thus need the notion of a norm. We have already used norms on vectors
 2511 that measure the length of a vector. By analogy we can also define a norm
 2512 on matrices (one of the many ways to define matrix norms).

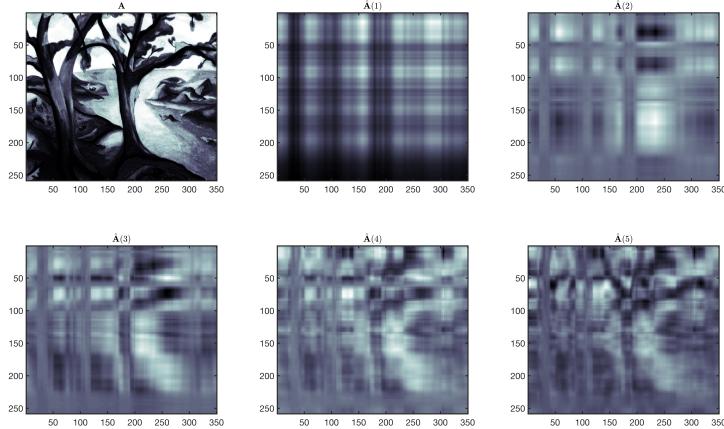
rank-k
approximation

Definition 4.23 (Spectral norm of a matrix). The spectral norm of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as the following for $\mathbf{x} \in \mathbb{R}^n$

$$\|\mathbf{A}\|_2 := \max_{\mathbf{x}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \quad \mathbf{x} \neq \mathbf{0}. \quad (4.110)$$

2513 The operator norm implies how long any vector \mathbf{x} can at most become
 2514 once it is multiplied by \mathbf{A} . This maximum lengthening is given by the SVD
 2515 of \mathbf{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 $\widehat{\mathbf{A}}(1) = \sigma_1 \mathbf{A}_1$. (Top right) is the rank-2 approximation $\widehat{\mathbf{A}}(2) = \sigma_1 \mathbf{A}_1 + \sigma_2 \mathbf{A}_2$. (Bottom left to Bottom right) are $\widehat{\mathbf{A}}(3)$ to $\widehat{\mathbf{A}}(5)$. Note how the shape of the trees becomes²⁵¹ 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 \mathbf{A} is its largest singular value σ_1 .*

We provide here a derivation of the largest singular value of matrix \mathbf{A} , illustrating the relation between the spectral norm and SVD.

$$\|\mathbf{A}\|_2 = \max_{\mathbf{x}} \frac{\|\mathbf{Ax}\|_2}{\|\mathbf{x}\|_2} = \max_{\mathbf{x}} \sqrt{\frac{\|\mathbf{Ax}\|_2^2}{\|\mathbf{x}\|_2^2}} \quad (4.111)$$

$$= \max_{\mathbf{x}} \sqrt{\frac{(\mathbf{x}\mathbf{A})^\top(\mathbf{Ax})}{\mathbf{x}^\top\mathbf{x}}} = \max_{\mathbf{x}} \sqrt{\frac{\mathbf{x}^\top(\mathbf{A}^\top\mathbf{A})\mathbf{x}}{\mathbf{x}^\top\mathbf{x}}} \quad (4.112)$$

the matrix $\mathbf{A}^\top\mathbf{A}$ is symmetric by construction and therefore we can compute the eigenvalue decomposition $\mathbf{A}^\top\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^\top$

$$\|\mathbf{A}\|_2 = \max_{\mathbf{x}} \sqrt{\frac{\mathbf{x}^\top(\mathbf{P}\mathbf{D}\mathbf{P}^\top)\mathbf{x}}{\mathbf{x}^\top\mathbf{x}}}, \quad (4.113)$$

$$(4.114)$$

where \mathbf{D} is a diagonal matrix containing the eigenvalues. Recall that \mathbf{P}^\top and \mathbf{P} perform merely a basis change and then undo it. Therefore, the most a vector \mathbf{x} can be lengthened is if it is collinear with the eigenvector associated with the largest eigenvalue.

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_1} \quad (4.115)$$

the largest eigenvalue of $\mathbf{A}^\top\mathbf{A}$ is by (4.87) the largest singular value of \mathbf{A}

$$\|\mathbf{A}\|_2 = \sigma_1 \quad (4.116)$$

Theorem 4.25 (Eckart-Young (or Eckart-Young-Minsky) theorem)). Let

$\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix of rank r and $\mathbf{B} \in \mathbb{R}^{m \times n}$ be a matrix of rank k . For any $k \leq r$ such that $\widehat{\mathbf{A}}(k) = \sum_i^k \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$, it holds that

$$\|\mathbf{A} - \widehat{\mathbf{A}}(k)\|_2 = \sigma_{k+1} \quad (4.117)$$

$$= \min_{\text{rk}(\mathbf{B}) \leq k} \|\mathbf{A} - \mathbf{B}\|_2. \quad (4.118)$$

2517 Remark. We can interpret the rank- k approximation obtained with the
 2518 SVD as a projection of the full rank matrix \mathbf{A} onto the lower-dimensional
 2519 space of rank at-most- k matrices. Of all possible projections the SVD rank-
 2520 k approximation minimizes the difference with respect to the spectral
 2521 norm between \mathbf{A} and any rank- k matrix. \diamond

We can retrace some of the steps to understand why (4.117) should hold. We observe that the difference between $\mathbf{A} - \widehat{\mathbf{A}}(k)$ is a matrix containing the sum of the remaining rank-1 matrices

$$\mathbf{A} - \widehat{\mathbf{A}}(k) = \sum_{i=k+1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \quad (4.119)$$

2522 Thus, by applying the definition of the spectral norm, (4.110), the most
 2523 a vector can be lengthened by the difference matrix is given its largest
 2524 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 \mathbf{B} with $\text{rk}(\mathbf{B}) \leq k$ such that

$$\|\mathbf{A} - \mathbf{B}\|_2 < \|\mathbf{A} - \widehat{\mathbf{A}}(k)\|_2 \quad (4.120)$$

Then there exists an $(n - k)$ -dimensional nullspace $Z \subseteq \mathbb{R}^n$ such that $\mathbf{x} \in Z \implies \mathbf{Bx} = \mathbf{0}$. In other words, have an n -dimensional space \mathbb{R}^n in which lies a lower dimensional nullspace of \mathbf{B} . Then it follows that

$$\|\mathbf{Ax}\|_2 = \|(\mathbf{A} - \mathbf{B})\mathbf{x}\|_2, \quad (4.121)$$

and by using a version of the Cauchy-Schwartz inequality (3.17) that encompasses norms of matrices we obtain

$$\|\mathbf{Ax}\|_2 \leq \|\mathbf{A} - \mathbf{B}\|_2 \|\mathbf{x}\|_2 < \sigma_{k+1} \|\mathbf{x}\|_2 \quad (4.122)$$

2525 Therefore, V is a $(n - k)$ dimensional subspace where $\|\mathbf{Ax}\|_2 < \sigma_{k+1} \|\mathbf{x}\|_2$.

2526 On the other hand there is a $(n + 1)$ -dimensional subspace where $\|\mathbf{Ax}\|_2 \geq$
 2527 $\sigma_{k+1} \|\mathbf{x}\|_2$ which is spanned by the right singular vector \mathbf{v}_{k+1} of \mathbf{A} . Adding
 2528 up dimensions of these two spaces yields a number greater n , as there
 2529 must be a non-zero vector in both spaces. This is a contradiction because
 2530 of the Rank-Nullity Theorem (recall Theorem 2.23 in Section 2.7.3).

2531 The Eckart-Young theorem implies that we can use SVD to reduce a
 2532 rank- r matrix \mathbf{A} to a rank- k matrix $\widehat{\mathbf{A}}$ in a principled, optimal (in the
 2533 spectral norm sense) manner. The effect of the low-rank approximation
 2534 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

$$\mathbf{M}_1 = \sigma_1(\mathbf{u}_1 \mathbf{v}_1^\top) \quad (4.123)$$

$$= 9.6438 \begin{bmatrix} -0.6710 \\ -0.7197 \\ -0.0939 \\ -0.1515 \end{bmatrix} \begin{bmatrix} -0.7367 & -0.6515 & -0.1811 \end{bmatrix} \quad (4.124)$$

$$= \begin{bmatrix} 4.7673 & 4.2154 & 1.1718 \\ 5.1138 & 4.5218 & 1.2570 \\ 0.6671 & 0.5899 & 0.1640 \\ 1.0765 & 0.9519 & 0.2646 \end{bmatrix} \quad (4.125)$$

This first rank-1 approximation \mathbf{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.

$$\mathbf{M}_2 = \sigma_2(\mathbf{u}_2 \mathbf{v}_2^\top) \quad (4.126)$$

$$= 6.3639 \begin{bmatrix} 0.0236 \\ 0.2054 \\ -0.7705 \\ -0.6030 \end{bmatrix} \begin{bmatrix} 0.0852 & 0.1762 & -0.9807 \end{bmatrix} \quad (4.127)$$

$$= \begin{bmatrix} 0.0128 & 0.0265 & -0.1475 \\ 0.1114 & 0.2304 & -1.2820 \\ -0.4178 & -0.8642 & 4.8084 \\ -0.3270 & -0.6763 & 3.7631 \end{bmatrix} \quad (4.128)$$

In this second rank-1 approximation \mathbf{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 $\hat{\mathbf{A}}(2)$ where we combine the first two rank-1 approximations

$$\hat{\mathbf{A}}(2) = \mathbf{M}_1 + \mathbf{M}_2 \quad (4.129)$$

$$= \begin{bmatrix} 4.7801 & 4.2419 & 1.0244 \\ 5.2252 & 4.7522 & -0.0250 \\ 0.2493 & -0.2743 & 4.9724 \\ 0.7495 & 0.2756 & 4.0278 \end{bmatrix} \quad (4.130)$$

$\hat{\mathbf{A}}(2)$ is close to the original movie ratings table

$$\mathbf{A} = \begin{bmatrix} 5 & 4 & 1 \\ 5 & 5 & 0 \\ 0 & 0 & 5 \\ 1 & 0 & 4 \end{bmatrix} \quad (4.131)$$

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.

2541

4.7 Matrix Phylogeny

2542 In Chapter 2 and 3 we covered the basics of linear algebra and analytic
 2543 geometry, in this chapter we now looked at fundamental characteristics
 2544 and methods on matrices and linear mappings. We are depicting in Fig-
 2545 ure 4.12 the phylogenetic tree of relationships between different types of
 2546 matrices (black arrows indicating “is a subset of”) and the covered opera-
 2547 tions we can perform on them (in red). For example, we already learned
 2548 in Chapter 2 about **square** matrices, which are a subset of **all (complex)**
 2549 **matrices** (top level node in the tree). We will then learn here that we can
 2550 compute a specific characteristic (**determinant**) in Section 4.1 that will
 2551 inform us whether a square matrix has an associate **inverse matrix**, thus
 2552 if it belongs to the class of non-singular, invertible matrices.

2553 Going backward through the chapter, we start with the most general
 2554 case of real matrices $\mathbb{R}^{n \times m}$ for which we can define a pseudo-inverse to
 2555 “invert” them, as well as perform **singular value decomposition (SVD)**
 2556 (Theorem 4.22). This superset of matrices is divided into the square $\mathbb{R}^{n \times n}$
 2557 matrices for which we can define the characteristic feature of the **deter-
 2558 minant** and the **trace** (Section 4.1).

The word **phylogenetic** describes how we capture the relationships among individuals or groups and derived from the greek words for “tribe” and “source”.

2559 Here the set of matrices splits in two: If the square $\mathbb{R}^{n \times n}$ matrix has n
 2560 distinct eigenvalues (or equivalently n linearly independent eigenvectors)
 2561 then the matrix is non-defective and a unique **diagonalisation/eigende-**
 2562 **composition** exists for these matrices (Theorem 4.11). In other cases we
 2563 know that a multiplicity of eigenvalues may result (see Definitions 4.13
 2564 and 4.14).

2565 Alternatively, if this square $\mathbb{R}^{n \times n}$ matrix has a non-zero determinant,
 2566 than the matrix is non-singular, i.e. an inverse matrix exists (Theorem 4.1).
 2567 Non-singular matrices are closed under addition and multiplication, have
 2568 an identity element (I) and an inverse element, thus they form a group.

2569 Note, that non-singular and non-defective matrices are not identical
 2570 sets, as for example a rotation matrix will be invertible (determinant is
 2571 non-zero) but not diagonalizable in the real numbers (non-distinct real
 2572 eigenvalues).

2573 Let us follow the branch of non-defective square $A \in \mathbb{R}^{n \times n}$ matrices.
 2574 A is normal if the condition $A^\top A = AA^\top$ holds. Moreover, if the more
 2575 restrictive condition holds $A^\top A = AA^\top = I$, then the matrix is called
 2576 orthogonal (see Definition 3.8) and is a subset of the non-singular (in-
 2577 vertible) matrices and satisfy the very useful condition $A^\top = A^{-1}$. Or-
 2578 thogonal matrices are closed under addition and multiplication, have an
 2579 identity element (I) and an inverse element, thus they also form a group.

2580 The normal matrices have a frequently encountered subset, the symmet-
 2581 ric matrices $S \in \mathbb{R}^{n \times n}$ which satisfy $S = S^\top$. Symmetric matrices have
 2582 only real eigenvalues. A subset of the symmetric matrices are the positive
 2583 definite matrices P that satisfy the condition of $x^\top Px > 0$, then a unique
 2584 a unique **Cholesky decomposition** exists (Theorem 4.17). Positive defi-
 2585 nite matrices have only positive eigenvalues and are always invertible (i.e.
 2586 have a non-zero determinant).

2587 Another subset of the symmetric matrices are the **diagonal matrices D**
 2588 in which the entries outside the main diagonal are all zero. Diagonal ma-
 2589 trices are closed under multiplication and addition, but do not necessarily
 2590 form a group (this is only the case if all diagonal entries are non-zero so
 2591 that the matrix is invertible). A prominent special case of the diagonal
 2592 matrices is the identity matrix I .

2593 4.8 Further Reading

2594 Most of the content in this chapter establishes underlying mathematics
 2595 and connects them to methods for studying mappings, many of these un-
 2596 derly machine learning at the level of underpinning software solutions and
 2597 building blocks for almost all machine learning theory. Matrix characteri-
 2598 zation using determinants, eigenspectra and eigenspaces are fundamental
 2599 features and conditions for categorizing and analyzing matrices, this ex-
 2600 tends 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,

2643 or when we want to perform data preprocessing, e.g. to decorrelate pre-
 2644 predictor variables of a design matrix (e.g. Ormoneit et al. (2001)). SVD
 2645 is the basic two-dimensional version of a more general decomposition of
 2646 data in, so called, tensors (Kolda and Bader, 2009). Tensors reflect higher-
 2647 dimensional arrays and SVD-like and low-rank approximations on tensors
 2648 are for example the CP (Carroll and Chang, 1970) or Tucker Decomposi-
 2649 tion (Tucker, 1966).

2650 The SVD low-rank approximation is frequently used in machine learn-
 2651 ing for both computational efficiency reasons. This is because it reduces
 2652 the amount of memory and operations with non-zero multiplications we
 2653 need to perform on potentially very large matrices of data (Trefethen and
 2654 Bau III, 1997). Moreover, low-rank approximation is used to operate on
 2655 matrices that may contain missing values as well as for purposes of lossy
 2656 compression and dimensionality reduction (Moonen and De Moor, 1995;
 2657 Markovsky, 2011).

2658

Exercises

- 4.1 Compute the determinant using the Laplace expansion (using the first row) and the Sarrus Rule for

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 0 & 2 & 4 \end{bmatrix} \quad (4.132)$$

- 4.2 Compute the following determinant efficiently.

$$\begin{bmatrix} 2 & 0 & 1 & 2 & 0 \\ 2 & -1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 1 & 2 \\ -2 & 0 & 2 & -1 & 2 \\ 2 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (4.133)$$

- 2659 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

$$\mathbf{A} = \begin{bmatrix} 0 & -1 & 1 & 1 \\ -1 & 1 & -2 & 3 \\ 2 & -1 & 0 & 0 \\ 1 & -1 & 1 & 0 \end{bmatrix} \quad (4.134)$$

- 2660 4.5 Diagonalizability of a matrix is unrelated to its invertibility. Determine for
 2661 the following for matrices if it is diagonalizable and/or invertible $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,

2662 $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

- 4.6 Find the SVD of the following matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \quad (4.135)$$

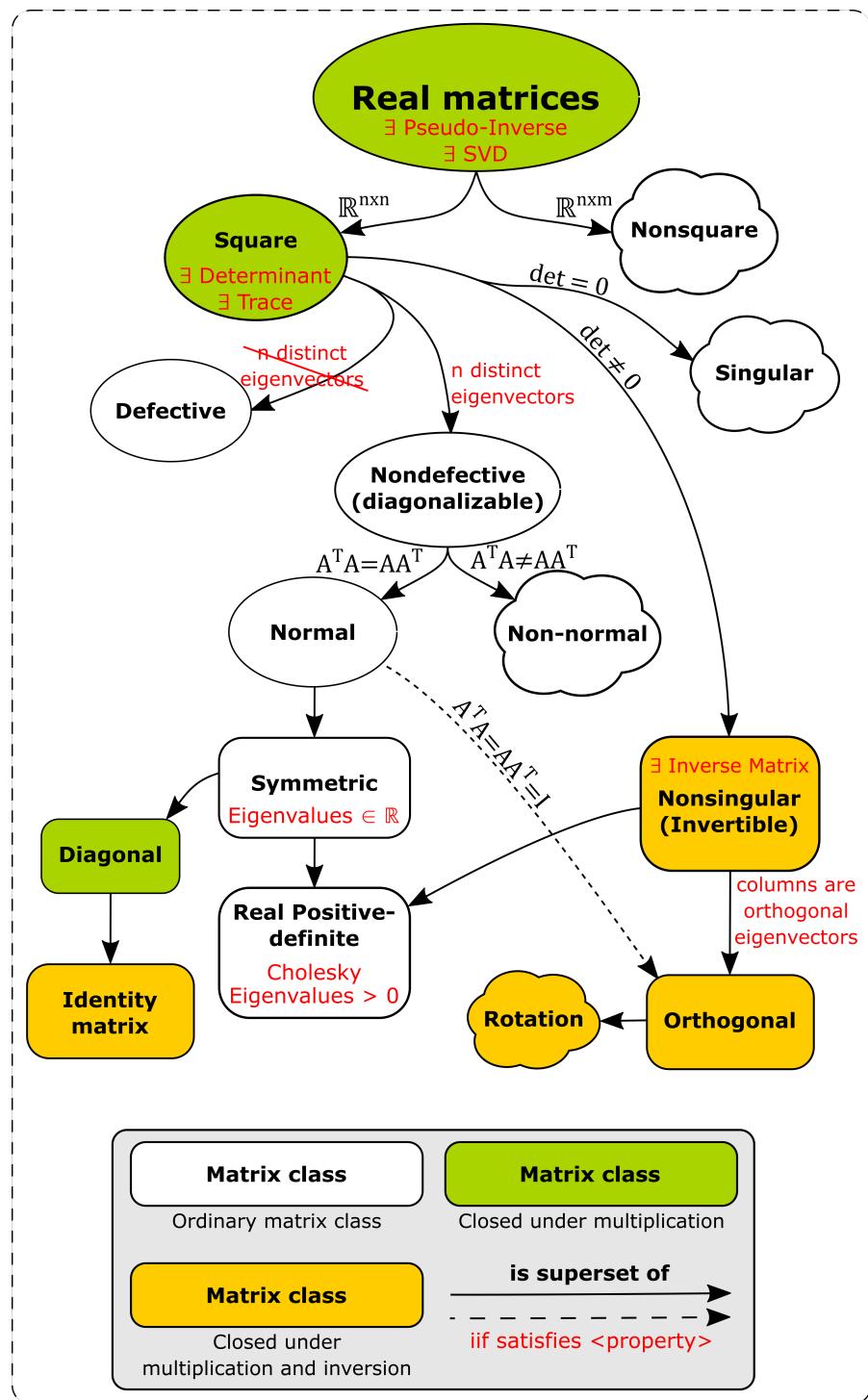
4.7 Let us find the singular value decomposition of

$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}. \quad (4.136)$$

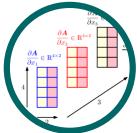
4.8 Find the best rank-1 approximation of

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \quad (4.137)$$

Figure 4.12 A functional phylogeny of matrices encountered in machine learning.



Vector Calculus

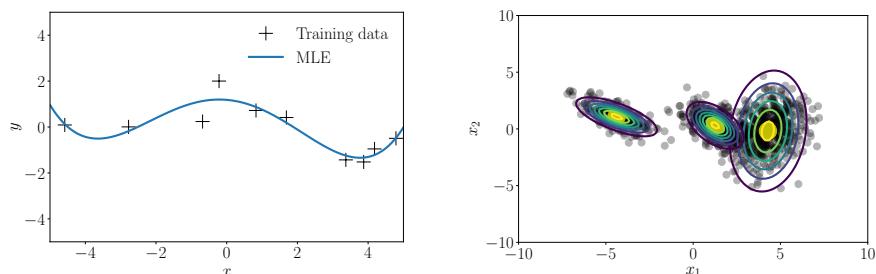


Many algorithms in machine learning are inherently based on optimizing an objective function with respect to a set of desired model parameters that control how well a model explains the data: Finding good parameters can be phrased as an optimization problem (Section 8.2 and 8.3). Examples include linear 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 (Section 7.1). 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.

Central to this chapter is the concept of a function. A function f is a quantity that relates two quantities to each other. In this book, these quantities are typically inputs $\mathbf{x} \in \mathbb{R}^D$ and targets (function values) $f(\mathbf{x})$, which we assume are real-valued if not stated otherwise. Here \mathbb{R}^D is the *domain* of f , and the function values $f(\mathbf{x})$ are the *image/codomain* of f .

domain
image

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.

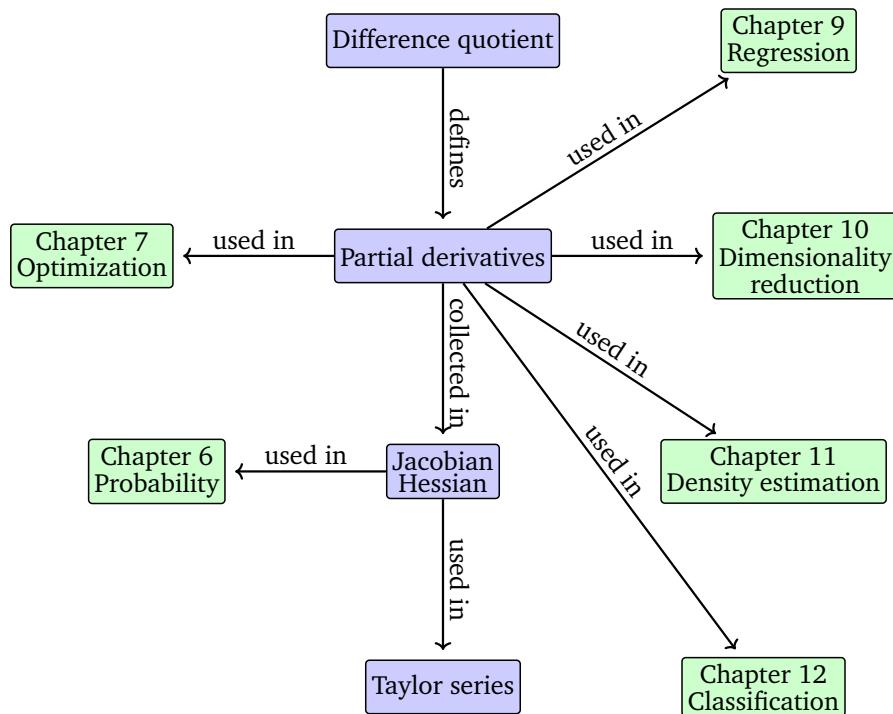


Figure 5.2 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

Section 2.7.3 provides much more detailed discussion in the context of linear functions. We often write

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \quad (5.1a)$$

$$\mathbf{x} \mapsto f(\mathbf{x}) \quad (5.1b)$$

to specify a function, where (5.1a) specifies that f is a mapping from \mathbb{R}^D to \mathbb{R} and (5.1b) specifies the explicit assignment of an input \mathbf{x} to a function value $f(\mathbf{x})$. A function f assigns every input \mathbf{x} exactly one function value $f(\mathbf{x})$.

Example 5.1

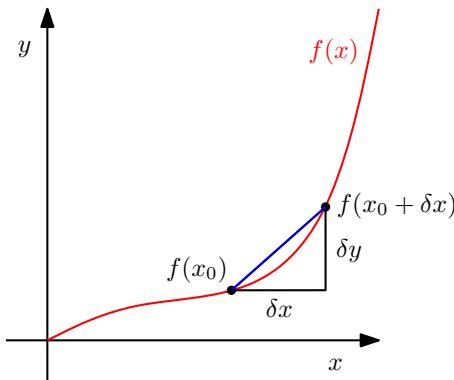
Recall the dot product as a special case of an inner product (Section 3.2). In the above notation, the function $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{x}$, $\mathbf{x} \in \mathbb{R}^2$, would be specified as

$$f : \mathbb{R}^2 \rightarrow \mathbb{R} \quad (5.2a)$$

$$\mathbf{x} \mapsto x_1^2 + x_2^2. \quad (5.2b)$$

²⁸⁵⁵ In this chapter, we will discuss how to compute gradients of functions,
²⁸⁵⁶ which is often essential to facilitate learning in machine learning models
²⁸⁵⁷ since the gradient points in the direction of steepest ascent. Therefore,

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 $\delta y / \delta x$.



vector calculus is one of the fundamental mathematical tools we need in machine learning. Throughout this book, we assume that functions are differentiable. With some additional technical definitions, which we do not cover here, many of the approaches presented can be extended to sub-differentials (functions that are continuous but not differentiable at certain points). We will look at an extension to the case of functions with constraints in Chapter 7.

5.1 Differentiation of Univariate Functions

In the following, we briefly revisit differentiation of a univariate function, which may be familiar from high-school mathematics. 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.

difference quotient **Definition 5.1** (Difference Quotient). The *difference quotient*

$$\frac{\delta y}{\delta x} := \frac{f(x + \delta x) - f(x)}{\delta x} \quad (5.3)$$

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 f to be a linear function. In the limit for $\delta x \rightarrow 0$, 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

$$\frac{df}{dx} := \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}, \quad (5.4)$$

and the secant in Figure 5.3 becomes a tangent.

The derivative of f points in the direction of steepest ascent of f .

Example 5.2 (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 (5.4) we obtain

$$\frac{df}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (5.5a)$$

$$= \lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} \quad (5.5b)$$

$$= \lim_{h \rightarrow 0} \frac{\sum_{i=0}^n \binom{n}{i} x^{n-i} h^i - x^n}{h}. \quad (5.5c)$$

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

$$\frac{df}{dx} = \lim_{h \rightarrow 0} \frac{\sum_{i=1}^n \binom{n}{i} x^{n-i} h^i}{h} \quad (5.6a)$$

$$= \lim_{h \rightarrow 0} \sum_{i=1}^n \binom{n}{i} x^{n-i} h^{i-1} \quad (5.6b)$$

$$= \lim_{h \rightarrow 0} \left(\binom{n}{1} x^{n-1} + \underbrace{\sum_{i=2}^n \binom{n}{i} x^{n-i} h^{i-1}}_{\rightarrow 0 \text{ as } h \rightarrow 0} \right) \quad (5.6c)$$

$$= \frac{n!}{1!(n-1)!} x^{n-1} = nx^{n-1}. \quad (5.6d)$$

2878

5.1.1 Taylor Series

2879 The Taylor series is a representation of a function f as an infinite sum of
2880 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 $f : \mathbb{R} \rightarrow \mathbb{R}$ at x_0 is defined as

Taylor polynomial
We define $t^0 := 1$
for all $t \in \mathbb{R}$.

$$T_n(x) := \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k, \quad (5.7)$$

2881 where $f^{(k)}(x_0)$ is the k th derivative of f at x_0 (which we assume exists)

2882 and $\frac{f^{(k)}(x_0)}{k!}$ are the coefficients of the polynomial.

Definition 5.4 (Taylor Series). For a smooth function $f \in \mathcal{C}^\infty, f : \mathbb{R} \rightarrow \mathbb{R}$, the *Taylor series* of f at x_0 is defined as

Taylor series

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k. \quad (5.8)$$

$f \in C^\infty$ means that²⁸⁸³
 f is continuously²⁸⁸⁴
differentiable

infinitely many²⁸⁸⁵
times.

Maclaurin series²⁸⁸⁶
analytic²⁸⁸⁷

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*.

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 $k \leq n$ since all derivatives $f^{(i)}$, $i > k$ vanish. \diamond

Example 5.3 (Taylor Polynomial)

We consider the polynomial

$$f(x) = x^4 \quad (5.9)$$

and seek the Taylor polynomial T_6 , evaluated at $x_0 = 1$. We start by computing the coefficients $f^{(k)}(1)$ for $k = 0, \dots, 6$:

$$f(1) = 1 \quad (5.10)$$

$$f'(1) = 4 \quad (5.11)$$

$$f''(1) = 12 \quad (5.12)$$

$$f^{(3)}(1) = 24 \quad (5.13)$$

$$f^{(4)}(1) = 24 \quad (5.14)$$

$$f^{(5)}(1) = 0 \quad (5.15)$$

$$f^{(6)}(1) = 0 \quad (5.16)$$

Therefore, the desired Taylor polynomial is

$$T_6(x) = \sum_{k=0}^6 \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \quad (5.17a)$$

$$= 1 + 4(x - 1) + 6(x - 1)^2 + 4(x - 1)^3 + (x - 1)^4 + 0. \quad (5.17b)$$

Multiplying out and re-arranging yields

$$\begin{aligned} T_6(x) &= (1 - 4 + 6 - 4 + 1) + x(4 - 12 + 12 - 4) \\ &\quad + x^2(6 - 12 + 6) + x^3(4 - 4) + x^4 \end{aligned} \quad (5.18a)$$

$$= x^4 = f(x), \quad (5.18b)$$

i.e., we obtain an exact representation of the original function.

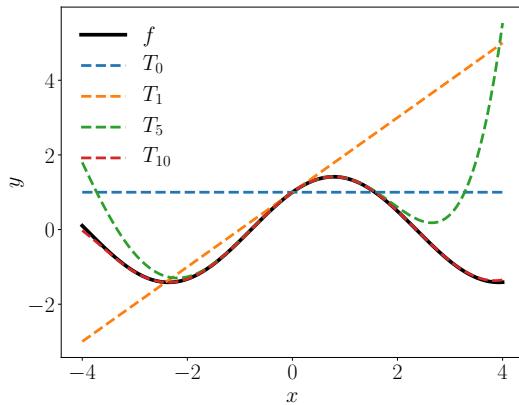


Figure 5.4 Taylor polynomials. The original function $f(x) = \sin(x) + \cos(x)$ (black, solid) is approximated by Taylor polynomials (dashed) around $x_0 = 0$. Higher-order Taylor polynomials approximate the function f better and more globally. T_{10} is already similar to f in $[-4, 4]$.

Example 5.4 (Taylor Series)

Consider the function in Figure 5.4 given by

$$f(x) = \sin(x) + \cos(x) \in \mathcal{C}^\infty. \quad (5.19)$$

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:

$$f(0) = \sin(0) + \cos(0) = 1 \quad (5.20)$$

$$f'(0) = \cos(0) - \sin(0) = 1 \quad (5.21)$$

$$f''(0) = -\sin(0) - \cos(0) = -1 \quad (5.22)$$

$$f^{(3)}(0) = -\cos(0) + \sin(0) = -1 \quad (5.23)$$

$$f^{(4)}(0) = \sin(0) + \cos(0) = f(0) = 1 \quad (5.24)$$

⋮

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

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \quad (5.25a)$$

$$= 1 + x - \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 - \dots \quad (5.25b)$$

$$= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 \mp \dots + x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 \mp \dots \quad (5.25c)$$

$$= \sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k)!} x^{2k} + \sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k+1)!} x^{2k+1} \quad (5.25d)$$

$$= \cos(x) + \sin(x), \quad (5.25e)$$

where we used the *power series representations*

$$\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k)!} x^{2k}, \quad (5.26)$$

$$\sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k+1)!} x^{2k+1}. \quad (5.27)$$

power series
representations

Figure 5.4 shows the corresponding first Taylor polynomials T_n for $n = 0, 1, 5, 10$.

Remark. A Taylor series is a special case of a power series

$$f(x) = \sum_{k=0}^{\infty} a_k (x - c)^k \quad (5.28)$$

2890 where a_k are coefficients and c is a constant, which has the special form
2891 in Definition 5.4. \diamond

5.1.2 Differentiation Rules

2892 In the following, we briefly state basic differentiation rules, where we
2893 denote the derivative of f by f' .
2894

$$\text{Product Rule: } (f(x)g(x))' = f'(x)g(x) + f(x)g'(x) \quad (5.29)$$

$$\text{Quotient Rule: } \left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2} \quad (5.30)$$

$$\text{Sum Rule: } (f(x) + g(x))' = f'(x) + g'(x) \quad (5.31)$$

$$\text{Chain Rule: } (g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x) \quad (5.32)$$

2895 Here, $g \circ f$ denotes function composition $x \mapsto f(x) \mapsto g(f(x))$.

Example 5.5 (Chain rule)

Let us compute the derivative of the function $h(x) = (2x + 1)^4$ using the chain rule. With

$$h(x) = (2x + 1)^4 = g(f(x)), \quad (5.33)$$

$$f(x) = 2x + 1, \quad (5.34)$$

$$g(f) = f^4 \quad (5.35)$$

we obtain the derivatives of f and g as

$$f'(x) = 2, \quad (5.36)$$

$$g'(f) = 4f^3, \quad (5.37)$$

such that the derivative of h is given as

$$h'(x) = g'(f) f'(x) = (4f^3) \cdot 2 \stackrel{(5.34)}{=} 4(2x+1)^3 \cdot 2 = 8(2x+1)^3, \quad (5.38)$$

where we used the chain rule (5.32), and substituted the definition of f in (5.34) 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 $\mathbf{x} \in \mathbb{R}^n$, e.g., $f(\mathbf{x}) = f(x_1, x_2)$. The generalization of the derivative to functions of several variables is the *gradient*.

We find the gradient of the function f with respect to \mathbf{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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$ of n variables x_1, \dots, x_n we define the *partial derivatives* as

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \lim_{h \rightarrow 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(\mathbf{x})}{h} \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_{n-1}, x_n + h) - f(\mathbf{x})}{h} \end{aligned} \quad (5.39)$$

and collect them in the row vector

$$\nabla_{\mathbf{x}} f = \text{grad } f = \frac{df}{d\mathbf{x}} = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right] \in \mathbb{R}^{1 \times n}, \quad (5.40)$$

where n is the number of variables and 1 is the dimension of the image/range/co-domain of f . Here, we defined the column vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$. The row vector in (5.40) 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. \diamond

Example 5.6 (Partial Derivatives using the Chain Rule)

For $f(x, y) = (x + 2y^3)^2$, we obtain the partial derivatives

$$\frac{\partial f(x, y)}{\partial x} = 2(x + 2y^3) \frac{\partial}{\partial x} (x + 2y^3) = 2(x + 2y^3), \quad (5.41)$$

We can use results from scalar differentiation: Each partial derivative is a derivative with respect to a scalar.

$$\frac{\partial f(x, y)}{\partial y} = 2(x + 2y^3) \frac{\partial}{\partial y}(x + 2y^3) = 12(x + 2y^3)y^2. \quad (5.42)$$

where we used the chain rule (5.32) to compute the partial derivatives.

2912 **Remark** (Gradient as a Row Vector). It is not uncommon in the literature
2913 to define the gradient vector as a column vector, following the convention
2914 that vectors are generally column vectors. The reason why we define
2915 the gradient vector as a row vector is twofold: First, we can consistently
2916 generalize the gradient to vector-valued functions $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (then
2917 the gradient becomes a matrix). Second, we can immediately apply the
2918 multi-variate chain-rule without paying attention to the dimension of the
2919 gradient. We will discuss both points in Section 5.3. \diamond

Example 5.7 (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

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3 \quad (5.43)$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2 \quad (5.44)$$

and the gradient is then

$$\frac{df}{dx} = \begin{bmatrix} \frac{\partial f(x_1, x_2)}{\partial x_1} & \frac{\partial f(x_1, x_2)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 x_2 + x_2^3 & x_1^2 + 3x_1 x_2^2 \end{bmatrix} \in \mathbb{R}^{1 \times 2}. \quad (5.45)$$

5.2.1 Basic Rules of Partial Differentiation

2920 Product rule:

$(fg)' = f'g + fg'$

2921 Sum rule:

$(f + g)' = f' + g'$

2922 Chain rule:

$(g(f))' = g'(f)f'$

2923

2924

2925

2926

In the multivariate case, where $\mathbf{x} \in \mathbb{R}^n$, 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 $\mathbf{x} \in \mathbb{R}^n$ we need to pay attention: Our gradients now involve vectors and matrices, and matrix multiplication is not commutative (Section 2.2.1), i.e., the order matters.

Here are the general product rule, sum rule and chain rule:

$$\text{Product Rule: } \frac{\partial}{\partial \mathbf{x}}(f(\mathbf{x})g(\mathbf{x})) = \frac{\partial f}{\partial \mathbf{x}}g(\mathbf{x}) + f(\mathbf{x})\frac{\partial g}{\partial \mathbf{x}} \quad (5.46)$$

$$\text{Sum Rule: } \frac{\partial}{\partial \mathbf{x}}(f(\mathbf{x}) + g(\mathbf{x})) = \frac{\partial f}{\partial \mathbf{x}} + \frac{\partial g}{\partial \mathbf{x}} \quad (5.47)$$

$$\text{Chain Rule: } \frac{\partial}{\partial \mathbf{x}}(g \circ f)(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}}(g(f(\mathbf{x}))) = \frac{\partial g}{\partial f} \frac{\partial f}{\partial \mathbf{x}} \quad (5.48)$$

Let us have a closer look at the chain rule. The chain rule (5.48) 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 $\partial g/\partial \mathbf{x}$ remains.

This is only an intuition, but not mathematically correct since the partial derivative is not a fraction.

2935

5.2.2 Chain Rule

Consider a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ 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 (5.48) for multivariate functions as

$$\frac{df}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{bmatrix} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \quad (5.49)$$

where d denotes the gradient and ∂ partial derivatives.

Example 5.8

Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin t$ and $x_2 = \cos t$, then

$$\frac{df}{dt} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \quad (5.50a)$$

$$= 2 \sin t \frac{\partial \sin t}{\partial t} + 2 \frac{\partial \cos t}{\partial t} \quad (5.50b)$$

$$= 2 \sin t \cos t - 2 \sin t = 2 \sin t (\cos t - 1) \quad (5.50c)$$

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

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial s}, \quad (5.51)$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t}, \quad (5.52)$$

and the gradient is obtained by the matrix multiplication

$$\frac{df}{d(s,t)} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial (s,t)} = \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix}}_{= \frac{\partial f}{\partial \mathbf{x}}} \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix}}_{= \frac{\partial \mathbf{x}}{\partial (s,t)}}. \quad (5.53)$$

The chain rule can⁹³⁷
be written as a
matrix
multiplication.

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.

Gradient checking²⁹⁴⁶

Remark (Verifying the Correctness of a Gradient Implementation). The definition of the partial derivatives as the limit of the corresponding difference quotient, see (5.39), 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 (5.39) 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 i th variable x_i . \diamond

2955

5.3 Gradients of Vector-Valued Functions

2956
2957
2958
2959

Thus far, we discussed partial derivatives and gradients of functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$ mapping to the real numbers. In the following, we will generalize the concept of the gradient to vector-valued functions (vector fields) $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, where $n \geq 1$ and $m > 1$.

For a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a vector $\mathbf{x} = [x_1, \dots, x_n]^\top \in \mathbb{R}^n$, the corresponding vector of function values is given as

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^m. \quad (5.54)$$

2960
2961
2962
2963

Writing the vector-valued function in this way allows us to view a vector-valued function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ as a vector of functions $[f_1, \dots, f_m]^\top$, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ that map onto \mathbb{R} . The differentiation rules for every f_i are exactly the ones we discussed in Section 5.2.

Therefore, the partial derivative of a vector-valued function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with respect to $x_i \in \mathbb{R}$, $i = 1, \dots, n$, is given as the vector

$$\frac{\partial \mathbf{f}}{\partial x_i} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{bmatrix} = \begin{bmatrix} \lim_{h \rightarrow 0} \frac{f_1(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots, x_n) - f_1(\mathbf{x})}{h} \\ \vdots \\ \lim_{h \rightarrow 0} \frac{f_m(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots, x_n) - f_m(\mathbf{x})}{h} \end{bmatrix} \in \mathbb{R}^m. \quad (5.55)$$

From (5.40), we know that we obtain the gradient of f with respect to a vector as the row vector of the partial derivatives. In (5.55), every partial derivative $\partial f / \partial x_i$ is a column vector. Therefore, we obtain the gradient of $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with respect to $\mathbf{x} \in \mathbb{R}^n$ by collecting these partial derivatives:

$$\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \left[\boxed{\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1}} \dots \boxed{\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n}} \right] \quad (5.56a)$$

$$= \left[\begin{array}{ccc} \boxed{\frac{\partial f_1(\mathbf{x})}{\partial x_1}} & \dots & \boxed{\frac{\partial f_1(\mathbf{x})}{\partial x_n}} \\ \vdots & & \vdots \\ \boxed{\frac{\partial f_m(\mathbf{x})}{\partial x_1}} & \dots & \boxed{\frac{\partial f_m(\mathbf{x})}{\partial x_n}} \end{array} \right] \in \mathbb{R}^{m \times n}. \quad (5.56b)$$

Definition 5.6 (Jacobian). The collection of all first-order partial derivatives of a vector-valued function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is called the *Jacobian*. The Jacobian \mathbf{J} is an $m \times n$ matrix, which we define and arrange as follows:

$$\mathbf{J} = \nabla_{\mathbf{x}} \mathbf{f} = \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \dots \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}, \quad (5.57)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad J(i, j) = \frac{\partial f_i}{\partial x_j}. \quad (5.58)$$

As a special case, a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$, which maps a vector $\mathbf{x} \in \mathbb{R}^n$ onto a scalar (e.g., $f(\mathbf{x}) = \sum_{i=1}^n x_i$), possesses a Jacobian that is a row vector (matrix of dimension $1 \times n$), see (5.40).

We will see how the Jacobian is used in the change-of-variable method in the context of probability distributions in Section 6.7. The amount of scaling due to the transformation of a variable is provided by the determinant.

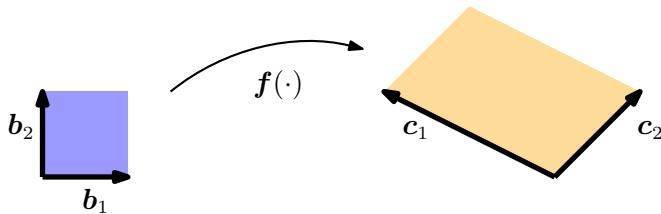
In Section 4.1, we saw that the determinant can be used to compute

Jacobian

The gradient of a function

$\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a matrix of size $m \times n$.

Figure 5.5 The determinant of the Jacobian of 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 $\mathbf{b}_1 = [1, 0]^\top$, $\mathbf{b}_2 = [0, 1]^\top$ as the sides of the unit square (blue, see Figure 5.5), the area of this square is

$$\left| \det \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = 1. \quad (5.59)$$

If we now take a parallelogram with the sides $\mathbf{c}_1 = [-2, 1]^\top$, $\mathbf{c}_2 = [1, 1]^\top$ (orange in Figure 5.5) its area is given as the absolute value of the determinant

$$\left| \det \begin{pmatrix} -2 & 1 \\ 1 & 1 \end{pmatrix} \right| = |-3| = 3, \quad (5.60)$$

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 $(\mathbf{b}_1, \mathbf{b}_2)$ to $(\mathbf{c}_1, \mathbf{c}_2)$. 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 $\{\mathbf{b}_1, \mathbf{b}_2\}$ and $\{\mathbf{c}_1, \mathbf{c}_2\}$ as bases of \mathbb{R}^2 (see Section 2.6.1 for a recap). What we effectively perform is a change of basis from $(\mathbf{b}_1, \mathbf{b}_2)$ to $(\mathbf{c}_1, \mathbf{c}_2)$, 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

$$\mathbf{J} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}, \quad (5.61)$$

such that $\mathbf{J}\mathbf{b}_1 = \mathbf{c}_1$ and $\mathbf{J}\mathbf{b}_2 = \mathbf{c}_2$. The absolute value of the determinant of \mathbf{J} , which yields the scaling factor we are looking for, is given as $|\det(\mathbf{J})| = 3$, i.e., the area of the square spanned by $(\mathbf{c}_1, \mathbf{c}_2)$ is three times greater than the area spanned by $(\mathbf{b}_1, \mathbf{b}_2)$.

Approach 2 The linear algebra approach works nicely for linear

transformations; for nonlinear transformations (which become relevant in Section 6.7), we can follow a more general approach using partial derivatives.

For this approach, we consider a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that performs a variable transformation. In our example, f maps the coordinate representation of any vector $x \in \mathbb{R}^2$ with respect to (b_1, b_2) onto the coordinate representation $y \in \mathbb{R}^2$ with respect to (c_1, 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 f . For this we need to find out how $f(x)$ changes if we modify x a bit. This question is exactly answered by the Jacobian matrix $\frac{df}{dx} \in \mathbb{R}^{2 \times 2}$. Since we can write

$$y_1 = -2x_1 + x_2 \quad (5.62)$$

$$y_2 = x_1 + x_2 \quad (5.63)$$

we obtain the functional relationship between x and y , which allows us to get the partial derivatives

$$\frac{\partial y_1}{\partial x_1} = -2, \quad \frac{\partial y_1}{\partial x_2} = 1, \quad \frac{\partial y_2}{\partial x_1} = 1, \quad \frac{\partial y_2}{\partial x_2} = 1 \quad (5.64)$$

and compose the Jacobian as

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}. \quad (5.65)$$

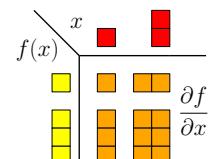
The Jacobian represents the coordinate transformation we are looking for and is exact if the coordinate transformation is linear (as in our case), and (5.65) recovers exactly the basis change matrix in (5.61). If the coordinate transformation is nonlinear, the Jacobian approximates this nonlinear transformation locally with a linear one. The absolute value of the *Jacobian determinant* $|\det(J)|$ is the factor areas or volumes are scaled by when coordinates are transformed. In our case, we obtain $|\det(J)| = 3$.

The Jacobian determinant and variable transformations will become relevant in Section 6.7 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*.

Throughout this chapter, we encountered derivatives of functions. Figure 5.6 summarizes the dimensions of those derivatives. If $f : \mathbb{R} \rightarrow \mathbb{R}$ the gradient is simply a scalar (top-left entry). For $f : \mathbb{R}^D \rightarrow \mathbb{R}$ the gradient is a $1 \times D$ row vector (top-right entry). For $f : \mathbb{R} \rightarrow \mathbb{R}^E$, the gradient is an $E \times 1$ column vector, and for $f : \mathbb{R}^D \rightarrow \mathbb{R}^E$ the gradient is an $E \times D$ matrix.

Geometrically, the Jacobian determinant gives the magnification/scaling factor when we transform an area or volume. Jacobian determinant

Figure 5.6
Overview of the dimensionality of (partial) derivatives.



Example 5.9 (Gradient of a Vector-Valued Function)

We are given

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}, \quad \mathbf{f}(\mathbf{x}) \in \mathbb{R}^M, \quad \mathbf{A} \in \mathbb{R}^{M \times N}, \quad \mathbf{x} \in \mathbb{R}^N.$$

To compute the gradient $d\mathbf{f}/d\mathbf{x}$ we first determine the dimension of $d\mathbf{f}/d\mathbf{x}$: Since $\mathbf{f} : \mathbb{R}^N \rightarrow \mathbb{R}^M$, it follows that $d\mathbf{f}/d\mathbf{x} \in \mathbb{R}^{M \times N}$. Second, to compute the gradient we determine the partial derivatives of f with respect to every x_j :

$$f_i(\mathbf{x}) = \sum_{j=1}^N A_{ij}x_j \implies \frac{\partial f_i}{\partial x_j} = A_{ij} \quad (5.66)$$

Finally, we collect the partial derivatives in the Jacobian and obtain the gradient as

$$\frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} = \mathbf{A} \in \mathbb{R}^{M \times N}. \quad (5.67)$$

Example 5.10 (Chain Rule)

Consider the function $h : \mathbb{R} \rightarrow \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$f : \mathbb{R}^2 \rightarrow \mathbb{R} \quad (5.68)$$

$$g : \mathbb{R} \rightarrow \mathbb{R}^2 \quad (5.69)$$

$$f(\mathbf{x}) = \exp(x_1x_2^2), \quad (5.70)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = g(t) = \begin{bmatrix} t \cos t \\ t \sin t \end{bmatrix} \quad (5.71)$$

and compute the gradient of h with respect to t . Since $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $g : \mathbb{R} \rightarrow \mathbb{R}^2$ we note that

$$\frac{\partial f}{\partial \mathbf{x}} \in \mathbb{R}^{1 \times 2}, \quad \frac{\partial g}{\partial t} \in \mathbb{R}^{2 \times 1}. \quad (5.72)$$

The desired gradient is computed by applying the chain-rule:

$$\frac{dh}{dt} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right] \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \quad (5.73a)$$

$$= [\exp(x_1x_2^2)x_2^2 \quad 2\exp(x_1x_2^2)x_1x_2] \begin{bmatrix} \cos t - t \sin t \\ \sin t + t \cos t \end{bmatrix} \quad (5.73b)$$

$$= \exp(x_1x_2^2)(x_2^2(\cos t - t \sin t) + 2x_1x_2(\sin t + t \cos t)), \quad (5.73c)$$

where $x_1 = t \cos t$ and $x_2 = t \sin t$, see (5.71).

Example 5.11 (Gradient of a Least-Squared Loss in a Linear Model)

Let us consider the linear model

$$\mathbf{y} = \Phi\boldsymbol{\theta}, \quad (5.74)$$

where $\boldsymbol{\theta} \in \mathbb{R}^D$ is a parameter vector, $\Phi \in \mathbb{R}^{N \times D}$ are input features and $\mathbf{y} \in \mathbb{R}^N$ are the corresponding observations. We define the functions

$$L(\mathbf{e}) := \|\mathbf{e}\|^2, \quad (5.75)$$

$$\mathbf{e}(\boldsymbol{\theta}) := \mathbf{y} - \Phi\boldsymbol{\theta}. \quad (5.76)$$

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

$$\frac{\partial L}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1 \times D}. \quad (5.77)$$

The chain rule allows us to compute the gradient as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}}, \quad (5.78)$$

where the d th element is given by

$$\frac{\partial L}{\partial \boldsymbol{\theta}}[1, d] = \sum_{n=1}^N \frac{\partial L}{\partial \mathbf{e}}[n] \frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}}[n, d]. \quad (5.79)$$

We know that $\|\mathbf{e}\|^2 = \mathbf{e}^\top \mathbf{e}$ (see Section 3.2) and determine

$$\frac{\partial L}{\partial \mathbf{e}} = 2\mathbf{e}^\top \in \mathbb{R}^{1 \times N}. \quad (5.80)$$

Furthermore, we obtain

$$\frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}} = -\Phi \in \mathbb{R}^{N \times D}, \quad (5.81)$$

such that our desired derivative is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -2\mathbf{e}^\top \Phi \stackrel{(5.76)}{=} -\underbrace{2(\mathbf{y}^\top - \boldsymbol{\theta}^\top \Phi^\top)}_{1 \times N} \underbrace{\Phi}_{N \times D} \in \mathbb{R}^{1 \times D}. \quad (5.82)$$

Remark. We would have obtained the same result without using the chain rule by immediately looking at the function

$$L_2(\boldsymbol{\theta}) := \|\mathbf{y} - \Phi\boldsymbol{\theta}\|^2 = (\mathbf{y} - \Phi\boldsymbol{\theta})^\top (\mathbf{y} - \Phi\boldsymbol{\theta}). \quad (5.83)$$

This approach is still practical for simple functions like L_2 but becomes impractical for deep function compositions. \diamond

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 $\boldsymbol{\theta}$.

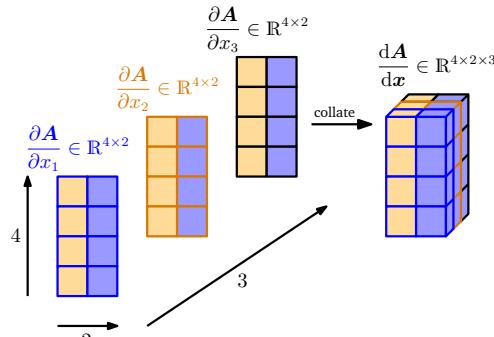
least-squares loss

```
dLdtheta =
np.einsum(
    'n,nd',
    dLde, dedtheta)
```

Figure 5.7
 Visualization of gradient computation of a matrix with respect to a vector. We are interested in computing the gradient of $\mathbf{A} \in \mathbb{R}^{4 \times 2}$ with respect to a vector $\mathbf{x} \in \mathbb{R}^3$. We know that gradient $\frac{d\mathbf{A}}{d\mathbf{x}} \in \mathbb{R}^{4 \times 2 \times 3}$. 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.

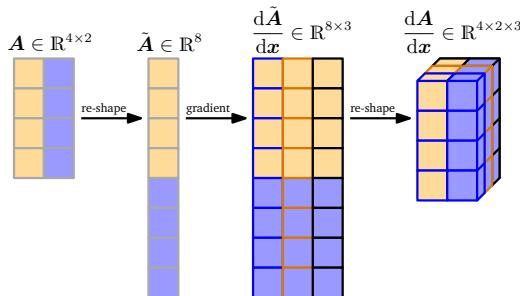
$$\mathbf{A} \in \mathbb{R}^{4 \times 2} \quad \mathbf{x} \in \mathbb{R}^3$$

Partial derivatives:



(a) Approach 1: We compute the partial derivative $\frac{\partial \mathbf{A}}{\partial x_1}, \frac{\partial \mathbf{A}}{\partial x_2}, \frac{\partial \mathbf{A}}{\partial x_3}$, each of which is a 4×2 matrix, and collate them in a $4 \times 2 \times 3$ tensor.

$$\mathbf{A} \in \mathbb{R}^{4 \times 2} \quad \mathbf{x} \in \mathbb{R}^3$$



(b) Approach 2: We re-shape (flatten) $\mathbf{A} \in \mathbb{R}^{4 \times 2}$ into a vector $\tilde{\mathbf{A}} \in \mathbb{R}^8$. Then, we compute the gradient $\frac{d\tilde{\mathbf{A}}}{d\mathbf{x}} \in \mathbb{R}^{8 \times 3}$. We obtain the gradient tensor by re-shaping this gradient as illustrated above.

We can think of a
tensor as a
multidimensional
array.
3015
3016
3017
3018

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 multidimensional tensor. We can think of this tensor as a multidimensional array that

3019 collects partial derivatives. For example, if we compute the gradient of an
 3020 $m \times n$ matrix \mathbf{A} with respect to a $p \times q$ matrix \mathbf{B} , the resulting Jacobian
 3021 would be $(p \times q) \times (m \times n)$, i.e., a four-dimensional tensor \mathbf{J} , whose entries
 3022 are given as $J_{ijkl} = \partial A_{ij} / \partial B_{kl}$.

3023 Since matrices represent linear mappings, we can exploit the fact that
 3024 there is a vector-space isomorphism (linear, invertible mapping) between
 3025 the space $\mathbb{R}^{m \times n}$ of $m \times n$ matrices and the space \mathbb{R}^{mn} of mn vectors.
 3026 Therefore, we can re-shape our matrices into vectors of lengths mn and
 3027 pq , respectively. The gradient using these mn vectors results in a Jacobian
 3028 of size $pq \times mn$. Figure 5.7 visualizes both approaches. In practical ap-
 3029 plications, it is often desirable to re-shape the matrix into a vector and
 3030 continue working with this Jacobian matrix: The chain rule (5.48) boils
 3031 down to simple matrix multiplication, whereas in the case of a Jacobian
 3032 tensor, we will need to pay more attention to what dimensions we need
 3033 to sum out.

Matrices can be transformed into vectors by stacking the columns of the matrix (“flattening”).

Example 5.12 (Gradient of Vectors with Respect to Matrices)

Let us consider the following example, where

$$\mathbf{f} = \mathbf{Ax}, \quad \mathbf{f} \in \mathbb{R}^M, \mathbf{A} \in \mathbb{R}^{M \times N}, \mathbf{x} \in \mathbb{R}^N \quad (5.84)$$

and where we seek the gradient $d\mathbf{f}/d\mathbf{A}$. Let us start again by determining the dimension of the gradient as

$$\frac{d\mathbf{f}}{d\mathbf{A}} \in \mathbb{R}^{M \times (M \times N)}. \quad (5.85)$$

By definition, the gradient is the collection of the partial derivatives:

$$\frac{d\mathbf{f}}{d\mathbf{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \mathbf{A}} \end{bmatrix}, \quad \frac{\partial f_i}{\partial \mathbf{A}} \in \mathbb{R}^{1 \times (M \times N)}. \quad (5.86)$$

To compute the partial derivatives, it will be helpful to explicitly write out the matrix vector multiplication:

$$f_i = \sum_{j=1}^N A_{ij} x_j, \quad i = 1, \dots, M, \quad (5.87)$$

and the partial derivatives are then given as

$$\frac{\partial f_i}{\partial A_{iq}} = x_q. \quad (5.88)$$

This allows us to compute the partial derivatives of f_i with respect to a row of \mathbf{A} , which is given as

$$\frac{\partial f_i}{\partial A_{i,:}} = \mathbf{x}^\top \in \mathbb{R}^{1 \times 1 \times N}, \quad (5.89)$$

$$\frac{\partial f_i}{\partial A_{k \neq i,:}} = \mathbf{0}^\top \in \mathbb{R}^{1 \times 1 \times N} \quad (5.90)$$

where we have to pay attention to the correct dimensionality. Since f_i maps onto \mathbb{R} and each row of 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 A .

We stack the partial derivatives to obtain the desired gradient as

$$\frac{\partial f_i}{\partial A} = \begin{bmatrix} \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \\ \mathbf{x}^\top \\ \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \end{bmatrix} \in \mathbb{R}^{1 \times (M \times N)}. \quad (5.91)$$

Example 5.13 (Gradient of Matrices with Respect to Matrices)

Consider a matrix $R \in \mathbb{R}^{M \times N}$ and $f : \mathbb{R}^{M \times N} \rightarrow \mathbb{R}^{N \times N}$ with

$$f(R) = R^\top R =: K \in \mathbb{R}^{N \times N}. \quad (5.92)$$

where we seek the gradient dK/dR . To solve this hard problem, let us first write down what we already know: We know that the gradient has the dimensions

$$\frac{dK}{dR} \in \mathbb{R}^{(N \times N) \times (M \times M)}, \quad (5.93)$$

which is a tensor. If we compute the partial derivative of f with respect to a single entry R_{ij} , $i, j \in \{1, \dots, N\}$, of R , we obtain an $N \times N$ -matrix

$$\frac{\partial K}{\partial R_{ij}} \in \mathbb{R}^{N \times N}. \quad (5.94)$$

Furthermore, we know that

$$\frac{dK_{pq}}{dR} \in \mathbb{R}^{1 \times M \times N} \quad (5.95)$$

for $p, q = 1, \dots, N$, where $K_{pq} = f_{pq}(R)$ is the (p, q) -th entry of $K = f(R)$.

Denoting the i -th column of R by r_i , we see that every entry of K is given by the dot product of two columns of R , i.e.,

$$K_{pq} = r_p^\top r_q = \sum_{k=1}^M R_{kp} R_{kq}. \quad (5.96)$$

When we now compute the partial derivative $\frac{\partial K_{pq}}{\partial R_{ij}}$, we obtain

$$\frac{\partial K_{pq}}{\partial R_{ij}} = \sum_{k=1}^M \frac{\partial}{\partial R_{ij}} R_{kp} R_{kq} = \partial_{pqij}, \quad (5.97)$$

$$\partial_{pqij} = \begin{cases} R_{iq} & \text{if } j = p, p \neq q \\ R_{ip} & \text{if } j = q, p \neq q \\ 2R_{iq} & \text{if } j = p, p = q \\ 0 & \text{otherwise} \end{cases}. \quad (5.98)$$

From (5.93), 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 (5.98), where $p, q, j = 1, \dots, N$ and $i = q, \dots, M$.

3034

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). Here, we use $\text{tr}(\cdot)$ as the trace (see Definition 4.4), $\det(\cdot)$ as the determinant (see Section 4.1) and $\mathbf{f}(\mathbf{X})^{-1}$ as the inverse of $\mathbf{f}(\mathbf{X})$, assuming it exists.

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^\top = \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right)^\top \quad (5.99)$$

$$\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{f}(\mathbf{X})) = \text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.100)$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{f}(\mathbf{X})) = \det(\mathbf{f}(\mathbf{X})) \text{tr} \left(\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.101)$$

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} = -\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} \quad (5.102)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} = -(\mathbf{X}^{-1})^\top \mathbf{a} \mathbf{b}^\top (\mathbf{X}^{-1})^\top \quad (5.103)$$

$$\frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}^\top \quad (5.104)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}^\top \quad (5.105)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^\top \quad (5.106)$$

$$\frac{\partial \mathbf{x}^\top \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^\top (\mathbf{B} + \mathbf{B}^\top) \quad (5.107)$$

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{A}\mathbf{s})^\top \mathbf{W} (\mathbf{x} - \mathbf{A}\mathbf{s}) = -2(\mathbf{x} - \mathbf{A}\mathbf{s})^\top \mathbf{W} \mathbf{A} \quad \text{for symmetric } \mathbf{W} \quad (5.108)$$

3035 Remark. In this book, we only cover traces and transposes of matrices.
 3036 However, we have seen that derivatives can be higher-dimensional ten-
 3037 sors, in which case the usual trace and transpose are not defined. In these
 3038 cases, the trace of a $D \times D \times E \times F$ tensor would be an $E \times F$ -dimensional
 3039 matrix. This is a special case of a tensor contraction. Similarly, when we
 3040 “transpose” a tensor, we mean swapping the first two dimensions.

3041 Specifically in (5.99)–(5.102) we require tensor-related computations
 3042 when we work with multivariate functions $f(\cdot)$ and compute derivatives
 3043 with respect to matrices (and choose not to vectorize them as discussed in
 3044 Section 5.4). \diamond

3045 A good discussion
 about 3046
 backpropagation 3047
 and the chain rule is
 available at a blog 3048
 by Tim Viera at 3049
<https://tinyurl13050.com/ycfm2yrw>. 3051

 3052
 3053

5.6 Backpropagation and Automatic Differentiation

In many machine learning applications, we find good model parameters by performing gradient descent (Section 7.1), 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

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos(x^2 + \exp(x^2)). \quad (5.109)$$

By application of the chain rule, and noting that differentiation is linear we compute the gradient

$$\begin{aligned} \frac{df}{dx} &= \frac{2x + 2x \exp(x^2)}{2\sqrt{x^2 + \exp(x^2)}} - \sin(x^2 + \exp(x^2)) (2x + 2x \exp(x^2)) \\ &= 2x \left(\frac{1}{2\sqrt{x^2 + \exp(x^2)}} - \sin(x^2 + \exp(x^2)) \right) (1 + \exp(x^2)). \end{aligned} \quad (5.110)$$

3054 Writing out the gradient in this explicit way is often impractical since it
 3055 often results in a very lengthy expression for a derivative. In practice,
 3056 it means that, if we are not careful, the implementation of the gradient
 3057 could be significantly more expensive than computing the function, which
 3058 is an unnecessary overhead. For training deep neural network models, the
 backpropagation 3059 algorithm (Kelley, 1960; Bryson, 1961; Dreyfus, 1962;
 3060 Rumelhart et al., 1986) is an efficient way to compute the gradient of an
 3061 error function with respect to the parameters of the model.

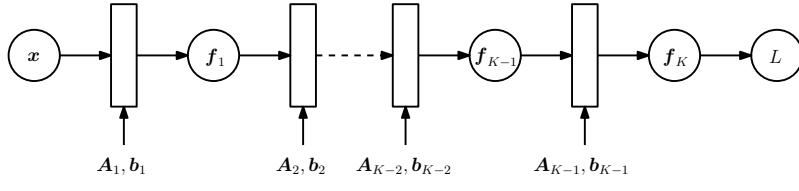


Figure 5.8 Forward pass in a multi-layer neural network to compute the loss L as a function of the inputs x and the parameters A_i, b_i .

3062

5.6.1 Gradients in a Deep Network

An area, where the chain rule is used to an extreme, is Deep Learning, where the function value y is computed as a many-level function composition

$$\mathbf{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\mathbf{x}) = f_K(f_{K-1}(\cdots(f_1(\mathbf{x}))\cdots)), \quad (5.111)$$

where \mathbf{x} are the inputs (e.g., images), \mathbf{y} are the observations (e.g., class labels) and every function $f_i, i = 1, \dots, K$, possesses its own parameters.

In neural networks with multiple layers, we have functions $f_i(\mathbf{x}_{i-1}) = \sigma(\mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{b}_i)$ in the i th layer. Here \mathbf{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 $\mathbf{A}_j, \mathbf{b}_j$ for $j = 1, \dots, K$. This also requires us to compute the gradient of L with respect to the inputs of each layer. For example, if we have inputs \mathbf{x} and observations \mathbf{y} and a network structure defined by

$$\mathbf{f}_0 := \mathbf{x} \quad (5.112)$$

$$\mathbf{f}_i := \sigma_i(\mathbf{A}_{i-1} \mathbf{f}_{i-1} + \mathbf{b}_{i-1}), \quad i = 1, \dots, K, \quad (5.113)$$

see also Figure 5.8 for a visualization, we may be interested in finding $\mathbf{A}_j, \mathbf{b}_j$ for $j = 0, \dots, K - 1$, such that the squared loss

$$L(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{f}_K(\boldsymbol{\theta}, \mathbf{x})\|^2 \quad (5.114)$$

3063 is minimized, where $\boldsymbol{\theta} = \{\mathbf{A}_0, \mathbf{b}_0, \dots, \mathbf{A}_{K-1}, \mathbf{b}_{K-1}\}$.

To obtain the gradients with respect to the parameter set $\boldsymbol{\theta}$, we require the partial derivatives of L with respect to the parameters $\boldsymbol{\theta}_j = \{\mathbf{A}_j, \mathbf{b}_j\}$ of each layer $j = 0, \dots, K - 1$. The chain rule allows us to determine the partial derivatives as

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \mathbf{f}_K} \frac{\partial \mathbf{f}_K}{\partial \boldsymbol{\theta}_{K-1}} \quad (5.115)$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \mathbf{f}_K} \boxed{\frac{\partial \mathbf{f}_K}{\partial \mathbf{f}_{K-1}} \frac{\partial \mathbf{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}} \quad (5.116)$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \mathbf{f}_K} \boxed{\frac{\partial \mathbf{f}_K}{\partial \mathbf{f}_{K-1}} \frac{\partial \mathbf{f}_{K-1}}{\partial \mathbf{f}_{K-2}} \frac{\partial \mathbf{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}} \quad (5.117)$$

We discuss the case, where the activation functions are identical in each layer to unclutter notation.

Figure 5.9
Backward pass in a multi-layer neural network to compute the gradients of the loss function.

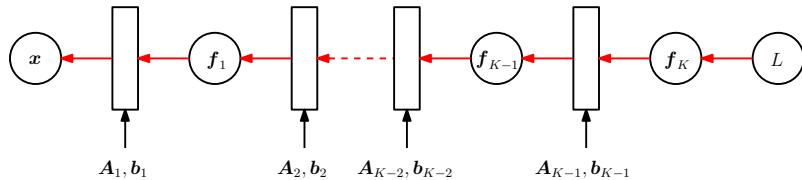
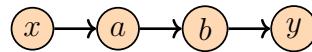


Figure 5.10 Simple graph illustrating the flow of data from x to y via some intermediate variables a, b .



$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial f_K} \frac{\partial f_K}{\partial f_{K-1}} \dots \boxed{\frac{\partial f_{i+2}}{\partial f_{i+1}} \frac{\partial f_{i+1}}{\partial \theta_i}} \quad (5.118)$$

A more in-depth discussion about gradients of neural networks can be found in Justin Domke's lecture notes at <https://tinyurl.com/yalcxgvt>.
3064
3065
3066
3067
3068
3069
3070
3071

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 $\partial L / \partial \theta_{i+1}$, then most of the computation can be reused to compute $\partial L / \partial \theta_i$. 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.

3071

5.6.2 Automatic Differentiation

automatic differentiation
3072
3073
3074
3075
3076
Automatic differentiation is different from symbolic differentiation and numerical approximations of the gradient, e.g., by using finite differences.
3077
3078
3079
3080
3081
3082

It turns out that backpropagation is a special case of a general technique in numerical analysis called *automatic differentiation*. We can think of automatic differentiation 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.

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

$$\frac{dy}{dx} = \frac{dy}{db} \frac{db}{da} \frac{da}{dx}. \quad (5.119)$$

In the general case, we work with Jacobians, which can be vectors, matrices or tensors.

Intuitively, the forward and reverse mode differ in the order of multiplication. Due to the associativity of matrix multiplication we can choose between

$$\frac{dy}{dx} = \left(\frac{dy}{db} \frac{db}{da} \right) \frac{da}{dx}, \quad (5.120)$$

$$\frac{dy}{dx} = \frac{dy}{db} \left(\frac{db}{da} \frac{da}{dx} \right). \quad (5.121)$$

3083 Equation (5.120) would be the *reverse mode* because gradients are prop-
 3084 agated backward through the graph, i.e., reverse to the data flow. Equation
 3085 (5.121) would be the *forward mode*, where the gradients flow with
 3086 the data from left to right through the graph.

reverse mode

forward mode

3087 In the following, we will focus on reverse mode automatic differentia-
 3088 tion, which is backpropagation. In the context of neural networks, where
 3089 the input dimensionality is often much higher than the dimensionality of
 3090 the labels, the reverse mode is computationally significantly cheaper than
 3091 the forward mode. Let us start with an instructive example.

Example 5.14

Consider the function

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos(x^2 + \exp(x^2)) \quad (5.122)$$

from (5.109). If we were to implement a function f on a computer, we would be able to save some computation by using *intermediate variables*:

$$a = x^2, \quad (5.123)$$

$$b = \exp(a), \quad (5.124)$$

$$c = a + b, \quad (5.125)$$

$$d = \sqrt{c}, \quad (5.126)$$

$$e = \cos(c), \quad (5.127)$$

$$f = d + e. \quad (5.128)$$

intermediate variables

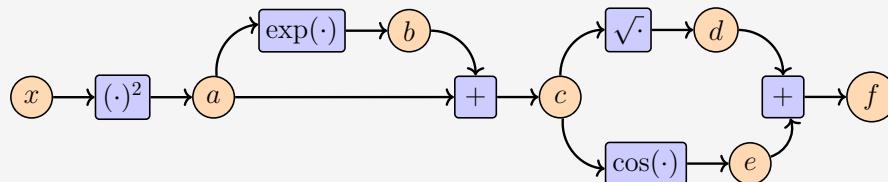


Figure 5.11
Computation graph with inputs x , function values f and intermediate variables a, b, c, d, e .

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 (5.109). 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:

$$\frac{\partial a}{\partial x} = 2x, \quad (5.129)$$

$$\frac{\partial b}{\partial a} = \exp(a), \quad (5.130)$$

$$\frac{\partial c}{\partial a} = 1 = \frac{\partial c}{\partial b}, \quad (5.131)$$

$$\frac{\partial d}{\partial c} = \frac{1}{2\sqrt{c}}, \quad (5.132)$$

$$\frac{\partial e}{\partial c} = -\sin(c), \quad (5.133)$$

$$\frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}. \quad (5.134)$$

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:

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d} \frac{\partial d}{\partial c} + \frac{\partial f}{\partial e} \frac{\partial e}{\partial c}, \quad (5.135)$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \frac{\partial c}{\partial b}, \quad (5.136)$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b} \frac{\partial b}{\partial a} + \frac{\partial f}{\partial c} \frac{\partial c}{\partial a}, \quad (5.137)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \frac{\partial a}{\partial x}. \quad (5.138)$$

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

$$\frac{\partial f}{\partial c} = 1 \cdot \frac{1}{2\sqrt{c}} + 1 \cdot (-\sin(c)), \quad (5.139)$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \cdot 1, \quad (5.140)$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b} \exp(a) + \frac{\partial f}{\partial c} \cdot 1, \quad (5.141)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \cdot 2x. \quad (5.142)$$

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 counter-intuitive since the mathematical expression for the derivative $\frac{\partial f}{\partial x}$ (5.110)

is significantly more complicated than the mathematical expression of the function $f(x)$ in (5.109).

Automatic differentiation is a formalization of the example above. Let x_1, \dots, x_d be the input variables to the function, x_{d+1}, \dots, x_{D-1} be the intermediate variables and x_D the output variable. Then the computation graph can be expressed as an equation

$$\text{For } i = d + 1, \dots, D : \quad x_i = g_i(x_{\text{Pa}(x_i)}) \quad (5.143)$$

where $g_i(\cdot)$ are elementary functions and $x_{\text{Pa}(x_i)}$ 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

$$\frac{\partial f}{\partial x_D} = 1. \quad (5.144)$$

For other variables x_i , we apply the chain rule

$$\frac{\partial f}{\partial x_i} = \sum_{x_j: x_i \in \text{Pa}(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i} = \sum_{x_j: x_i \in \text{Pa}(x_j)} \frac{\partial f}{\partial x_j} \frac{\partial g_j}{\partial x_i}, \quad (5.145)$$

where $\text{Pa}(x_j)$ is the set of parent nodes of x_j in the computation graph.
 Equation (5.143) is the forward propagation of a function, whereas (5.145) 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.

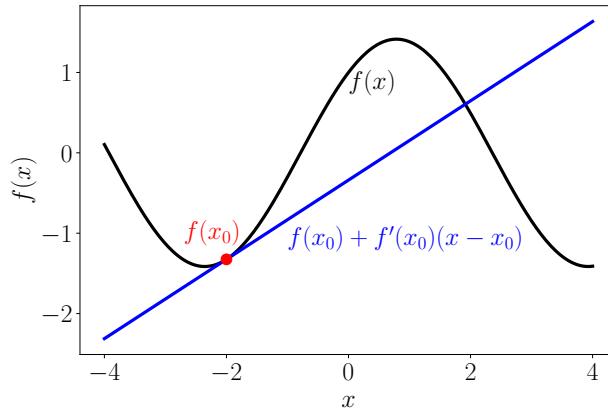
The automatic differentiation approach above works whenever we have a function that can be expressed as a computation graph, where the elementary functions are differentiable. In fact, the function may not even be a mathematical function but a computer program. However, not all computer programs can 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 $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ of two variables x, y . We use the following notation for higher-order partial derivatives (and for gradients):

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.



- $\frac{\partial^2 f}{\partial x^2}$ is the second partial derivative of f with respect to x
- $\frac{\partial^n f}{\partial x^n}$ is the n th 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 3120 The *Hessian* is the collection of all second-order partial derivatives.
If $f(x, y)$ is a twice (continuously) differentiable function then

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}, \quad (5.146)$$

Hessian matrix i.e., the order of differentiation does not matter, and the corresponding *Hessian matrix*

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \quad (5.147)$$

3121 is symmetric. The Hessian is denoted as $\nabla_{x,y}^2 f(x, y)$. Generally, for $\mathbf{x} \in \mathbb{R}^n$ 3122 and $f : \mathbb{R}^n \rightarrow \mathbb{R}$, the Hessian is an $n \times n$ matrix. The Hessian measures 3123 the curvature of the function locally around (x, y) .

3124 *Remark* (Hessian of a Vector Field). If $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a vector field, the 3125 Hessian is an $(m \times n \times n)$ -tensor. ◇

3126 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 \mathbf{x}_0 :

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + (\nabla_{\mathbf{x}} f)(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0). \quad (5.148)$$

3127 Here $(\nabla_{\mathbf{x}} f)(\mathbf{x}_0)$ is the gradient of f with respect to \mathbf{x} , evaluated at \mathbf{x}_0 .
 3128 Figure 5.12 illustrates the linear approximation of a function f at an input
 3129 \mathbf{x}_0 . The original function is approximated by a straight line. This approx-
 3130 imation is locally accurate, but the further we move away from \mathbf{x}_0 the
 3131 worse the approximation gets. Equation (5.148) is a special case of a mul-
 3132 tivariate Taylor series expansion of f at \mathbf{x}_0 , where we consider only the
 3133 first two terms. We discuss the more general case in the following, which
 3134 will allow for better approximations.

Definition 5.7 (Multivariate Taylor Series). For the *multivariate Taylor series*, we consider a function

$$f : \mathbb{R}^D \rightarrow \mathbb{R} \quad (5.149)$$

$$\mathbf{x} \mapsto f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^D, \quad (5.150)$$

3135 that is smooth at \mathbf{x}_0 .

When we define the difference vector $\boldsymbol{\delta} := \mathbf{x} - \mathbf{x}_0$, the Taylor series of f at (\mathbf{x}_0) is defined as

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{D_{\mathbf{x}}^k f(\mathbf{x}_0)}{k!} \boldsymbol{\delta}^k, \quad (5.151)$$

3136 where $D_{\mathbf{x}}^k f(\mathbf{x}_0)$ is the k -th (total) derivative of f with respect to \mathbf{x} , eval-
 3137 uated at \mathbf{x}_0 .

Definition 5.8 (Taylor Polynomial). The *Taylor polynomial* of degree n of f at \mathbf{x}_0 contains the first $n + 1$ components of the series in (5.151) and is defined as

$$T_n(\mathbf{x}) = \sum_{k=0}^n \frac{D_{\mathbf{x}}^k f(\mathbf{x}_0)}{k!} \boldsymbol{\delta}^k. \quad (5.152)$$

In (5.151) and (5.152), we used the slightly sloppy notation of $\boldsymbol{\delta}^k$, which is not defined for vectors $\mathbf{x} \in \mathbb{R}^D$, $D > 1$, and $k > 1$. Note that both $D_{\mathbf{x}}^k f$ and $\boldsymbol{\delta}^k$ are k -th order tensors, i.e., k -dimensional arrays. The

k -th order tensor $\boldsymbol{\delta}^k \in \mathbb{R}^{\overbrace{D \times D \times \dots \times D}^{k \text{ times}}}$ is obtained as a k -fold outer product, denoted by \otimes , of the vector $\boldsymbol{\delta} \in \mathbb{R}^D$. For example,

$$\boldsymbol{\delta}^2 = \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta} \boldsymbol{\delta}^\top, \quad \boldsymbol{\delta}^2[i, j] = \delta[i]\delta[j] \quad (5.153)$$

$$\boldsymbol{\delta}^3 = \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta}, \quad \boldsymbol{\delta}^3[i, j, k] = \delta[i]\delta[j]\delta[k]. \quad (5.154)$$

Figure 5.13 visualizes two such outer products. In general, we obtain the terms

$$D_{\mathbf{x}}^k f(\mathbf{x}_0) \boldsymbol{\delta}^k = \sum_{i_1=1}^D \cdots \sum_{i_k=1}^D D_{\mathbf{x}}^k f(\mathbf{x}_0)[i_1, \dots, i_k] \delta[i_1] \cdots \delta[i_k] \quad (5.155)$$

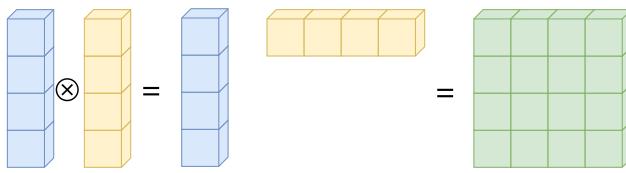
3138 in the Taylor series, where $D_{\mathbf{x}}^k f(\mathbf{x}_0) \boldsymbol{\delta}^k$ contains k -th order polynomials.

multivariate Taylor series

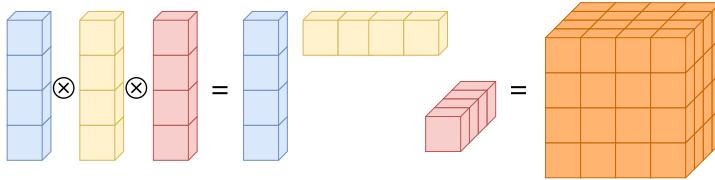
Taylor polynomial

A vector can be implemented as a 1-dimensional array, a matrix as a 2-dimensional array.

Figure 5.13
Visualizing outer products. Outer products of vectors increase the dimensionality of the array by 1 per term.



(a) Given a vector $\delta \in \mathbb{R}^4$, we obtain the outer product $\delta^2 := \delta \otimes \delta = \delta\delta^\top \in \mathbb{R}^{4 \times 4}$ as a matrix.



(b) An outer product $\delta^3 := \delta \otimes \delta \otimes \delta \in \mathbb{R}^{4 \times 4 \times 4}$ results in a third-order tensor (“three-dimensional matrix”), i.e., an array with three indexes.

Now that we defined the Taylor series for vector fields, let us explicitly write down the first terms $D_x^k f(\mathbf{x}_0) \delta^k$ of the Taylor series expansion for $k = 0, \dots, 3$ and $\delta := \mathbf{x} - \mathbf{x}_0$:

```
np.einsum(
    'i,i', Df1,d
)
np.einsum(
    'ij,i,j',
    Df2,d,d
)
np.einsum(
    'ijk,i,j,k',
    Df3,d,d,d
)
```

$$k = 0 : D_x^0 f(\mathbf{x}_0) \delta^0 = f(\mathbf{x}_0) \in \mathbb{R} \quad (5.156)$$

$$k = 1 : D_x^1 f(\mathbf{x}_0) \delta^1 = \underbrace{\nabla_{\mathbf{x}} f(\mathbf{x}_0)}_{1 \times D} \underbrace{\delta}_{D \times 1} = \sum_{i=1}^D \nabla_{\mathbf{x}} f(\mathbf{x}_0)[i] \delta[i] \in \mathbb{R} \quad (5.157)$$

$$k = 2 : D_x^2 f(\mathbf{x}_0) \delta^2 = \text{tr} \left(\underbrace{\mathbf{H}(\mathbf{x}_0)}_{D \times D} \underbrace{\delta}_{D \times 1} \underbrace{\delta^\top}_{1 \times D} \right) = \delta^\top \mathbf{H}(\mathbf{x}_0) \delta \quad (5.158)$$

$$= \sum_{i=1}^D \sum_{j=1}^D H[i, j] \delta[i] \delta[j] \in \mathbb{R} \quad (5.159)$$

$$k = 3 : D_x^3 f(\mathbf{x}_0) \delta^3 = \sum_{i=1}^D \sum_{j=1}^D \sum_{k=1}^D D_x^3 f(\mathbf{x}_0)[i, j, k] \delta[i] \delta[j] \delta[k] \in \mathbb{R} \quad (5.160)$$

3139 Here, $\mathbf{H}(\mathbf{x}_0)$ is the Hessian of f evaluated at \mathbf{x}_0 .

Example 5.15 (Taylor-Series Expansion of a Function with Two Variables)

Consider the function

$$f(x, y) = x^2 + 2xy + y^3. \quad (5.161)$$

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 (5.161) 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 (5.151) for an exact alternative representation of (5.161).

To determine the Taylor series expansion, start of with the constant term and the first-order derivatives, which are given by

$$f(1, 2) = 13 \quad (5.162)$$

$$\frac{\partial f}{\partial x} = 2x + 2y \implies \frac{\partial f}{\partial x}(1, 2) = 6 \quad (5.163)$$

$$\frac{\partial f}{\partial y} = 2x + 3y^2 \implies \frac{\partial f}{\partial y}(1, 2) = 14. \quad (5.164)$$

Therefore, we obtain

$$D_{x,y}^1 f(1, 2) = \nabla_{x,y} f(1, 2) = \left[\frac{\partial f}{\partial x}(1, 2) \quad \frac{\partial f}{\partial y}(1, 2) \right] = [6 \quad 14] \in \mathbb{R}^{1 \times 2} \quad (5.165)$$

such that

$$\frac{D_{x,y}^1 f(1, 2)}{1!} \boldsymbol{\delta} = [6 \quad 14] \begin{bmatrix} x - 1 \\ y - 2 \end{bmatrix} = 6(x - 1) + 14(y - 2). \quad (5.166)$$

Note that $D_{x,y}^1 f(1, 2) \boldsymbol{\delta}$ contains only linear terms, i.e., first-order polynomials.

The second-order partial derivatives are given by

$$\frac{\partial^2 f}{\partial x^2} = 2 \implies \frac{\partial^2 f}{\partial x^2}(1, 2) = 2 \quad (5.167)$$

$$\frac{\partial^2 f}{\partial y^2} = 6y \implies \frac{\partial^2 f}{\partial y^2}(1, 2) = 12 \quad (5.168)$$

$$\frac{\partial^2 f}{\partial y \partial x} = 2 \implies \frac{\partial^2 f}{\partial y \partial x}(1, 2) = 2 \quad (5.169)$$

$$\frac{\partial^2 f}{\partial x \partial y} = 2 \implies \frac{\partial^2 f}{\partial x \partial y}(1, 2) = 2. \quad (5.170)$$

When we collect the second-order partial derivatives, we obtain the Hessian

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix}, \quad (5.171)$$

such that

$$\mathbf{H}(1, 2) = \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \in \mathbb{R}^{2 \times 2}. \quad (5.172)$$

Therefore, the next term of the Taylor-series expansion is given by

$$\frac{D_{x,y}^2 f(1, 2)}{2!} \boldsymbol{\delta}^2 = \frac{1}{2} \boldsymbol{\delta}^\top \mathbf{H}(1, 2) \boldsymbol{\delta} \quad (5.173a)$$

$$= \frac{1}{2} [x - 1 \quad y - 2] \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} [x - 1 \quad y - 2] \quad (5.173b)$$

$$= (x - 1)^2 + 2(x - 1)(y - 2) + 6(y - 2)^2. \quad (5.173c)$$

Here, $D_{x,y}^2 f(1, 2) \boldsymbol{\delta}^2$ contains only quadratic terms, i.e., second-order polynomials.

The third-order derivatives are obtained as

$$D_{x,y}^3 f = \begin{bmatrix} \frac{\partial \mathbf{H}}{\partial x} & \frac{\partial \mathbf{H}}{\partial y} \end{bmatrix} \in \mathbb{R}^{2 \times 2}, \quad (5.174)$$

$$D_{x,y}^3 f[:, :, 1] = \frac{\partial \mathbf{H}}{\partial x} = \begin{bmatrix} \frac{\partial^3 f}{\partial x^3} & \frac{\partial^3 f}{\partial x^2 \partial y} \\ \frac{\partial^3 f}{\partial x \partial y \partial x} & \frac{\partial^3 f}{\partial x \partial y^2} \end{bmatrix}, \quad (5.175)$$

$$D_{x,y}^3 f[:, :, 2] = \frac{\partial \mathbf{H}}{\partial y} = \begin{bmatrix} \frac{\partial^3 f}{\partial y \partial x^2} & \frac{\partial^3 f}{\partial y \partial x \partial y} \\ \frac{\partial^3 f}{\partial y^2 \partial x} & \frac{\partial^3 f}{\partial y^3} \end{bmatrix}. \quad (5.176)$$

Since most second-order partial derivatives in the Hessian in (5.171) are constant the only non-zero third-order partial derivative is

$$\frac{\partial^3 f}{\partial y^3} = 6 \implies \frac{\partial^3 f}{\partial y^3}(1, 2) = 6. \quad (5.177)$$

Higher-order derivatives and the mixed derivatives of degree 3 (e.g., $\frac{\partial^3 f}{\partial x^2 \partial y}$) vanish, such that

$$D_{x,y}^3 f[:, :, 1] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad D_{x,y}^3 f[:, :, 2] = \begin{bmatrix} 0 & 0 \\ 0 & 6 \end{bmatrix} \quad (5.178)$$

and

$$\frac{D_{x,y}^3 f(1, 2)}{3!} \boldsymbol{\delta}^3 = (y - 2)^3, \quad (5.179)$$

which collects all cubic terms of the Taylor series. Overall, the (exact) Taylor series expansion of f at $(x_0, y_0) = (1, 2)$ is

$$f(x) = \textcolor{red}{f}(1, 2) + \textcolor{blue}{D}_{x,y}^1 f(1, 2) \boldsymbol{\delta} + \frac{D_{x,y}^2 f(1, 2)}{2!} \boldsymbol{\delta}^2 + \frac{D_{x,y}^3 f(1, 2)}{3!} \boldsymbol{\delta}^3 \quad (5.180a)$$

$$= \textcolor{red}{f}(1, 2) + \frac{\partial f(1, 2)}{\partial x} (x - 1) + \frac{\partial f(1, 2)}{\partial y} (y - 2) \quad (5.180b)$$

$$+ \frac{1}{2!} \left(\frac{\partial^2 f(1, 2)}{\partial x^2} (x - 1)^2 + \frac{\partial^2 f(1, 2)}{\partial y^2} (y - 2)^2 \right) \quad (5.180c)$$

$$\left. + 2 \frac{\partial^2 f(1, 2)}{\partial x \partial y} (x - 1)(y - 2) \right) + \frac{1}{6} \frac{\partial^3 f(1, 2)}{\partial y^3} (y - 2)^3 \quad (5.180d)$$

$$= 13 + 6(x - 1) + 14(y - 2) \quad (5.180e)$$

$$+ (x - 1)^2 + 6(y - 2)^2 + 2(x - 1)(y - 2) + (y - 2)^3. \quad (5.180f)$$

In this case, we obtained an exact Taylor series expansion of the polynomial in (5.161), i.e., the polynomial in (5.180f) is identical to the original polynomial in (5.161). 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 (5.180f).

3140

5.9 Further Reading

3141 Further details of matrix differentials, along with a short review of the re-
 3142 quired linear algebra can be found in Magnus and Neudecker (2007). Au-
 3143 tomatic differentiation has had a long history, and the reader is referred to
 3144 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

$$\mathbb{E}_x[f(\mathbf{x})] = \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \quad (5.181)$$

3145 Even if $p(\mathbf{x})$ is in a convenient form (e.g., Gaussian), this integral gen-
 3146 erally cannot be solved analytically. The Taylor series expansion of f is
 3147 one way of finding an approximate solution: Assuming $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
 3148 is Gaussian, then the first-order Taylor series expansion around $\boldsymbol{\mu}$ locally
 3149 linearizes the nonlinear function f . For linear functions, we can compute
 3150 the mean (and the covariance) exactly if $p(\mathbf{x})$ is Gaussian distributed (see
 3151 Section 6.5). This property is heavily exploited by the *Extended Kalman*
 3152 *Filter* (Maybeck, 1979) for online state estimation in nonlinear dynamical
 3153 systems (also called “state-space models”). Other deterministic ways
 3154 to approximate the integral in (5.181) are the *unscented transform* (Julier
 3155 and Uhlmann, 1997), which does not require any gradients, or the *Laplace*
 3156 *approximation* (MacKay, 2003; Bishop, 2006; Murphy, 2012), which uses
 3157 a second-order Taylor series expansion (requiring the Hessian) for a local
 3158 Gaussian approximation of $p(\mathbf{x})$ around the mode of the posterior distri-
 3159 bution.

Extended Kalman
Filter

unscented transform
Laplace
approximation

3160

Exercises

5.1 Compute the derivative $f'(x)$ for

$$f(x) = \log(x^4) \sin(x^3). \quad (5.182)$$

5.2 Compute the derivative $f'(x)$ of the logistic sigmoid

$$f(x) = \frac{1}{1 + \exp(-x)}. \quad (5.183)$$

5.3 Compute the derivative $f'(x)$ of the function

$$f(x) = \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right), \quad (5.184)$$

where $\mu, \sigma \in \mathbb{R}$ are constants.

3162 5.4 Compute the Taylor polynomials T_n , $n = 0, \dots, 5$ of $f(x) = \sin(x) + \cos(x)$
3163 at $x_0 = 0$.

5.5 Consider the following functions

$$f_1(\mathbf{x}) = \sin(x_1) \cos(x_2), \quad \mathbf{x} \in \mathbb{R}^2 \quad (5.185)$$

$$f_2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \quad (5.186)$$

$$f_3(\mathbf{x}) = \mathbf{x} \mathbf{x}^\top, \quad \mathbf{x} \in \mathbb{R}^n \quad (5.187)$$

3164 1 What are the dimensions of $\frac{\partial f_i}{\partial \mathbf{x}}$?

3165 2 Compute the Jacobians

5.6 Differentiate f with respect to t and g with respect to \mathbf{X} , where

$$f(\mathbf{t}) = \sin(\log(\mathbf{t}^\top \mathbf{t})), \quad \mathbf{t} \in \mathbb{R}^D \quad (5.188)$$

$$g(\mathbf{X}) = \text{tr}(\mathbf{A} \mathbf{X} \mathbf{B}), \quad \mathbf{A} \in \mathbb{R}^{D \times E}, \mathbf{X} \in \mathbb{R}^{E \times F}, \mathbf{B} \in \mathbb{R}^{F \times D}, \quad (5.189)$$

3166 where tr denotes the trace.

3167 5.7 Compute the derivatives df/dx of the following functions by using the chain
3168 rule. Provide the dimensions of every single partial derivative. Describe your
3169 steps in detail.

1

$$f(z) = \log(1 + z), \quad z = \mathbf{x}^\top \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^D$$

2

$$f(\mathbf{z}) = \sin(\mathbf{z}), \quad \mathbf{z} = \mathbf{A} \mathbf{x} + \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{E \times D}, \mathbf{x} \in \mathbb{R}^D, \mathbf{b} \in \mathbb{R}^E$$

3170 where $\sin(\cdot)$ is applied to every element of \mathbf{z} .

3171 5.8 Compute the derivatives df/dx of the following functions.
3172 Describe your steps in detail.

1 Use the chain rule. Provide the dimensions of every single partial derivative.

$$\begin{aligned} f(z) &= \exp\left(-\frac{1}{2}z\right) \\ z &= g(\mathbf{y}) = \mathbf{y}^\top \mathbf{S}^{-1} \mathbf{y} \\ \mathbf{y} &= h(\mathbf{x}) = \mathbf{x} - \boldsymbol{\mu} \end{aligned}$$

3173 where $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^D, \mathbf{S} \in \mathbb{R}^{D \times D}$.

2

$$f(\mathbf{x}) = \text{tr}(\mathbf{x} \mathbf{x}^\top + \sigma^2 \mathbf{I}), \quad \mathbf{x} \in \mathbb{R}^D$$

3174 Here $\text{tr}(\mathbf{A})$ is the trace of \mathbf{A} , i.e., the sum of the diagonal elements A_{ii} .

3175 Hint: Explicitly write out the outer product.

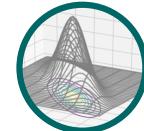
- 3 Use the chain rule. Provide the dimensions of every single partial derivative. You do not need to compute the product of the partial derivatives explicitly.

$$\begin{aligned}\mathbf{f} &= \tanh(\mathbf{z}) \in \mathbb{R}^M \\ \mathbf{z} &= \mathbf{Ax} + \mathbf{b}, \quad \mathbf{x} \in \mathbb{R}^N, \mathbf{A} \in \mathbb{R}^{M \times N}, \mathbf{b} \in \mathbb{R}^M.\end{aligned}$$

3176 Here, \tanh is applied to every component of \mathbf{z} .

6

Probability and Distributions



random variable

probability distribution

3014 Probability, loosely speaking, concerns the study of uncertainty. Probabil-
 3015 ity can be thought of as the fraction of times an event occurs, or as a degree
 3016 of belief about an event. We then would like to use this probability to mea-
 3017 sure the chance of something occurring in an experiment. As mentioned
 3018 in Chapter 1, we often quantify: uncertainty in the data, uncertainty in the
 3019 machine learning model, and uncertainty in the predictions produced by
 3020 the model. Quantifying uncertainty requires the idea of a *random variable*,
 3021 which is a function that maps outcomes of random experiments to a set of
 3022 properties that we are interested in. Associated with the random variable
 3023 is a function which measures the probability that a particular outcome (or
 3024 set of outcomes) will occur, this is called the *probability distribution*.

3025 Probability distributions are used as a building block for other con-
 3026 cepts, such as probabilistic modeling (Section 8.3), graphical models (Sec-
 3027 tion 8.4) and model selection (Section 8.5). In the next section, we present
 3028 the three concepts that define a probability space (the state space, the
 3029 events and the probability of an event) and how they are related to a
 3030 fourth concept called the random variable. The presentation is deliber-
 3031 ately slightly hand wavy since a rigorous presentation would occlude the
 3032 main idea. An outline of the concepts presented in this chapter are shown
 3033 in Figure 6.1.

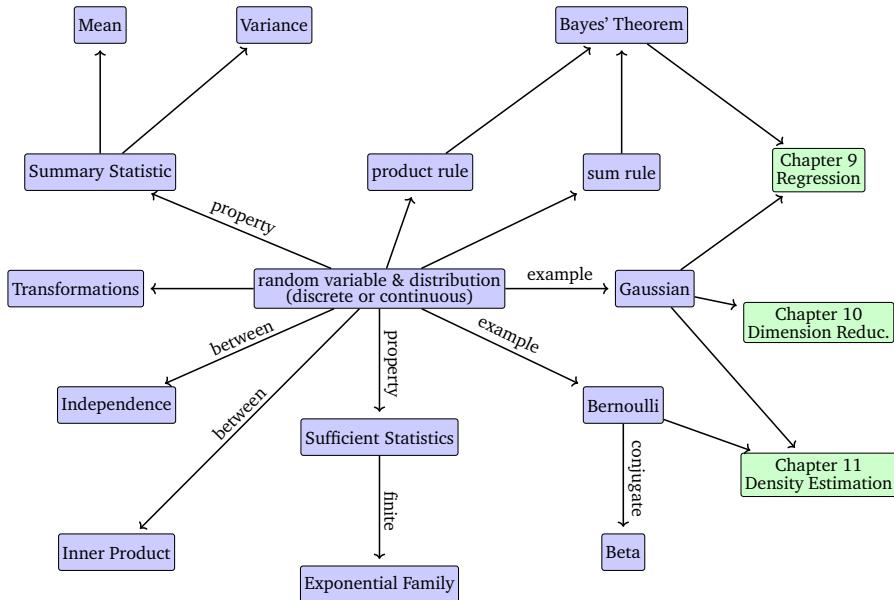
6.1 Construction of a Probability Space

3034 The theory of probability aims at defining a mathematical structure to
 3035 describe random outcomes of experiments. For example, when tossing a
 3036 single coin, one cannot determine the outcome, but by doing a large num-
 3037 ber of coin tosses, one can observe a regularity in the average outcome.
 3038 Using this mathematical structure of probability, the goal is to perform
 3039 automated reasoning, and in this sense probability generalizes logical rea-
 3040 soning (Jaynes, 2003).

6.1.1 Philosophical Issues

3041 When constructing automated reasoning systems, classical Boolean logic
 3042 does not allow us to express certain forms of plausible reasoning. Consider

Figure 6.1 A mind map of the concepts related to random variables and probability distributions, as described in this chapter.



3045 the following scenario: We observe that A is false. We find B becomes
 3046 less plausible although no conclusion can be drawn from classical logic.
 3047 We observe that B is true. It seems A becomes more plausible. We use
 3048 this form of reasoning daily. We are waiting for a friend, and consider
 3049 three possibilities; H1: she is on time, H2: she has been delayed by traffic
 3050 and H3: she has been abducted by aliens. When we observe our friend
 3051 is late, we must logically rule out H1. We also tend to consider H2 to be
 3052 more likely, though we are not logically required to do so. Finally, we may
 3053 consider H3 to be possible, but we continue to consider it quite unlikely.

3054 "For plausible
 3055 reasoning it is
 3056 necessary to extend
 3057 the discrete true and
 3058 false values of truth³⁰⁵⁷
 3059 to continuous plausibilities." (Jaynes,³⁰⁵⁹
 2003)

3060 The philosophical basis of probability and how it should be somehow
 3061 related to what we think should be true (in the logical sense) was studied
 3062 by Cox (Jaynes, 2003). Another way to think about it is that if we are pre-
 3063 cise about our common sense we end up constructing probabilities. E.T.
 3064 Jaynes (1922–1998) identified three mathematical criteria, which must
 3065 apply to all plausibilities:

- 3066 1 The degrees of plausibility are represented by real numbers.
- 3067 2 These numbers must be based on the rules of common sense.
- 3068 a) Consistency or non-contradiction: when the same result can be reached

3069 through different means, the same plausibility value must be found
 3070 in all cases.

- 3071 b) Honesty: All available data must be taken into account.
 3072 c) Reproducibility: If our state of knowledge about two problems are the
 3073 same, then we must assign the same degree of plausibility to both of
 3074 them.

3075 The Cox-Jaynes's theorem proves these plausibilities to be sufficient to
 3076 define the universal mathematical rules that apply to plausibility p , up to
 3077 transformation by an arbitrary monotonic function. Crucially, these rules
 3078 are the rules of probability.

3079 *Remark.* In machine learning and statistics, there are two major interpre-
 3080 tations of probability: the Bayesian and frequentist interpretations (Bishop,
 3081 2006). The Bayesian interpretation uses probability to specify the degree
 3082 of uncertainty that the user has about an event, and is sometimes referred
 3083 to as subjective probability or degree of belief. The frequentist interpreta-
 3084 tion considers probability to be the relative frequencies of events, in the
 3085 limit when one has infinite data. \diamond

3086 Some machine learning texts on probabilistic models use lazy notation
 3087 and jargon, which is confusing. Multiple distinct concepts are all referred
 3088 to as “probability distribution”, and the reader has to often disentangle
 3089 the meaning from the context. One trick to help make sense of probability
 3090 distributions is to check whether we are trying to model something cate-
 3091 gorical (a discrete random variable) or something continuous (a continu-
 3092 us random variable). The kinds of questions we tackle in machine learning
 3093 are closely related to whether we are considering categorical or continu-
 3094 ous models.

3095 6.1.2 Probability and Random Variables

3096 Modern probability is based on a set of axioms proposed by Kolmogorov (Ja-
 3097 cod and Protter, 2004, Chapter 1 and 2) that introduce the three concepts
 3098 of state space, event space and probability measure. The probability space
 3099 models a real world process (referred to as an experiment) with random
 3100 outcomes.

3101 The state space Ω

3102 The *state space* is the set of all possible outcomes of the experiment,
 3103 usually denoted by Ω . For example, two successive coin tosses have a
 3104 state space of $\{\text{hh}, \text{tt}, \text{ht}, \text{th}\}$, where “h” denotes “heads” and “t” denotes
 3105 “tails”.

state space

3106 The event space \mathcal{A}

3107 The *event space* is the space of potential results of the experiment. A
 3108 subset A of the state space Ω is in the event space \mathcal{A} if at the end of the
 3109 experiment we can observe whether a particular state $\omega \in \Omega$ is in A .

event space

3110 The event space \mathcal{A} is obtained by considering the collection of subsets
 3111 of Ω , and for discrete probability distributions (Section 6.2.1) \mathcal{A} is often
 3112 the powerset of Ω .

3113 **The probability P**

3114 With each event $A \in \mathcal{A}$, we associate a number $P(A)$ that measures the
 3115 probability or degree of belief that the event will occur. $P(A)$ is called
 3116 the *probability* of A .

3117 The probability of a single event must lie in the interval $[0, 1]$, and
 3118 the total probability over all states in the state space Ω must be 1, i.e.,
 3119 $P(\Omega) = 1$. Given a probability space (Ω, \mathcal{A}, P) we are interested to use
 3120 it to model some real world phenomenon. In machine learning we often
 3121 avoid explicitly referring to the probability space, but instead refer to
 3122 probabilities on quantities of interest which we denote by \mathcal{T} . We introduce
 3123 a function $x : \Omega \rightarrow \mathcal{T}$ that takes an element of Ω (an event) and returns
 3124 a particular quantity of interest, a value in \mathcal{T} . For example in the case of
 3125 tossing two coins and counting the number of heads, a random variable x
 3126 maps to the three possible outcomes: $x(\text{hh}) = 2$, $x(\text{ht}) = 1$, $x(\text{th}) = 1$ and
 3127 $x(\text{tt}) = 0$. In this particular case $\mathcal{T} = \{0, 1, 2\}$, and it is the probabilities
 3128 on elements of \mathcal{T} that we are interested in. This association or mapping
 random variable 3129 from Ω to \mathcal{T} is called a *random variable*. The name “random variable” is a
 3130 great source of misunderstanding as it is neither random nor is it a vari-
 3131 able. It is a function. For a finite state space Ω and finite \mathcal{T} , the function
 3132 corresponding to a random variable is essentially a look up table. For any
 3133 subset $S \subseteq \mathcal{T}$ we associate $P_x(S) \in [0, 1]$ (the probability) to a partic-
 3134 ular event occurring corresponding to the random variable x . Example 6.1
 3135 provides a concrete example illustrating the above terminology.

3136 *Remark.* The state space Ω above unfortunately is referred to by differ-
 3137 ent names in different books. Another common name for Ω is sample
 3138 space (Grinstead and Snell, 1997; Jaynes, 2003), and state space is some-
 3139 times reserved for referring to states in a dynamical system (Hasselblatt
 3140 and Katok, 2003). Other names sometimes used to describe Ω are: sample
 3141 description space, possibility space and (very confusingly) event space.
 3142 ◇

Example 6.1

We assume that the reader is already familiar with computing probabilities of intersections and unions of sets of events. A more gentle introduction to probability with many examples can be found in Chapter 2 of Walpole et al. (2011).

Consider a statistical experiment where we model a funfair game consisting of drawing two coins from a bag (with replacement). There are coins from USA (denoted as $\$$) and UK (denoted as \pounds) in the bag, and since we draw two coins from the bag, there are four outcomes in total.

This toy example is
 essentially a biased
 coin flip example.

The state space or sample space Ω of this experiment is then $(\$, \$)$, $(\$, \mathcal{L})$, $(\mathcal{L}, \$)$, $(\mathcal{L}, \mathcal{L})$. Let us assume that the composition of the bag of coins is such that a draw returns at random a $\$$ with probability 0.3.

The event we are interested in is the total number of times the repeated draw returns $\$$. Let us define a random variable x that maps the state space Ω to \mathcal{T} , that denotes the number of times we draw $\$$ out of the bag. We can see from the above state space we can get zero $\$$, one $\$$ or two $\$$ s, and therefore $\mathcal{T} = \{0, 1, 2\}$. The random variable x (a function or look up table) can be represented as a table like below

$$x((\$, \$)) = 2 \quad (6.1)$$

$$x((\$, \mathcal{L})) = 1 \quad (6.2)$$

$$x((\mathcal{L}, \$)) = 1 \quad (6.3)$$

$$x((\mathcal{L}, \mathcal{L})) = 0. \quad (6.4)$$

Since we return the first coin we draw before drawing the second, this implies that the two draws are independent of each other, which we will discuss in Section 6.4.5. Note that there are two experimental outcomes which map to the same event, where only one of the draws return $\$$. Therefore the probability mass function (Section 6.2.1) of x is given by the calculations below

$$\begin{aligned} P(x = 2) &= P((\$, \$)) \\ &= P(\$) \times P(\$) \\ &= 0.3 \times 0.3 = 0.09 \end{aligned} \quad (6.5)$$

$$\begin{aligned} P(x = 1) &= P((\$, \mathcal{L}) \cup (\mathcal{L}, \$)) \\ &= P((\$, \mathcal{L})) + P((\mathcal{L}, \$)) \\ &= 0.3 \times (1 - 0.3) + (1 - 0.3) \times 0.3 = 0.42 \end{aligned} \quad (6.6)$$

$$\begin{aligned} P(x = 0) &= P((\mathcal{L}, \mathcal{L})) \\ &= P(\mathcal{L}) \times P(\mathcal{L}) \\ &= (1 - 0.3) \times (1 - 0.3) = 0.49. \end{aligned} \quad (6.7)$$

In the above calculation, notice that we have equated two different concepts, the probability of the output of x and the probability of the states in Ω . For example in (6.7) we say $P(x = 0) = P((\mathcal{L}, \mathcal{L}))$.

Consider the random variable $x : \Omega \rightarrow \mathcal{T}$ and a subset $S \subseteq \mathcal{T}$ (for example a single element of \mathcal{T} such as the outcome that one head is obtained when tossing two coins). Let $x^{-1}(S)$ be the pre-image of S by x , that is the set of elements of Ω that map to S under x ; $\{\omega \in \Omega : x(\omega) \in S\}$. Another way to understand the transformation of probability from events in Ω via the random variable x is to associate it with the probability of the pre-image of S (Jacod and Protter, 2004). That is for $S \subseteq \mathcal{T}$, we have the

following notation

$$P_x(S) = P(x \in S) = P(x^{-1}(S)) = P(\{\omega \in \Omega : x(\omega) \in S\}). \quad (6.8)$$

3143 The left hand side of (6.8) is the probability of the set of possible outcomes
3144 (e.g. number of heads = 1) that we are interested in. Via the random
3145 variable x that maps states to outcomes, we see in the right hand side
3146 of (6.8) that this is the probability of the set of states (in Ω) that have
3147 the property (e.g. ht, th). We say that a random variable x is distributed
3148 according to a particular probability distribution P_x , which defines the
3149 probability mapping between the event and the probability of the outcome
3150 of the random variable. The two concepts are intertwined, but for ease
3151 of presentation we will discuss some properties with respect to random
3152 variables and others with respect to their distributions.

3153 *Remark.* The range of the random variable \mathcal{T} is used to indicate the kind
3154 of probability space, that is a \mathcal{T} random variable. When \mathcal{T} is finite or
3155 countably infinite, this is called a discrete random variable (Section 6.2.1).
3156 For continuous random variables (Section 6.2.2) we only consider $\mathcal{T} = \mathbb{R}$
3157 or $\mathcal{T} = \mathbb{R}^d$. \diamond

6.1.3 Statistics

3158 Probability theory and statistics are often presented together, but they con-
3159 cern different aspects of uncertainty. One way of contrasting them is by the
3160 kinds of problems that are considered. Using probability we can consider
3161 a model of some process where the underlying uncertainty is captured by
3162 random variables, and we use the rules of probability to derive what hap-
3163 pens. Using statistics we observe that something has happened, and try
3164 to figure out the underlying process that explains the observations. In this
3165 sense machine learning is close to statistics in its goals, that is to construct
3166 a model that adequately represents the process that generated the data.
3167 When the machine learning model is a probabilistic model, we can use the
3168 rules of probability to calculate the “best fitting” model for some data.

3169 Another aspect of machine learning systems is that we are interested
3170 in generalization error (see Chapter 8). This means that we are actually
3171 interested in the performance of our system on instances that we will ob-
3172 serve in future, which are not identical to the instances that we have seen
3173 so far. This analysis of future performance relies on probability and statis-
3174 tics, most of which is beyond what will be presented in this chapter. The
3175 interested reader is encouraged to look at the books by Shalev-Shwartz
3176 and Ben-David (2014); Boucheron et al. (2013). We will see more about
3177 statistics in Chapter 8.

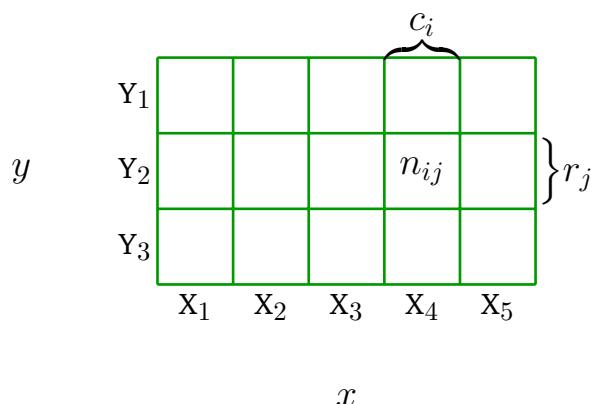


Figure 6.2
Visualization of a discrete bivariate probability mass function, with random variables x and y . This diagram is from Bishop (2006).

3179

6.2 Discrete and Continuous Probabilities

3180 Let us focus our attention on ways to describe the probability of an event
 3181 as introduced in Section 6.1. Depending on whether the state space is
 3182 discrete or continuous the natural way to refer to distributions is different.
 3183 When the outcome space \mathcal{T} is discrete, we can specify the probability that
 3184 a random variable x takes a particular value $x \in \mathcal{T}$, denoted as $P(x = x)$.
 3185 The expression $P(x = x)$ for a discrete random variable x is known as the
 3186 *probability mass function*. When the outcome space \mathcal{T} is continuous, e.g.,
 3187 the real line \mathbb{R} , it is more natural to specify the probability that a random
 3188 variable x is in an interval. By convention we specify the probability that
 3189 a random variable x is less than a particular value x , denoted $P(x \leq x)$.
 3190 The expression $P(x \leq x)$ for a continuous random variable x is known as
 3191 the *cumulative distribution function*. We will discuss continuous random
 3192 variables in Section 6.2.2. We will revisit the nomenclature and contrast
 3193 discrete and continuous random variables in Section 6.2.3.

3194 *Remark.* We will use the phrase *univariate* distribution to refer to distributions of only one random variable (denoted by non-bold x). We will refer to distributions of more than one random variable as *multivariate* distributions, and will usually consider a vector of random variables (denoted by bold \mathbf{x}). ◇

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
 univariate
 multivariate

3199

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 (for example in Figure 6.2). The state space of the joint probability is the Cartesian product of the state spaces of each of the random variables. We define the *joint probability* as the entry of both values jointly

joint probability

$$P(x = x_i, y = y_i) = \frac{n_{ij}}{N}, \quad (6.9)$$

where n_{ij} is the number of events with x_i and y_j and N the total number of events. The joint probability is probability of the intersection of both events, that is $P(x = x_i, y = y_j) = P(x = x_i \cap y = y_j)$. Figure 6.2 illustrates the *probability mass function* (pmf) of a discrete probability distribution. For two random variables x and y , the probability that $x = x_i$ and $y = y_j$ 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.

Example 6.2

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 frequencies for the i^{th} column, that is $c_i = \sum_{j=1}^3 n_{ij}$. Similarly, 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

$$P(x = x_i) = \frac{c_i}{N} = \frac{\sum_{j=1}^3 n_{ij}}{N} \quad (6.10)$$

and

$$P(y = y_j) = \frac{r_j}{N} = \frac{\sum_{i=1}^5 n_{ij}}{N}, \quad (6.11)$$

where c_i and r_j are the i^{th} column and j^{th} row of the probability table, respectively. By convention for discrete random variables with a finite number of events, we assume that probabilities sum up to one, that is

$$\sum_{i=1}^5 P(x = x_i) = 1 \quad \text{and} \quad \sum_{j=1}^3 P(y = y_j) = 1. \quad (6.12)$$

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

$$P(y = y_j | x = x_i) = \frac{n_{ij}}{c_i}, \quad (6.13)$$

and the conditional probability of x given y is

$$P(x = x_i | y = y_j) = \frac{n_{ij}}{r_j}. \quad (6.14)$$

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

3211 the instances where $x = \mathbf{x}$, then the fraction of instances (the conditional
 3212 probability) for which $y = \mathbf{y}$ is written (lazily) as $p(y | x)$.

3213 In machine learning, we use discrete probability distributions to model
 3214 *categorical variables*, i.e., variables that take a finite set of unordered val-
 3215 ues. These could be categorical features such as the degree taken at uni-
 3216 versity when used for predicting the salary of a person, or categorical la-
 3217 bels such as letters of the alphabet when doing handwritten recognition.
 3218 Discrete distributions are also often used to construct probabilistic models
 3219 that combine a finite number of continuous distributions. We will see the
 3220 Gaussian mixture model in Chapter 11.

categorical variables

3221 6.2.2 Continuous Probabilities

3222 We consider real valued random variables in this section, that is we con-
 3223 sider state spaces which are intervals of the real line \mathbb{R} . In this book we will
 3224 pretend that we can perform operations on real random variables as if we
 3225 have discrete probability spaces with finite states. However this simplifica-
 3226 tion is not precise for two situations: when we repeat something infinitely
 3227 often, and when we want to draw a point from an interval. The first sit-
 3228 uation arises when we discuss generalization error in machine learning
 3229 (Chapter 8). The second situation arises when we want to discuss contin-
 3230 uous distributions such as the Gaussian (Section 6.5). For our purposes,
 3231 the lack of precision allows a more brief introduction to probability.

3232 *Remark.* In continuous spaces there are two additional technicalities which
 3233 are counterintuitive. First the set of all subsets (used to define the event
 3234 space \mathcal{A} in Section 6.1) is not well behaved enough. \mathcal{A} needs to be re-
 3235 stricted to behave well under set complements, set intersections and set
 3236 unions. Second the size of a set (which in discrete spaces can be obtained
 3237 by counting the elements) turns out to be tricky. The size of a set is called
 3238 its measure, for example the cardinality of discrete sets, the length of an
 3239 interval in \mathbb{R} and the volume of a region in \mathbb{R}^d are all measures. Sets that
 3240 behave well under set operations and furthermore have a topology are
 3241 called a Borel σ -algebras. Betancourt (2018) details a careful construction
 3242 of probability spaces from set theory without being bogged down in tech-
 3243 nicalities. A reader interested in a more precise construction is referred to
 3244 Jacod and Protter (2004); Billingsley (1995). Further references can be
 3245 found in the further reading section. In this book, we consider real valued
 3246 random variables with their corresponding Borel σ -algebra. We consider
 3247 random variables with values in \mathbb{R}^d to be a vector of real valued random
 3248 variables. ◇

3249 **Definition 6.1** (Probability Density Function). A function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is
 3250 called a *probability density function* (pdf) if

probability density
function

3251 1 $\forall \mathbf{x} \in \mathbb{R}^D : f(\mathbf{x}) \geq 0$

2 Its integral exists and

$$\int_{\mathbb{R}^D} f(\mathbf{x}) d\mathbf{x} = 1. \quad (6.15)$$

Here, $\mathbf{x} \in \mathbb{R}^D$ is a (continuous) random variable. For probability mass functions (pmf) of discrete random variables the integral in (6.15) is replaced with a sum (see (6.12)).

$P(x = x)$ is a set of
measure zero.)

In contrast to discrete random variables, the probability of a continuous random variable x taking a particular value $P(x = x)$ is zero. However this does not mean that such events never occur.

cumulative
distribution function

Definition 6.2 (Cumulative Distribution Function). A *cumulative distribution function* (cdf) of a multivariate real-valued random variable $\mathbf{x} \in \mathbb{R}^D$ is given by

$$F_{\mathbf{x}}(\mathbf{x}) = P(x_1 \leq x_1, \dots, x_D \leq x_D), \quad (6.16)$$

where the right-hand side represents the probability that random variable x_i takes the value smaller than or equal to x_i . This can be expressed also as the integral of the probability density function so that

$$F_{\mathbf{x}}(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} f(\mathbf{x}) d\mathbf{x}. \quad (6.17)$$

6.2.3 Contrasting Discrete and Continuous Distributions

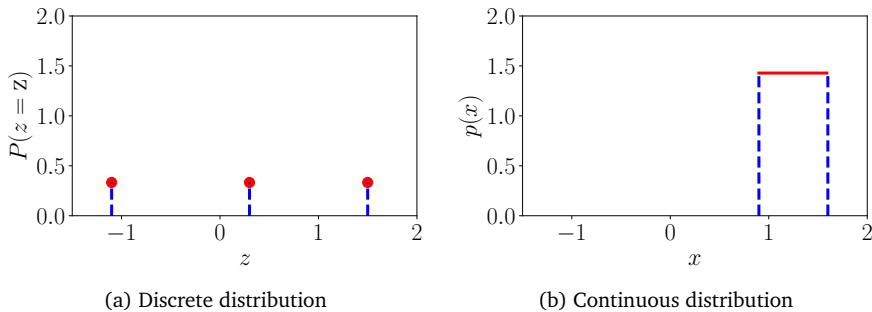
Let us consider both discrete and continuous distributions and contrast them. The aim here is to see that while both discrete and continuous distributions seem to have similar requirements, such as the total probability mass is 1, they are subtly different. Since the total probability mass of a discrete random variable is 1 (see (6.12)) and there are a finite number of states, the probability of each state must be in the interval $[0, 1]$. However, the analogous requirement for continuous random variables (see (6.15)) does not imply that the value of the density is less than or equal to 1 for all values. We illustrate this in Figure 6.3 using the *uniform distribution* for both discrete and continuous random variables.

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 chose numbers to drive home the point that we do not want to use (and should ignore) the ordering of

Figure 6.3
Examples of discrete
and continuous
probability distributions. See
Example 6.3 for
details of the
distributions.



the states. The probability mass function can be represented as a table of probability values.

z	-1.1	0.3	1.5
$P(z = z)$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Alternatively, one could think of this as a graph (Figure 6.3(a)), 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 Figure 6.3(a) is deliberately extended so that it is the same as in Figure 6.3(b).

Let x be a continuous random variable taking values in the range $0.9 \leq x \leq 1.6$, as represented by the graph in Figure 6.3(b). Observe that the height of the density can be greater than 1. However, it needs to hold that

$$\int_{0.9}^{1.6} p(x) dx = 1. \quad (6.18)$$

3269 3270 3271 3272 3273 3274 3275 3276

Remark. There is an additional subtlety with regards to discrete probability distributions. The states x_1, \dots, x_d do not in principle have any structure, that is there is usually no way to compare them, for example $x_1 = \text{red}$, $x_2 = \text{green}$, $x_3 = \text{blue}$. However in many machine learning applications the discrete states take numerical values, for example $x_1 = -1.1$, $x_2 = 0.3$, $x_3 = 1.5$, where we could say $x_1 < x_2 < x_3$. Discrete states which take numerical values are particularly useful because we often consider expected values (Section 6.4.1) of random variables. ◇

3277 3278 3279 3280 3281 3282

Unfortunately machine learning literature uses notation and nomenclature that hides the distinction between the state space, the set of outcomes and the random variable. For a value x of the set of possible outcomes of the random variable x , that is $x \in \mathcal{T}$, $p(x)$ denotes the probability that random variable x has the outcome x . For discrete random variables this is written as $P(x = x)$, which is known as the probability mass func-

Table 6.1
Nomenclature for probability distributions.

	“point probability”	“interval probability”
discrete	$P(x = x)$ probability mass function	not applicable
continuous	$p(x)$ probability density function	$P(x \leq x)$ cumulative distribution function

3283 This is often referred to as the “distribution”. For continuous vari-
 3284 ables, $p(x)$ is called the probability density function (often referred to as
 3285 a density), and to muddy things even further the cumulative distribution
 3286 function $P(x \leq x)$ is often also referred to as the “distribution”. In this
 3287 chapter, we will use the notation x or \mathbf{x} to refer to univariate and multi-
 3288 variate random variables respectively. We summarise the nomenclature in
 3289 Table 6.1.

3290 *Remark.* We will be using the expression “probability distribution” not
 3291 only for discrete probability mass functions but also for continuous prob-
 3292 ability density functions, although this is technically incorrect. Unfortu-
 3293 nately the majority of machine learning literature is also sloppy about the
 3294 phrase. \diamond

3295 6.3 Sum Rule, Product Rule and Bayes’ Theorem

3296 When we think of a probabilistic model as an extension to logical reason-
 3297 ing, as we discussed in Section 6.1.1, the rules of probability presented
 3298 here follow naturally from fulfilling the desiderata (Jaynes, 2003, Chapter
 3299 2). Probabilistic modeling provides a principled foundation for designing
 3300 machine learning methods. Once we have defined probability distribu-
 3301 tions (Section 6.2) corresponding to the uncertainties of the data and our
 3302 problem, it turns out that there are only two fundamental rules, the sum
 3303 rule and the product rule, that govern probabilistic inference.

3304 Given the definitions of marginal and conditional probability for dis-
 3305 crete and continuous random variables in the previous section, we can
 3306 now present the two fundamental rules in probability theory. These two
 3307 rules arise naturally (Jaynes, 2003) from the requirements we discussed
 3308 in Section 6.1.1. Recall from (6.9) that $p(\mathbf{x}, \mathbf{y})$ is the joint distribution of
 3309 the two random variables \mathbf{x}, \mathbf{y} , $p(\mathbf{x}), p(\mathbf{y})$ are the corresponding marginal
 3310 distributions, and $p(\mathbf{y} | \mathbf{x})$ is the conditional distribution of \mathbf{y} given \mathbf{x} .

sum rule The first rule, the *sum rule*, is

$$p(\mathbf{x}) = \begin{cases} \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}, \mathbf{y}) & \text{if } \mathbf{y} \text{ is discrete} \\ \int_{\mathcal{Y}} p(\mathbf{x}, \mathbf{y}) d\mathbf{y} & \text{if } \mathbf{y} \text{ is continuous} \end{cases} \quad (6.19)$$

marginalization so that we sum out (or integrate out) the set of states \mathbf{y} of the random
 property variable \mathbf{y} . The sum rule is also known as the *marginalization property*.

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. More concretely, if $\mathbf{x} = (x_1, \dots, x_D)$, we obtain the marginal

$$p(x_i) = \int p(x_1, \dots, x_D) d\mathbf{x}_{\setminus i} \quad (6.20)$$

3311 by repeated application of the sum rule where we integrate/sum out all
3312 random variables except x_i , which is indicated by $\setminus i$.

3313 *Remark.* Many of the computational challenges of probabilistic modeling
3314 are due to the application of the sum rule. When there are many variables
3315 or discrete variables with many states, the sum rule boils down to per-
3316 forming a high-dimensional sum or integral. Performing high dimensional
3317 sums or integrals is generally computationally hard, in the sense that there
3318 is no known polynomial time algorithm to calculate them exactly. \diamond

The second rule, known as the *product rule*, relates the joint distribution to the conditional distribution via product rule

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} | \mathbf{x})p(\mathbf{x}). \quad (6.21)$$

3319 The product rule can be interpreted as the fact that every joint distribu-
3320 tion of two random variables can be factorized (written as a product)
3321 of two other distributions. The two factors are the marginal distribu-
3322 tion of the first random variable $p(\mathbf{x})$, and the conditional distribution
3323 of the second random variable given the first $p(\mathbf{y} | \mathbf{x})$. Since the ordering
3324 of random variables is arbitrary in $p(\mathbf{x}, \mathbf{y})$ the product rule also implies
3325 $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} | \mathbf{y})p(\mathbf{y})$. To be precise, (6.21) is expressed in terms of the
3326 probability mass functions for discrete random variables. For continuous
3327 random variables, the product rule is expressed in terms of the probability
3328 density functions (Section 6.2.3).

3329 Let us briefly explore how to use probabilistic models to capture uncer-
3330 tainty (Ghahramani, 2015). At the lowest modeling level, measurement
3331 noise introduces model uncertainty, for example a camera sensor intro-
3332 duces random error in the value of each pixel it records. We will see in
3333 Chapter 9 how to use Gaussian (Section 6.5) noise models for linear re-
3334 gression. At higher modeling levels, we would be interested in modeling
3335 the uncertainty of the coefficients in linear regression. This uncertainty
3336 captures which values of these parameters will be good at predicting new
3337 data. Finally at the highest levels, we may want to capture uncertainties
3338 about the model structure. We will discuss model choice in Section 8.5.
3339 Once we have the probabilistic models (described in Section 8.3), the
3340 basic rules of probability presented in this section are used to infer the
3341 unobserved quantities given the observed data.

In machine learning and Bayesian statistics, we are often interested in

Bayes' theorem is also called the "probabilistic inverse"
Bayes' theorem

making inferences of unobserved (latent) random variables given that we have observed other random variables. Let us assume we have some prior knowledge $p(\mathbf{x})$ about an unobserved random variable \mathbf{x} and some relationship $p(\mathbf{y} | \mathbf{x})$ between \mathbf{x} and a second random variable \mathbf{y} , which we can observe. If we observe \mathbf{y} we can use Bayes' theorem to draw some conclusions about \mathbf{x} given the observed values of \mathbf{y} . *Bayes' theorem* (also: *Bayes' rule* or *Bayes' law*)

$$\underbrace{p(\mathbf{x} | \mathbf{y})}_{\text{posterior}} = \frac{\overbrace{p(\mathbf{y} | \mathbf{x})}^{\text{likelihood}} \overbrace{p(\mathbf{x})}^{\text{prior}}}{\underbrace{p(\mathbf{y})}_{\text{evidence}}} \quad (6.22)$$

is a direct consequence of the product rule in (6.19) since

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} | \mathbf{y})p(\mathbf{y}) \quad (6.23)$$

and

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} | \mathbf{x})p(\mathbf{x}) \quad (6.24)$$

so that

$$p(\mathbf{x} | \mathbf{y})p(\mathbf{y}) = p(\mathbf{y} | \mathbf{x})p(\mathbf{x}) \iff p(\mathbf{x} | \mathbf{y}) = \frac{p(\mathbf{y} | \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \quad (6.25)$$

3342 prior In (6.22), $p(\mathbf{x})$ is the *prior*, which encapsulates our subjective prior
3343 knowledge of the unobserved (latent) variable \mathbf{x} before observing any
3344 data. We can choose any prior that makes sense to us, but it is critical to
3345 ensure that the prior has a non-zero pdf (or pmf) on all plausible \mathbf{x} , even
3346 if they are very rare.

3347 likelihood The *likelihood* $p(\mathbf{y} | \mathbf{x})$ describes how \mathbf{x} and \mathbf{y} are related, and it is the
3348 probability of the data \mathbf{y} if we were to know the latent variable \mathbf{x} . Note
3349 that the likelihood is not a distribution in \mathbf{x} , but only in \mathbf{y} . We call $p(\mathbf{y} | \mathbf{x})$
3350 either the "likelihood of \mathbf{x} (given \mathbf{y})" or the "probability of \mathbf{y} given \mathbf{x} " but
3351 never the likelihood of \mathbf{y} (MacKay, 2003a).

3352 posterior The *posterior* $p(\mathbf{x} | \mathbf{y})$ is the quantity of interest in Bayesian statistics
3353 because it expresses exactly what we are interested in, i.e., what we know
3354 about \mathbf{x} after having observed \mathbf{y} .

The quantity

$$p(\mathbf{y}) := \int p(\mathbf{y} | \mathbf{x})p(\mathbf{x})d\mathbf{x} = \mathbb{E}_{\mathbf{x}}[p(\mathbf{y} | \mathbf{x})] \quad (6.26)$$

3355 marginal likelihood/evidence is the *marginal likelihood/evidence*. By definition the marginal likelihood
3356 integrates the numerator of (6.22) with respect to the latent variable \mathbf{x} .
3357 Therefore, the marginal likelihood is independent of \mathbf{x} and it ensures
3358 that the posterior $p(\mathbf{x} | \mathbf{y})$ is normalized. The marginal likelihood can also
3359 be interpreted as the expected likelihood where we take the expectation
3360 with respect to the prior $p(\mathbf{x})$. Beyond normalization of the posterior the

3361 marginal likelihood also plays an important role in Bayesian model selec-
3362 tion as we will discuss in Section 8.5. Due to the integration in (8.41), the
3363 evidence is often hard to compute.

3364 Bayes' theorem in (6.22) allows us to invert the causal relationship be-
3365 tween x and y given by the likelihood. Therefore, Bayes' theorem is some-
3366 times called the *probabilistic inverse*.

probabilistic inverse

3367 *Remark.* In Bayesian statistics, the posterior distribution is the quantity
3368 of interest as it encapsulates all available information from the prior and
3369 the data. Instead of carrying the posterior around, it is possible to fo-
3370 cus on some statistic of the posterior, such as the maximum of the pos-
3371 terior, which we will discuss in Section 9.2.3. However, focusing on the
3372 some statistic of the posterior leads to loss of information. If we think in
3373 a bigger context, then the posterior can be used within a decision mak-
3374 ing system, and having the full posterior around can be extremely useful
3375 and lead to decisions that are robust to disturbances. For example, in the
3376 context of model-based reinforcement learning, Deisenroth et al. (2015)
3377 show that using the full posterior distribution of plausible transition func-
3378 tions leads to very fast (data/sample efficient) learning, whereas focusing
3379 on the maximum of the posterior leads to consistent failures. Therefore,
3380 having the full posterior around in a downstream task can be very use-
3381 ful. In Chapter 9, we will continue this discussion in the context of linear
3382 regression. ◇

3383 6.4 Summary Statistics and Independence

3384 We are often interested in summarizing sets of random variables and com-
3385 paring pairs of random variables. A statistic of a random variable is a de-
3386 terministic function of that random variable. The summary statistics of a
3387 distribution provides one useful view of how a random variable behaves,
3388 and as the name suggests, provides numbers that summarize and charac-
3389 terize the distribution. We describe the mean and the variance, two well-
3390 known summary statistics. Then we discuss two ways to compare a pair
3391 of random variables: first how to say that two random variables are inde-
3392 pendent, and second how to compute an inner product between them.

3393 6.4.1 Means and Covariances

3394 Mean and (co)variance are often useful to describe properties of probabili-
3395 ty distributions (expected values and spread). We will see in Section 6.6
3396 that there is a useful family of distributions (called the exponential fam-
3397 ily), where the statistics of the random variable capture all possible infor-
3398 mation.

3399 The main tool we use to compute statistics of a random variable is its
3400 expected value with respect to a particular function.

expected value

Definition 6.3 (Expected value). The *expected value* of a function $g : \mathbb{R} \rightarrow \mathbb{R}$ of a univariate continuous random variable $x \sim p(x)$ is given by

$$\mathbb{E}_x[g(x)] = \int g(x)p(x)dx. \quad (6.27)$$

Correspondingly the expected value of a function g of a discrete random variable $x \sim p(x)$ is given by

$$\mathbb{E}_x[g(x)] = \sum_{x \in \mathcal{T}} g(x)p(x) \quad (6.28)$$

where \mathcal{T} is the set of possible outcomes of the random variable x .

In this section, we consider discrete random variables to have outcomes which are numerical. This can be seen by observing that the function g takes real numbers as inputs.

Remark. We consider multivariate random variables \mathbf{x} as a finite vector of univariate random variables $[x_1, \dots, x_n]^\top$. For multivariate random variables, we define the expected value element wise

$$\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})] = \begin{bmatrix} \mathbb{E}_{x_1}[g(x_1)] \\ \vdots \\ \mathbb{E}_{x_D}[g(x_D)] \end{bmatrix} \in \mathbb{R}^D, \quad (6.29)$$

where the subscript \mathbb{E}_{x_d} indicates that we are taking the expected value with respect to the d^{th} element of the vector \mathbf{x} . \diamond

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).

mean

Definition 6.3 defines the meaning of the notation \mathbb{E}_x and $\mathbb{E}_{\mathbf{x}}$ as the operator indicating that we should take the integral with respect to the probability density (for continuous distributions) or the sum over all states (for discrete distributions). The definition of the mean (Definition 6.4), is a special case of the expected value, obtained by choosing g to be the identity function.

Definition 6.4 (Mean). The *mean* of a random variable $\mathbf{x} \in \mathbb{R}^D$ is an average and defined as

$$\mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \begin{bmatrix} \mathbb{E}_{x_1}[x_1] \\ \vdots \\ \mathbb{E}_{x_D}[x_D] \end{bmatrix} \in \mathbb{R}^D, \quad (6.30)$$

where

$$\mathbb{E}_{x_d}[x_d] := \begin{cases} \int x_d p(x_d) dx_d & \text{if } x_d \text{ has a continuous domain} \\ \sum_{x_i} x_i p(x_d = x_i) & \text{if } x_d \text{ has a discrete domain} \end{cases} \quad (6.31)$$

for $d = 1, \dots, D$, where the subscript d indicates the corresponding dimension of \mathbf{x} , and the integral and sum are over the states x_i .

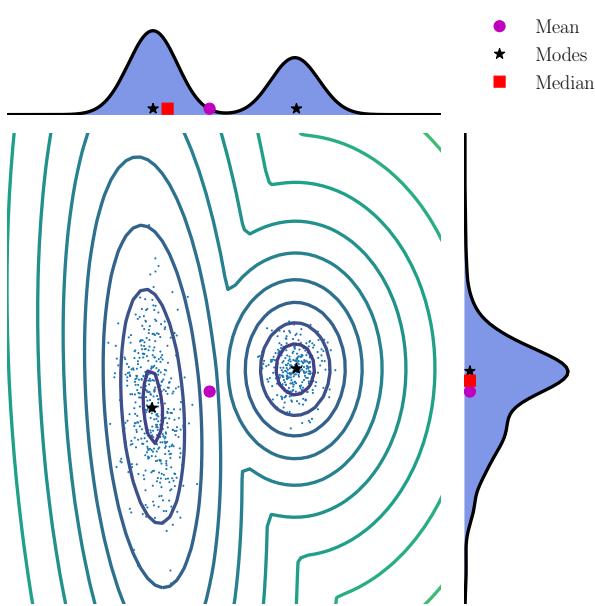


Figure 6.4
Illustration of the mean, mode and median for a two-dimensional dataset, as well as its marginal densities.

3415 In one dimension, there are two other intuitive notions of “average”,
 3416 which are the *median* and the *mode*. The median is the “middle” value if
 3417 we sort the values, i.e., 50% of the values are greater than the median and
 3418 50% are smaller than the median. This idea can be generalized to contin-
 3419 uous values by considering the value where the CDF (Definition 6.2) is
 3420 0.5. For distributions which are asymmetric or have long tails, the median
 3421 provides an estimate of a typical value that is closer to human intuition
 3422 than the mean value. Furthermore the median is more robust to outliers
 3423 than the mean. The generalization of the median to higher dimensions is
 3424 non-trivial as there is no obvious way to “sort” in more than one dimen-
 3425 sion (Hallin et al., 2010; Kong and Mizera, 2012). The *mode* is the most
 3426 frequently occurring value. For a discrete random variable, the mode is
 3427 defined as the value of x having the highest frequency of occurrence. For
 3428 a continuous random random variable, the mode is defined as a peak in
 3429 the density $p(x)$. A particular density $p(x)$ may have more than one mode,
 3430 and, therefore, finding the mode may be computationally challenging in
 3431 high dimensions.

median
mode

mode

Example 6.4

Consider the 2 dimensional distribution illustrated in Figure 6.4

$$\mathcal{N} \left(\begin{bmatrix} 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) + 1.5\mathcal{N} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & 2.9 \\ -1.1 & 0.7 \end{bmatrix} \right). \quad (6.32)$$

Also shown is its corresponding marginal distribution in each dimension.

Observe that the distribution is bimodal (has two modes), but one of the marginal distributions is unimodal (has one mode). The horizontal bimodal univariate distribution illustrates the fact that the mean and median can be quite different from each other. While it is tempting to define the two dimensional median to be the concatenation of the medians in each dimension, the fact that we cannot define an ordering of two dimensional points makes it difficult. When we say cannot define an ordering, we mean that there is more than one way to define $<$ such that $\begin{bmatrix} 1 \\ 0 \end{bmatrix} < \begin{bmatrix} 2 \\ 3 \end{bmatrix}$.

3432 The mean is recovered if we set the function g in Definition 6.3 to the
3433 identity function. This indicates that we can think about functions of ran-
3434 dom variables, which we will revisit in Section 6.7.

Remark. The expected value is a linear operator. For example, given multivariate real-valued functions $f(\mathbf{x}) = ag(\mathbf{x}) + bh(\mathbf{x})$ where $a, b \in \mathbb{R}$,

$$\mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x} \quad (6.33a)$$

$$= \int [ag(\mathbf{x}) + bh(\mathbf{x})]p(\mathbf{x})d\mathbf{x} \quad (6.33b)$$

$$= a \int g(\mathbf{x})p(\mathbf{x})d\mathbf{x} + b \int h(\mathbf{x})p(\mathbf{x})d\mathbf{x} \quad (6.33c)$$

$$= a\mathbb{E}_{\mathbf{x}}[g(\mathbf{x})] + b\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})]. \quad (6.33d)$$

3435



3436 For two random variables, we may wish to characterize their correspon-
3437 dence to each other. The covariance intuitively represents the notion of
3438 how dependent random variables are to one another.

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

$$\text{Cov}[x, y] = \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])]. \quad (6.34)$$

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, i.e.,

$$\text{Cov}[x, y] = \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]. \quad (6.35)$$

3439 variance The covariance of a variable with itself $\text{Cov}[x, x]$ is called the *variance* and
3440 standard deviation is denoted by $\text{V}[x]$. The square root of the variance is called the *standard*
3441 deviation and is often denoted by $\sigma(x)$.
3442 When we want to compare the covariances between different pairs of

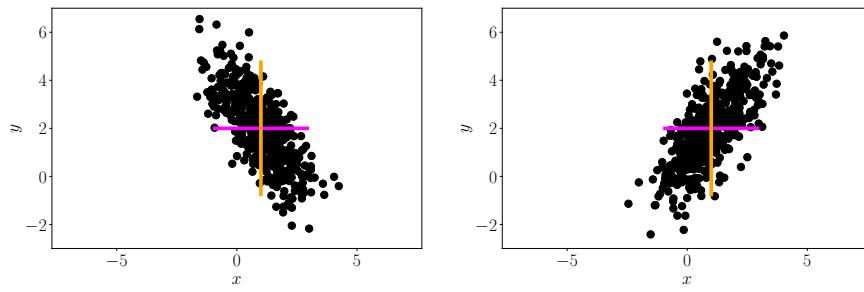
(a) x and y are negatively correlated.(b) x and y are positively correlated.

Figure 6.5
Two-dimensional datasets with identical means and variances along each axis (colored lines) but with different covariances.

3443 random variables, it turns out that the variance of each random variable
3444 affects the value of the covariance. The normalized version of covariance
3445 is called the correlation.

Definition 6.6 (Correlation). The *correlation* between two random variables x, y is given by

$$\text{corr}[x, y] = \frac{\text{Cov}[x, y]}{\sqrt{\text{V}[x]\text{V}[y]}}. \quad (6.36)$$

3446 The correlation matrix is the covariance matrix of standardized random
3447 variables, $x/\sigma(x)$. In other words, each random variable is divided by its
3448 standard deviation (the square root of the variance) in the correlation
3449 matrix.

3450 The covariance (and correlation) indicate how two random variables
3451 are related, see Figure 6.5. Positive correlation $\text{corr}[x, y]$ means that when
3452 x grows then y is also expected to grow. Negative correlation means that
3453 as x increases then y decreases.

3454 The notion of covariance can be generalised to multivariate random
3455 variables.

Definition 6.7 (Covariance). If we consider two random variables $x \in \mathbb{R}^D, y \in \mathbb{R}^E$, the *covariance* between x and y is defined as

$$\text{Cov}[x, y] = \mathbb{E}_{x,y}[xy^\top] - \mathbb{E}_x[x]\mathbb{E}_y[y]^\top = \text{Cov}[y, x]^\top \in \mathbb{R}^{D \times E}. \quad (6.37)$$

3456 Here, the subscript makes it explicit with respect to which variable we
3457 need to average.

3458 Definition 6.7 can be applied with the same multivariate random vari-
3459 able in both arguments, which results in a useful concept that intuitively
3460 captures the “spread” of a random variable.

Definition 6.8 (Variance). The *variance* of a random variable $x \in \mathbb{R}^D$ with mean vector $\mu \in \mathbb{R}^D$ is defined as

$$\text{V}_x[x] = \mathbb{E}_x[(x - \mu)(x - \mu)^\top] = \mathbb{E}_x[xx^\top] - \mathbb{E}_x[x]\mathbb{E}_x[x]^\top \quad (6.38a)$$

correlation

Terminology:
Covariance of
multivariate random
variables $\text{Cov}[x, y]$
is sometimes
referred to as
cross-covariance,
with covariance
referring to
 $\text{Cov}[x, x]$.
covariance

variance

$$= \begin{bmatrix} \text{Cov}[x_1, x_1] & \text{Cov}[x_1, x_2] & \dots & \text{Cov}[x_1, x_D] \\ \text{Cov}[x_2, x_1] & \text{Cov}[x_2, x_2] & \dots & \text{Cov}[x_2, x_D] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[x_D, x_1] & \dots & \dots & \text{Cov}[x_D, x_D] \end{bmatrix} \in \mathbb{R}^{D \times D}. \quad (6.38b)$$

covariance matrix 3461

This matrix is called the *covariance matrix* of the random variable \mathbf{x} . The covariance matrix is symmetric and positive definite and tells us something about the spread of the data.

marginal 3463

The covariance matrix contains the variances of the *marginals*

$$p(x_i) = \int p(x_1, \dots, x_D) dx_{\setminus i} \quad (6.39)$$

cross-covariance 3462

on its diagonal, where “ $\setminus i$ ” denotes “all variables but i ”. The off-diagonal entries are the *cross-covariance* terms $\text{Cov}[x_i, x_j]$ for $i, j = 1, \dots, D$, $i \neq j$. It generally holds that

$$\mathbb{V}_{\mathbf{x}}[\mathbf{x}] = \text{Cov}_{\mathbf{x}}[\mathbf{x}, \mathbf{x}]. \quad (6.40)$$

3464 We will revisit the idea of covariance again in Section 6.4.5.

3465

6.4.2 Empirical Means and Covariances

population mean 3466
and covariance 3467

The definitions in Section 6.4.1 are often also called the *population mean and covariance*, as it refers to the true statistics for the population. In machine learning we need to learn from empirical observations of data. Consider a random variable x . There are two conceptual steps to go from population statistics to the realization of empirical statistics. First we use the fact that we have a finite dataset (of size N) to construct an empirical statistic which is a function of a finite number of identical random variables, x_1, \dots, x_N . Second we observe the data, that is we look at the realization of each of the random variables x_1, \dots, x_N and apply the empirical statistic.

empirical mean 3476
sample mean 3477

Specifically for the mean (Definition 6.4), given a particular dataset 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 3478

Definition 6.9 (Empirical Mean and Covariance). The *empirical mean* vector is the arithmetic average of the observations for each variable, and it is defined as

$$\bar{\mathbf{x}} := \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n, \quad (6.41)$$

3479

where $\mathbf{x}_n \in \mathbb{R}^D$.

empirical covariance

Similar to the empirical mean, the *empirical covariance* matrix is a $D \times D$

matrix

$$\Sigma = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top. \quad (6.42)$$

In the above definition, we have expressed the statistic in terms of the multivariate random variables $\mathbf{x}_1, \dots, \mathbf{x}_N$. To compute the statistics for a particular dataset, we would use the realizations (observations) x_1, \dots, x_N and use (6.41) and (6.42). Empirical covariance matrices are symmetric, positive semi-definite (see Section 3.2.3).

Remark. The notation we use in this book is imprecise in the sense that we do not explicitly make a distinction between the random variable \mathbf{x}_n and its deterministic realization x_n . \diamond

Throughout the book we use the empirical covariance, which is a biased estimate. The unbiased (sometimes called corrected) covariance has the factor $N - 1$ in the denominator instead of N .

6.4.3 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 we need 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 μ , i.e.,

$$\mathbb{V}_x[x] := \mathbb{E}_x[(x - \mu)^2]. \quad (6.43)$$

Depending on whether x is a discrete or continuous random variable, the expectation in (6.43) and the mean $\mu = \mathbb{E}_x(x)$ are computed using (6.31). The variance as expressed in (6.43) is the mean of a new random variable $z := (x - \mu)^2$.

When estimating the variance in (6.43) empirically, we need to resort to a two-pass algorithm: one pass through the data to calculate the mean μ using (6.41), 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 (6.43) can be converted to the so-called *raw-score formula for variance*:

$$\mathbb{V}_x[x] = \mathbb{E}_x[x^2] - \mathbb{E}_x[x]^2. \quad (6.44)$$

The expression in (6.44) can be remembered as “the mean of the square minus the square of the mean”. It can be calculated empirically 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 can be numerically unstable. The raw-score version of the variance can be useful in machine learning, e.g., when deriving the bias-variance decomposition (Bishop, 2006).

A third way to understand the variance is that it is a sum of pairwise differences between all pairs of observations. Consider a sample x_1, \dots, x_N

raw-score formula for variance

The two terms can cancel out, resulting in a loss of numerical precision in floating point arithmetic.

of realizations of random variable x , and we compute the squared difference between pairs of x_i and x_j . By expanding the square we can show that the sum of N^2 pairwise differences is the empirical variance of the observations,

$$\frac{1}{N^2} \sum_{i,j=1}^N (x_i - x_j)^2 = 2 \left[\frac{1}{N} \sum_{i=1}^N x_i^2 - \left(\frac{1}{N} \sum_{i=1}^N x_i \right)^2 \right] \quad (6.45)$$

We see that (6.45) is twice the raw-score expression (6.44). 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. From a computational perspective, this means that by computing the mean (N terms in the summation), and then computing the variance (again N terms in the summation) we can obtain an expression (left-hand side of (6.45)) that has N^2 terms.

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 Sections 6.5 and 6.6), and hence may perform simple manipulations of random variables (such as adding two random variables).

Consider two random variables $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$. It holds that

$$\mathbb{E}[\mathbf{x} + \mathbf{y}] = \mathbb{E}[\mathbf{x}] + \mathbb{E}[\mathbf{y}] \quad (6.46)$$

$$\mathbb{E}[\mathbf{x} - \mathbf{y}] = \mathbb{E}[\mathbf{x}] - \mathbb{E}[\mathbf{y}] \quad (6.47)$$

$$\mathbb{V}[\mathbf{x} + \mathbf{y}] = \mathbb{V}[\mathbf{x}] + \mathbb{V}[\mathbf{y}] + \text{Cov}[\mathbf{x}, \mathbf{y}] + \text{Cov}[\mathbf{y}, \mathbf{x}] \quad (6.48)$$

$$\mathbb{V}[\mathbf{x} - \mathbf{y}] = \mathbb{V}[\mathbf{x}] + \mathbb{V}[\mathbf{y}] - \text{Cov}[\mathbf{x}, \mathbf{y}] - \text{Cov}[\mathbf{y}, \mathbf{x}] \quad (6.49)$$

Mean and (co)variance exhibit some useful properties when it comes to affine transformation of random variables. Consider a random variable \mathbf{x} with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ and a (deterministic) affine transformation $\mathbf{y} = \mathbf{Ax} + \mathbf{b}$ of \mathbf{x} . Then \mathbf{y} is itself a random variable whose mean vector and covariance matrix are given by

$$\mathbb{E}_{\mathbf{y}}[\mathbf{y}] = \mathbb{E}_{\mathbf{x}}[\mathbf{Ax} + \mathbf{b}] = \mathbf{A}\mathbb{E}_{\mathbf{x}}[\mathbf{x}] + \mathbf{b} = \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \quad (6.50)$$

$$\mathbb{V}_{\mathbf{y}}[\mathbf{y}] = \mathbb{V}_{\mathbf{x}}[\mathbf{Ax} + \mathbf{b}] = \mathbb{V}_{\mathbf{x}}[\mathbf{Ax}] = \mathbf{A}\mathbb{V}_{\mathbf{x}}[\mathbf{x}]\mathbf{A}^\top = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top, \quad (6.51)$$

respectively. Furthermore,

$$\text{Cov}[\mathbf{x}, \mathbf{y}] = \mathbb{E}[\mathbf{x}(\mathbf{Ax} + \mathbf{b})^\top] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{Ax} + \mathbf{b}]^\top \quad (6.52)$$

$$= \mathbb{E}[\mathbf{x}]\mathbf{b}^\top + \mathbb{E}[\mathbf{x}\mathbf{x}^\top]\mathbf{A}^\top - \boldsymbol{\mu}\mathbf{b}^\top - \boldsymbol{\mu}\boldsymbol{\mu}^\top\mathbf{A}^\top \quad (6.53)$$

$$= \boldsymbol{\mu}\mathbf{b}^\top - \boldsymbol{\mu}\mathbf{b}^\top + (\mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top)\mathbf{A}^\top \quad (6.54)$$

This can be shown directly by using the definition of the mean and covariance.

$$\stackrel{(6.38a)}{=} \Sigma A^\top, \quad (6.55)$$

3514 where $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top$ is the covariance of \mathbf{x} .

3515 **6.4.5 Statistical Independence**

Definition 6.10 (Independence). Two random variables \mathbf{x}, \mathbf{y} are *statistically independent* if and only if

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}). \quad (6.56)$$

3516 Intuitively, two random variables \mathbf{x} and \mathbf{y} are independent if the value
3517 of \mathbf{y} (once known) does not add any additional information about \mathbf{x} (and
3518 vice versa).

3519 If \mathbf{x}, \mathbf{y} are (statistically) independent then

- 3520 • $p(\mathbf{y} | \mathbf{x}) = p(\mathbf{y})$
- 3521 • $p(\mathbf{x} | \mathbf{y}) = p(\mathbf{x})$
- 3522 • $\mathbb{V}_{\mathbf{x}, \mathbf{y}}[\mathbf{x} + \mathbf{y}] = \mathbb{V}_{\mathbf{x}}[\mathbf{x}] + \mathbb{V}_{\mathbf{y}}[\mathbf{y}]$
- 3523 • $\text{Cov}_{\mathbf{x}, \mathbf{y}}[\mathbf{x}, \mathbf{y}] = \mathbf{0}$

3524 The last point above may not hold in converse, that is two random vari-
3525 ables can have covariance zero but are not statistically independent. To
3526 understand why, recall that covariance measures only linear dependence,
3527 therefore random variables that are non-linearly dependent could have
3528 covariance zero.

Example 6.5

Consider a random variable x with zero mean ($\mathbb{E}_x[x] = 0$) and also $\mathbb{E}_x[x^3] = 0$. Let $y = x^2$ (hence y is dependent on x) and consider the covariance (6.35) between x and y . But this gives

$$\text{Cov}[x, y] = \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y] = \mathbb{E}[x^3] = 0. \quad (6.57)$$

3529 In machine learning we often consider problems that can be modelled
3530 as *independent and identically distributed* random variables, x_1, \dots, x_N .
3531 The word “independent” refers to Definition 6.10, that is any pair of
3532 random variables x_i and x_j are independent. The phrase identically dis-
3533 tributed means that all the random variables are from the same distribu-
3534 tion.

independent and
identically
distributed

3535 Another concept that is important in machine learning is conditional
3536 independence.

Definition 6.11 (Conditional Independence). Formally, two random vari-
 ables \mathbf{x} and \mathbf{y} are *conditionally independent given \mathbf{z}* if and only if

$$p(\mathbf{x}, \mathbf{y} | \mathbf{z}) = p(\mathbf{x} | \mathbf{z})p(\mathbf{y} | \mathbf{z}) \quad \text{for all } \mathbf{z} \in \mathcal{A}. \quad (6.58)$$

conditionally
independent given \mathbf{z}

3537 We write $x \perp\!\!\!\perp y | z$.

3538 Definition 6.11 requires that the relation in (6.58) must hold true for
3539 every value of z . The interpretation of (6.58) can be understood as “given
3540 knowledge about z , the distribution of x and y factorizes”. Independence
3541 can be cast as a special case of conditional independence if we write $x \perp\!\!\!\perp$
3542 $y | \emptyset$.

By using the product rule of probability from (6.21) we can expand the left-hand side of (6.58) to obtain

$$p(x, y | z) = p(x | y, z)p(y | z). \quad (6.59)$$

By comparing the right-hand side of (6.58) with (6.59) we see that $p(y | z)$ appears in both of them so that

$$p(x | y, z) = p(x | z). \quad (6.60)$$

3543 Equation (6.60) provides an alternative definition of conditional indepen-
3544 dence, i.e., $x \perp\!\!\!\perp y | z$. This alternative presentation provides the interpre-
3545 tation “given that we know z , knowledge about y does not change our
3546 knowledge of x ”.

3547 6.4.6 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

$$\mathbb{V}[x + y] = \mathbb{V}[x] + \mathbb{V}[y] \quad (6.61)$$

3548 Since variances are measured in squared units, this looks very much like
3549 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 (6.61). Random variables can be considered vectors in a vector space, and we can define inner products to obtain geometric properties of random variables (Eaton, 2007). If we define

$$\langle x, y \rangle := \text{Cov}[x, y] \quad (6.62)$$

for zero mean random variables x and y , we obtain an inner product. we see that the covariance is symmetric, positive definite, and linear in either argument. The length of a random variable is

$$\|x\| = \sqrt{\text{Cov}[x, x]} = \sqrt{\mathbb{V}[x]} = \sigma[x], \quad (6.63)$$

i.e., its standard deviation. The “longer” the random variable, the more uncertain it is; and a random variable with length 0 is deterministic.

3550 $\text{Cov}[x, x] > 0$ and
3551 $0 \iff x = 0$
 $\text{Cov}[\alpha x + z, y] =$
 $\alpha \text{Cov}[x, y] +$
 $\text{Cov}[z, y]$ for $\alpha \in \mathbb{R}$.

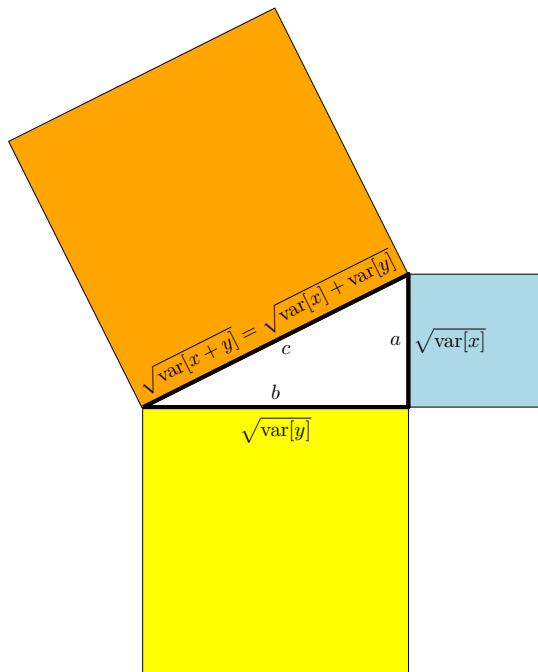


Figure 6.6
Geometry of random variables. If random variables x and y are uncorrelated they are orthogonal vectors in a corresponding vector space, and the Pythagorean theorem applies.

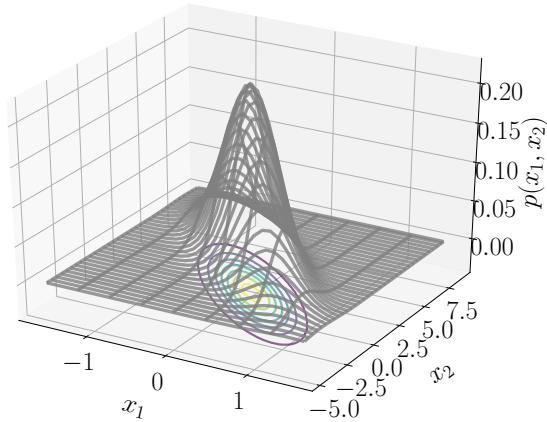
If we look at the angle θ between random two random variables x, y , we get

$$\cos \theta = \frac{\langle x, y \rangle}{\|x\| \|y\|} = \frac{\text{Cov}[x, y]}{\sqrt{\text{V}[x]\text{V}[y]}}, \quad (6.64)$$

which is the correlation (Definition 6.6) between the two random variables. This means that we can think of correlation as the angle between two random variables when we consider them geometrically. We know from Definition 3.7 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 $\text{Cov}[x, y] = 0$, i.e., they are uncorrelated. Figure 6.6 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 distributions. Recall that the probability mass (or density) is positive and needs to add up to 1. These constraints mean that distributions live on something called a statistical manifold. The study of this space of probability distributions is called information geometry. Computing distances between distributions are often done using Kullback-Leibler divergence which is a generalization of distances that account for properties of the statistical manifold. Just like the Euclidean distance is a special case of a metric (Section 3.3) the Kullback-Leibler divergence is a special case of two more general classes of divergences called Bregman divergences and

Figure 6.7
 Gaussian
 distribution of two
 random variables
 x, y .



³⁵⁷⁰ f -divergences. The study of divergences is beyond the scope of this book.
³⁵⁷¹ Interested readers are referred to a recent book (Amari, 2016) written by
³⁵⁷² one of the founders of the field of information geometry. ◇

The Gaussian ³⁵⁷³
 distribution arises ³⁵⁷⁴
 naturally when we ³⁵⁷⁵
 consider sums of ³⁵⁷⁶
 independent and ³⁵⁷⁷
 identically ³⁵⁷⁸
 distributed random ³⁵⁷⁹
 variables. This is ³⁵⁸⁰
 known as the ³⁵⁸¹
 Central Limit ³⁵⁸²
 Theorem (Grinstead and Snell, 1997). ³⁵⁸³
 normal distribution ³⁵⁸⁴

6.5 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 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

$$p(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right). \quad (6.65)$$

multivariate
 Gaussian
 distribution
 Also: multivariate
 normal distribution
 mean vector
 covariance matrix ³⁵⁸⁶
³⁵⁸⁷

The *multivariate Gaussian distribution* is fully characterized by a *mean vector* μ and a *covariance matrix* Σ and defined as

$$p(\mathbf{x} | \mu, \Sigma) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu)\right), \quad (6.66)$$

where $\mathbf{x} \in \mathbb{R}^D$ is a random variable. We write $\mathbf{x} \sim \mathcal{N}(\mathbf{x} | \mu, \Sigma)$ or $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$. Figure 6.7 shows a bi-variate Gaussian (mesh), with the

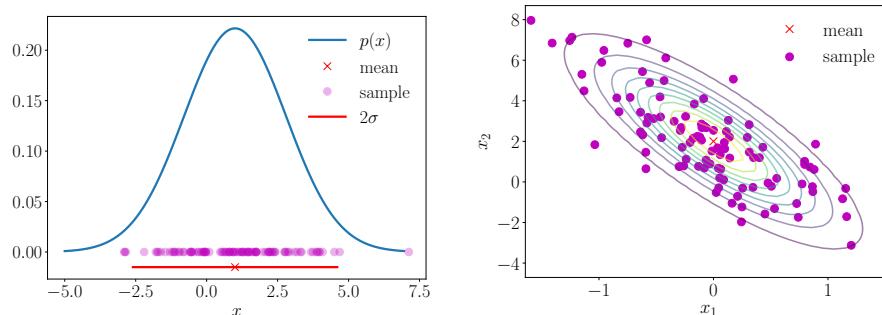


Figure 6.8
Gaussian distributions overlaid with 100 samples. Left: Univariate (1-dimensional) Gaussian; The red cross shows the mean and the red line shows the extent of the variance. Right: Multivariate (2-dimensional) Gaussian, viewed from top. The red standard normal cross shows the mean and the coloured lines shows the contour lines of the density.

corresponding contour plot. The special case of the Gaussian with zero mean and identity covariance, that is $\mu = \mathbf{0}$ and $\Sigma = \mathbf{I}$, is referred to as the *standard normal distribution*.

Gaussian distributions are widely used in statistical estimation and machine 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 6.7) 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.

6.5.1 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 \mathbf{x} and \mathbf{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 $[\mathbf{x}, \mathbf{y}]^\top$,

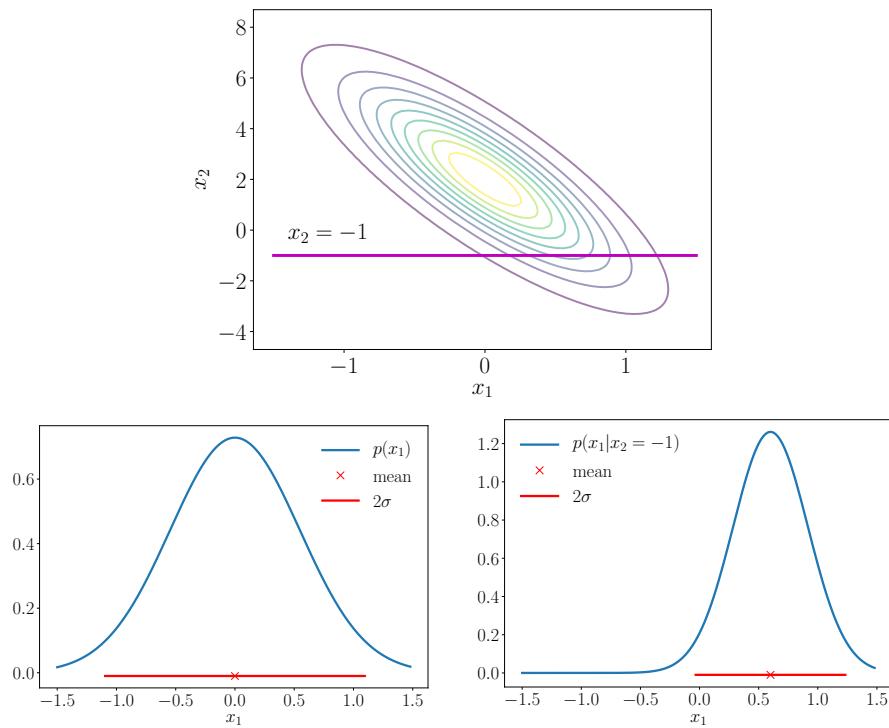
$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N} \left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right). \quad (6.67)$$

where $\Sigma_{xx} = \text{Cov}[\mathbf{x}, \mathbf{x}]$ and $\Sigma_{yy} = \text{Cov}[\mathbf{y}, \mathbf{y}]$ are the marginal covariance matrices of \mathbf{x} and \mathbf{y} , respectively, and $\Sigma_{xy} = \text{Cov}[\mathbf{x}, \mathbf{y}]$ is the cross-covariance matrix between \mathbf{x} and \mathbf{y} .

The conditional distribution $p(\mathbf{x} | \mathbf{y})$ is also Gaussian (illustrated on the bottom right of Figure 6.9) and given by (derived in Section 2.3 of Bishop (2006))

$$p(\mathbf{x} | \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_{x|y}, \Sigma_{x|y}) \quad (6.68)$$

Figure 6.9 Top: Bivariate Gaussian; Bottom left: Marginal of a joint Gaussian distribution is Gaussian; Bottom right: The conditional distribution of a Gaussian is also Gaussian



$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} (\mathbf{y} - \boldsymbol{\mu}_y) \quad (6.69)$$

$$\boldsymbol{\Sigma}_{x|y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx}. \quad (6.70)$$

3604 Note that in the computation of the mean in (6.69) the y -value is an
3605 observation and no longer random.

3606 *Remark.* The conditional Gaussian distribution shows up in many places,
3607 where we are interested in posterior distributions:

- 3608 • The Kalman filter (Kalman, 1960), one of the most central algorithms
3609 for state estimation in signal processing, does nothing but computing
3610 Gaussian conditionals of joint distributions (Deisenroth and Ohlsson,
3611 2011).
- 3612 • Gaussian processes (Rasmussen and Williams, 2006), which are a practical
3613 implementation of a distribution over functions. In a Gaussian process,
3614 we make assumptions of joint Gaussianity of random variables. By
3615 (Gaussian) conditioning on observed data, we can determine a posterior
3616 distribution over functions.
- 3617 • Latent linear Gaussian models (Roweis and Ghahramani, 1999; Murphy,
3618 2012), which include probabilistic PCA (Tipping and Bishop, 1999).

3619



The marginal distribution $p(\mathbf{x})$ of a joint Gaussian distribution $p(\mathbf{x}, \mathbf{y})$,

see (6.67), is itself Gaussian and computed by applying the sum-rule in (6.19) and given by

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}). \quad (6.71)$$

3620 The corresponding result holds for $p(\mathbf{y})$, which is obtained by marginalizing
3621 with respect to \mathbf{x} . Intuitively, looking at the joint distribution in (6.67),
3622 we ignore (i.e., integrate out) everything we are not interested in. This is
3623 illustrated on the bottom left of Figure 6.9.

Example 6.6

Consider the bivariate Gaussian distribution (illustrated in Figure 6.9)

$$p(x, y) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 0.3 & -1 \\ -1 & 5 \end{bmatrix}\right). \quad (6.72)$$

We can compute the parameters of the univariate Gaussian, conditioned on $y = -1$, by applying (6.69) and (6.70) to obtain the mean and variance respectively. Numerically, this is

$$\mu_{x|y=-1} = 0 + (-1)(0.2)(-1 - 2) = 0.6 \quad (6.73)$$

and

$$\sigma_{x|y=-1}^2 = 0.3 - (-1)(0.2)(-1) = 0.1. \quad (6.74)$$

Therefore the conditional Gaussian is given by

$$p(x | y = -1) = \mathcal{N}(0.6, 0.1). \quad (6.75)$$

The marginal distribution $p(x)$ in contrast can be obtained by applying (6.71), which is essentially using the mean and variance of the random variable x , giving us

$$p(x) = \mathcal{N}(0, 0.3) \quad (6.76)$$

6.5.2 Product of Gaussian Densities

3624 For linear regression (Chapter 9), we need to compute a Gaussian likelihood. 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 Gaussian densities. The *product* of two Gaussians $\mathcal{N}(\mathbf{x} | \mathbf{a}, \mathbf{A})\mathcal{N}(\mathbf{x} | \mathbf{b}, \mathbf{B})$ is a Gaussian distribution scaled by a $c \in \mathbb{R}$, given by $c\mathcal{N}(\mathbf{x} | \mathbf{c}, \mathbf{C})$ with

$$\mathbf{C} = (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1} \quad (6.77)$$

$$\mathbf{c} = \mathbf{C}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}) \quad (6.78)$$

$$c = (2\pi)^{-\frac{D}{2}} |\mathbf{A} + \mathbf{B}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{a} - \mathbf{b})^\top (\mathbf{A} + \mathbf{B})^{-1}(\mathbf{a} - \mathbf{b})\right). \quad (6.79)$$

3625 The scaling constant c itself can be written in the form of a Gaussian
3626 density either in \mathbf{a} or in \mathbf{b} with an “inflated” covariance matrix $\mathbf{A} + \mathbf{B}$,
3627 i.e., $c = \mathcal{N}(\mathbf{a} | \mathbf{b}, \mathbf{A} + \mathbf{B}) = \mathcal{N}(\mathbf{b} | \mathbf{a}, \mathbf{A} + \mathbf{B})$.

Remark. For notation convenience, we will sometimes use $\mathcal{N}(\mathbf{x} | \mathbf{m}, \mathbf{S})$ to describe the functional form of a Gaussian even if \mathbf{x} is not a random variable. We have just done this above when we wrote

$$c = \mathcal{N}(\mathbf{a} | \mathbf{b}, \mathbf{A} + \mathbf{B}) = \mathcal{N}(\mathbf{b} | \mathbf{a}, \mathbf{A} + \mathbf{B}). \quad (6.80)$$

3628 Here, neither \mathbf{a} nor \mathbf{b} are random variables. However, writing c in this way
3629 is more compact than (6.79). \diamond

3630 6.5.3 Sums and Linear Transformations

If \mathbf{x}, \mathbf{y} are independent Gaussian random variables (i.e., the joint is given as $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$) with $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_x, \Sigma_x)$ and $p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \boldsymbol{\mu}_y, \Sigma_y)$, then $\mathbf{x} + \mathbf{y}$ is also Gaussian distributed and given by

$$p(\mathbf{x} + \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{\mu}_y, \Sigma_x + \Sigma_y). \quad (6.81)$$

3631 Knowing that $p(\mathbf{x} + \mathbf{y})$ is Gaussian, the mean and covariance matrix can be
3632 determined immediately using the results from (6.46)–(6.49). This prop-
3633 erty will be important when we consider i.i.d. Gaussian noise acting on
3634 random variables as is the case for linear regression (Chapter 9).

Example 6.7

Since expectations are linear operations, we can obtain the weighted sum of independent Gaussian random variables

$$p(a\mathbf{x} + b\mathbf{y}) = \mathcal{N}(a\boldsymbol{\mu}_x + b\boldsymbol{\mu}_y, a^2\Sigma_x + b^2\Sigma_y). \quad (6.82)$$

3635 *Remark.* A case which will be useful in Chapter 11 is the weighted sum
3636 of Gaussian densities. This is different from the weighted sum of Gaussian
3637 random variables. \diamond

3638 In Theorem 6.12, the random variable x is from a density which a mix-
3639 ture of two densities $p_1(x)$ and $p_2(x)$, weighted by α . The theorem can
3640 be generalized to the multivariate random variable case, since linearity of
3641 expectations holds also for multivariate random variables. However the
3642 idea of a squared random variable needs to be replaced by xx^\top .

Theorem 6.12. Consider a weighted sum of two univariate Gaussian densities

$$p(x) = \alpha p_1(x) + (1 - \alpha)p_2(x) \quad (6.83)$$

3643 where the scalar $0 < \alpha < 1$ is the mixture weight, and $p_1(x)$ and $p_2(x)$ are

³⁶⁴⁴ univariate Gaussian densities (Equation (6.65)) with different parameters,
³⁶⁴⁵ that is $(\mu_1, \sigma_1^2) \neq (\mu_2, \sigma_2^2)$.

The mean of the mixture x is given by the weighted sum of the means of each random variable,

$$\mathbb{E}[x] = \alpha\mu_1 + (1 - \alpha)\mu_2. \quad (6.84)$$

The variance of the mixture x is the mean of the conditional variance and the variance of the conditional mean,

$$\mathbb{V}[x] = [\alpha\sigma_1^2 + (1 - \alpha)\sigma_2^2] + \left([\alpha\mu_1^2 + (1 - \alpha)\mu_2^2] - [\alpha\mu_1 + (1 - \alpha)\mu_2]^2 \right). \quad (6.85)$$

Proof The mean of the mixture x is given by the weighted sum of the means of each random variable. We apply the definition of the mean (Definition 6.4), and plug in our mixture (Equation (6.83)) above

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} xp(x)dx \quad (6.86a)$$

$$= \int_{-\infty}^{\infty} \alpha xp_1(x) + (1 - \alpha)xp_2(x)dx \quad (6.86b)$$

$$= \alpha \int_{-\infty}^{\infty} xp_1(x)dx + (1 - \alpha) \int_{-\infty}^{\infty} xp_2(x)dx \quad (6.86c)$$

$$= \alpha\mu_1 + (1 - \alpha)\mu_2. \quad (6.86d)$$

To compute the variance, we can use the raw score version of the variance (Equation (6.44)), 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 6.3).

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} x^2 p(x)dx \quad (6.87a)$$

$$= \int_{-\infty}^{\infty} \alpha x^2 p_1(x) + (1 - \alpha)x^2 p_2(x)dx \quad (6.87b)$$

$$= \alpha \int_{-\infty}^{\infty} x^2 p_1(x)dx + (1 - \alpha) \int_{-\infty}^{\infty} x^2 p_2(x)dx \quad (6.87c)$$

$$= \alpha(\mu_1^2 + \sigma_1^2) + (1 - \alpha)(\mu_2^2 + \sigma_2^2). \quad (6.87d)$$

³⁶⁴⁶ 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
³⁶⁴⁷ variable is the sum of the squared mean and the variance.

Therefore the variance is given by subtracting (6.86d) from (6.87d),

$$\mathbb{V}[x] = \mathbb{E}[x^2] - (\mathbb{E}[x])^2 \quad (6.88a)$$

$$= \alpha(\mu_1^2 + \sigma_1^2) + (1 - \alpha)(\mu_2^2 + \sigma_2^2) - (\alpha\mu_1 + (1 - \alpha)\mu_2)^2 \quad (6.88b)$$

$$= [\alpha\sigma_1^2 + (1 - \alpha)\sigma_2^2]$$

$$+ \left([\alpha\mu_1^2 + (1-\alpha)\mu_2^2] - [\alpha\mu_1 + (1-\alpha)\mu_2]^2 \right). \quad (6.88c)$$

3649 For a mixture, the individual components can be considered to be conditional distributions (conditioned on the component identity). The last line
3650 is an illustration of the conditional variance formula: “The variance of a
3651 mixture is the mean of the conditional variance and the variance of the
3652 conditional mean”. \square

3653
3654 *Remark.* The derivation above holds for any density, but in the case of
3655 the Gaussian since it is fully determined by the mean and variance, the
3656 mixture density can be determined in closed form. \diamond

3657 We consider in Example 6.18 a bivariate standard Gaussian random
3658 variable \mathbf{x} and performed a linear transformation \mathbf{Ax} on it. The outcome
3659 is a Gaussian random variable with zero mean and covariance \mathbf{AA}^\top . Ob-
3660serve that adding a constant vector will change the mean of the distribu-
3661tion, without affecting its variance, that is the random variable $\mathbf{x} + \boldsymbol{\mu}$ is
3662 Gaussian with mean $\boldsymbol{\mu}$ and identity covariance. Therefore, a linear (or
3663 affine) transformation of a Gaussian random variable is Gaussian dis-
3664tributed.

Consider a Gaussian distributed random variable $\mathbf{x} \sim \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \Sigma)$. For a given matrix \mathbf{A} of appropriate shape, let \mathbf{y} be a random variable $\mathbf{y} = \mathbf{Ax}$ which is a transformed version of \mathbf{x} . We can compute the mean of \mathbf{y} by using the fact that the expectation is a linear operator (Equation (6.50)) as follows:

$$\mathbb{E}[\mathbf{Ax}] = \mathbf{A}\mathbb{E}[\mathbf{x}] = \mathbf{A}\boldsymbol{\mu}. \quad (6.89)$$

Similarly the variance of \mathbf{y} can be found by using Equation (6.51):

$$\mathbb{V}[\mathbf{Ax}] = \mathbf{A}\mathbb{V}[\mathbf{x}]\mathbf{A}^\top = \mathbf{A}\Sigma\mathbf{A}^\top. \quad (6.90)$$

This means that the random variable \mathbf{y} is distributed according to

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{A}\boldsymbol{\mu}, \mathbf{A}\Sigma\mathbf{A}^\top). \quad (6.91)$$

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 full rank matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ where $m \geq n$, let $\mathbf{y} \in \mathbb{R}^m$ be a Gaussian random variable with mean \mathbf{Ax} , i.e.,

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{Ax}, \Sigma). \quad (6.92)$$

What is the corresponding probability distribution $p(\mathbf{x})$? If \mathbf{A} is invertible, then we can write $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$ and apply the transformation in the previous paragraph. However, in general \mathbf{A} is not invertible, and we use an approach similar to that of the pseudo-inverse (3.56). That is we premultiply both sides with \mathbf{A}^\top and then invert $\mathbf{A}^\top\mathbf{A}$ which is symmetric and positive definite, giving us the relation

$$\mathbf{y} = \mathbf{Ax} \iff (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{y} = \mathbf{x}. \quad (6.93)$$

Hence, \mathbf{x} is a linear transformation of \mathbf{y} , and we obtain

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{y}, (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \Sigma \mathbf{A} (\mathbf{A}^\top \mathbf{A})^{-1}). \quad (6.94)$$

3665 6.5.4 Sampling from Multivariate Gaussian Distributions

3666 We will not explain the subtleties of random sampling on a computer. In
 3667 the case of a multivariate Gaussian, this process consists of three stages:
 3668 first we need a source of pseudo-random numbers that provide a uniform
 3669 sample in the interval $[0,1]$, second we use a non-linear transformation
 3670 such as the Box-Müller transform (Devroye, 1986) to obtain a sample from
 3671 a univariate Gaussian, and third we collate a vector of these samples to
 3672 obtain a sample from a multivariate standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

3673 For a general multivariate Gaussian, that is where the mean is non-zero
 3674 and the covariance is not the identity matrix, we use the properties of
 3675 linear transformations of a Gaussian random variable. Assume we are
 3676 interested in generating samples $\mathbf{x}_i, i = 1, \dots, n$, from a multivariate Gaus-
 3677 sian distribution with mean μ and covariance matrix Σ . We would like
 3678 to construct the sample from a sampler that provides samples from the
 3679 multivariate standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

3680 To obtain samples from a multivariate normal $\mathcal{N}(\mu, \Sigma)$, we can use
 3681 the properties of a linear transformation of a Gaussian random variable:
 3682 If $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ then $\mathbf{y} = \mathbf{Ax} + \mu$, where $\mathbf{AA}^\top = \Sigma$, is Gaussian dis-
 3683 tributed with mean μ and covariance matrix Σ . Recall from Section 4.3
 3684 that when we can decompose $\Sigma = \mathbf{AA}^\top$, while there are many possible
 3685 decompositions, we often choose the Cholesky decomposition. This has
 3686 the benefit that \mathbf{A} is triangular, leading to efficient computation.

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.

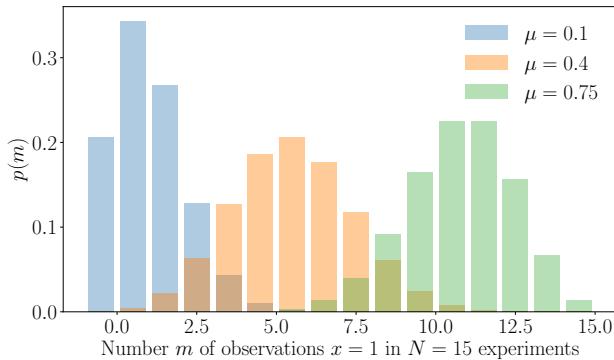
3687 6.6 Conjugacy and the Exponential Family

3688 Many of the probability distributions “with names” that we find in statis-
 3689 tics textbooks were discovered to model particular types of phenomena.
 3690 For example we have seen the Gaussian distribution in Section 6.5. The
 3691 distributions are also related to each other in complex ways (Leemis and
 3692 McQueston, 2008). For a beginner in the field, it can be overwhelming to
 3693 figure out which distribution to use. In addition, many of these distribu-
 3694 tions were discovered at a time that statistics and computation was done
 3695 by pencil and paper. It is natural to ask what are meaningful concepts
 3696 in the computing age (Efron and Hastie, 2016). In the previous section,
 3697 we saw that many of the operations required for inference can be conve-
 3698 niently calculated when the distribution is Gaussian. It is worth recalling
 3699 at this point the desiderata for manipulating probability distributions.

“Computers” were a job description.

- 3700 1 There is some “closure property” when applying the rules of probability,
 3701 e.g., Bayes’ theorem. By closure we mean that applying a particular
 3702 operation returns an object of the same type.

Figure 6.10
 Examples of the Binomial distribution for $\mu \in \{0.1, 0.4, 0.75\}$ and $N = 15$.



- 3703 2 As we collect more data, we do not need more parameters to describe
- 3704 the distribution.
- 3705 3 Since we are interested in learning from data, we want parameter esti-
- 3706 mation to behave nicely.

exponential family³⁷⁰⁷ It turns out that the class of distributions called the *exponential family*³⁷⁰⁸ 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, the Bernoulli (Example 6.8), Binomial (Example 6.9) and Beta (Example 6.10) distributions.³⁷¹¹³⁷¹²

Bernoulli distribution

Example 6.8

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 as

$$p(x | \mu) = \mu^x (1 - \mu)^{1-x}, \quad x \in \{0, 1\}, \quad (6.95)$$

$$\mathbb{E}[x] = \mu, \quad (6.96)$$

$$\mathbb{V}[x] = \mu(1 - \mu), \quad (6.97)$$

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.³⁷¹³³⁷¹⁴

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 occurrence of this is when expressing the Multinomial distribution. ◇³⁷¹⁵³⁷¹⁶³⁷¹⁷³⁷¹⁸

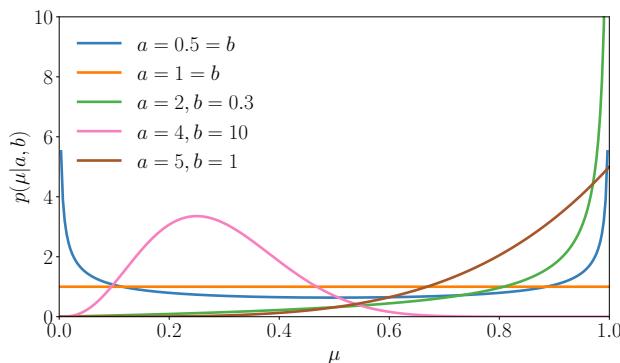


Figure 6.11
Examples of the Beta distribution for different values of α and β .

Example 6.9

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

$$p(m | N, \mu) = \binom{N}{m} \mu^m (1 - \mu)^{N-m}, \quad (6.98)$$

$$\mathbb{E}[m] = N\mu, \quad (6.99)$$

$$\mathbb{V}[m] = N\mu(1 - \mu) \quad (6.100)$$

where $\mathbb{E}[m]$ and $\mathbb{V}[m]$ are the mean and variance of m , respectively.

Binomial distribution

³⁷¹⁹ An example where the Binomial could be used is if we want to describe
³⁷²⁰ the probability of observing m “heads” in N coin-flip experiments if the
³⁷²¹ probability for observing head in a single experiment is μ .

Example 6.10

We may wish to model a continuous random variable on a finite interval. The *Beta distribution* is a distribution over a continuous random variable $\mu \in [0, 1]$, which is often used to represent the probability for some binary event (e.g., the parameter governing the Bernoulli distribution). The Beta distribution (illustrated in Figure 6.11) itself is governed by two parameters $\alpha > 0$, $\beta > 0$ and is defined as

$$p(\mu | \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1} (1 - \mu)^{\beta-1} \quad (6.101)$$

$$\mathbb{E}[\mu] = \frac{\alpha}{\alpha + \beta}, \quad \mathbb{V}[\mu] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \quad (6.102)$$

Beta distribution

where $\Gamma(\cdot)$ is the Gamma function defined as

$$\Gamma(t) := \int_0^\infty x^{t-1} \exp(-x) dx, \quad t > 0. \quad (6.103)$$

$$\Gamma(t+1) = t\Gamma(t). \quad (6.104)$$

Note that the fraction of Gamma functions in (6.101) normalizes the Beta distribution.

3722 Intuitively, α moves probability mass toward 1, whereas β moves prob-
 3723 ability mass toward 0. There are some special cases (Murphy, 2012):

- 3724 • For $\alpha = 1 = \beta$ we obtain the uniform distribution $\mathcal{U}[0, 1]$.
- 3725 • For $\alpha, \beta < 1$, we get a bimodal distribution with spikes at 0 and 1.
- 3726 • For $\alpha, \beta > 1$, the distribution is unimodal.
- 3727 • For $\alpha, \beta > 1$ and $\alpha = \beta$, the distribution is unimodal, symmetric and
 3728 centered in the interval $[0, 1]$, i.e., the mode/mean is at $\frac{1}{2}$.

3729 *Remark.* There is a whole zoo of distributions with names, and they are
 3730 related in different ways to each other (Leemis and McQueston, 2008).

3731 It is worth keeping in mind that each named distribution is created for a
 3732 particular reason, but may have other applications. Knowing the reason
 3733 behind the creation of a particular distribution often allows insight into
 3734 how to best use it. We introduced the above three distributions to be able
 3735 to illustrate the concepts of conjugacy (Section 6.6.1) and exponential
 3736 families (Section 6.6.3). \diamond

3737 6.6.1 Conjugacy

3738 According to Bayes' theorem (6.22), the posterior is proportional to the
 3739 product of the prior and the likelihood. The specification of the prior can
 3740 be tricky for two reasons: First, the prior should encapsulate our knowl-
 3741 edge about the problem before we see some data. This is often difficult to
 3742 describe. Second, it is often not possible to compute the posterior distribu-
 3743 tion analytically. However, there are some priors that are computationally
 3744 convenient: *conjugate priors*.

conjugate priors 3744 **Definition 6.13** (Conjugate Prior). A prior is *conjugate* for the likelihood
 3745 function if the posterior is of the same form/type as the prior.

3746 Conjugacy is particularly convenient because we can algebraically cal-
 3747 culate our posterior distribution by updating the parameters of the prior
 3748 distribution.

3749 *Remark.* When considering the geometry of probability distributions, con-
 3750 jugate priors retain the same distance structure as the likelihood (Agarwal
 3751 and III, 2010). \diamond

3753 To introduce a concrete example of conjugate priors, we describe below
3754 the Binomial distribution (defined on discrete random variables) and the
3755 Beta distribution (defined on continuous random variables).

Example 6.11 (Beta-Binomial Conjugacy)

Consider a Binomial random variable $x \sim \text{Bin}(N, \mu)$ where

$$p(x | N, \mu) = \binom{N}{x} \mu^x (1 - \mu)^{N-x} \quad x = 0, 1, \dots, N \quad (6.105)$$

is the probability of finding x times the outcome “head” in N coin flips, where μ is the probability of a “head”. We place a Beta prior on the parameter μ , that is $\mu \sim \text{Beta}(\alpha, \beta)$ where

$$p(\mu | \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1} (1 - \mu)^{\beta-1} \quad (6.106)$$

If we now observe some outcome $x = h$, that is we see h heads in N coin flips, we compute the posterior distribution on μ as

$$p(\mu | x = h, N, \alpha, \beta) \propto p(x | N, \mu)p(\mu | \alpha, \beta) \quad (6.107a)$$

$$= \mu^h (1 - \mu)^{(N-h)} \mu^{\alpha-1} (1 - \mu)^{\beta-1} \quad (6.107b)$$

$$= \mu^{h+\alpha-1} (1 - \mu)^{(N-h)+\beta-1} \quad (6.107c)$$

$$\propto \text{Beta}(h + \alpha, N - h + \beta) \quad (6.107d)$$

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.

In the following example, we will derive a result that is similar to the Beta-Binomial conjugacy result. Here we will show that the Beta distribution is a conjugate prior for the Bernoulli distribution.

Example 6.12 (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

$$p(\theta | x, \alpha, \beta) = p(x | \theta) \times p(\theta | \alpha, \beta) \quad (6.108a)$$

$$\propto \theta^x (1 - \theta)^{1-x} \times \theta^{\alpha-1} (1 - \theta)^{\beta-1} \quad (6.108b)$$

$$= \theta^{\alpha+x-1} (1 - \theta)^{\beta+(1-x)-1} \quad (6.108c)$$

$$\propto p(\theta | \alpha + x, \beta + (1 - x)). \quad (6.108d)$$

Table 6.2 Examples of conjugate priors for common likelihood functions.

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

The last line above is the Beta distribution with parameters $(\alpha + x, \beta + (1 - x))$.

3756 Table 6.2 lists examples for conjugate priors for the parameters of some
 3757 standard likelihoods used in probabilistic modeling. Distributions such as
 3758 Multinomial, inverse Gamma, inverse Wishart, and Dirichlet can be found
 3759 in any statistical text, and are for example described in Bishop (2006).

3760 The Beta distribution is the conjugate prior for the parameter μ in both
 3761 the Binomial and the Bernoulli likelihood. For a Gaussian likelihood function,
 3762 we can place a conjugate Gaussian prior on the mean. The reason
 3763 why the Gaussian likelihood appears twice in the table is that we need
 3764 to distinguish the univariate from the multivariate case. In the univariate
 Alternatively, the 3765 (scalar) case, the inverse Gamma is the conjugate prior for the variance.
 Gamma prior is 3766 In the multivariate case, we use a conjugate inverse Wishart distribution
 conjugate for the 3767 as a prior on the covariance matrix. The Dirichlet distribution is the conju-
 precision (inverse 3768 gate prior for the multinomial likelihood function. For further details, we
 variance) in the 3769 refer to Bishop (2006).

Alternatively, the 3770 Wishart prior is
 Wishart prior is 3771 conjugate for the
 conjugate for the 3772 precision matrix
 precision matrix 3773 (inverse covariance
 (inverse covariance 3774 matrix) in the
 Gaussian likelihood 3775 discovered the notion of *sufficient statistics*: the idea that there are statistics
 3776 that will contain all available information that can be inferred from data
 3777 corresponding to the distribution under consideration. In other words suf-
 3778 ficient statistics carry all the information needed to make inference about
 3779 the population, that is they are the statistics that are sufficient to represent
 3780 the distribution.

3781 For a set of distributions parameterized by θ , let x be a random vari-
 3782 able with distribution given an unknown θ_0 . A vector $\phi(x)$ of statistics
 3783 are called sufficient statistics for θ_0 if they contain all possible informa-
 3784 tion about θ_0 . To be more formal about “contain all possible information”:
 3785 this means that the probability of x given θ can be factored into a part
 3786 that does not depend on θ , and a part that depends on θ only via $\phi(x)$.

6.6.2 Sufficient Statistics

Recall that a statistic of a random variable is a deterministic function of that random variable. For example if $x = [x_1, \dots, x_N]^\top$ is a vector of univariate Gaussian random variables, that is $x_n \sim \mathcal{N}(\mu, \sigma^2)$, then the sample mean $\hat{\mu} = \frac{1}{N}(x_1 + \dots + x_N)$ 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.

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 $\phi(x)$.

3787 The Fisher-Neyman factorization theorem formalizes this notion, which
3788 we state below without proof.

Theorem 6.14 (Fisher-Neyman). *Let x have probability density function $p(x | \theta)$. Then the statistics $\phi(x)$ are sufficient for θ if and only if $p(x | \theta)$ can be written in the form*

$$p(x | \theta) = h(x)g_\theta(\phi(x)). \quad (6.109)$$

3789 where $h(x)$ is a distribution independent of θ and g_θ captures all the dependence
3790 on θ via sufficient statistics $\phi(x)$.

3791 If $p(x | \theta)$ does not depend on θ then $\phi(x)$ is trivially a sufficient statistic
3792 for any function ϕ . The more interesting case is that $p(x | \theta)$ is dependent
3793 only on $\phi(x)$ and not x itself. In this case, $\phi(x)$ is a sufficient statistic for
3794 x .

3795 In machine learning we consider a finite number of samples from a
3796 distribution. One could imagine that for simple distributions (such as the
3797 Bernoulli in Example 6.8) we only need a small number of samples to
3798 estimate the parameters of the distributions. We could also consider the
3799 opposite problem: if we have a set of data (a sample from an unknown
3800 distribution), which distribution gives the best fit? A natural question to
3801 ask is as we observe more data, do we need more parameters θ to de-
3802 scribe the distribution? It turns out that the answer is yes in general, and
3803 this is studied in non-parametric statistics (Wasserman, 2007). A converse
3804 question is to consider which class of distributions have finite dimensional
3805 sufficient statistics, that is the number of parameters needed to describe
3806 them do not increase arbitrarily. The answer is exponential family distri-
3807 butions, described in the following section.

3808 6.6.3 Exponential Family

3809 There are three possible levels of abstraction we can have when con-
3810 sidering distributions (of discrete or continuous random variables). At
3811 level one (the most concrete end of the spectrum), we have a particu-
3812 lar named distribution with fixed parameters, for example a univariate
3813 Gaussian $\mathcal{N}(0, 1)$ with zero mean and unit variance. In machine learning
3814 we often use the second level of abstraction, that is we fix the paramet-
3815 ric form (the univariate Gaussian) and infer the parameters from data. For
3816 example, we assume a univariate Gaussian $\mathcal{N}(\mu, \sigma^2)$ with unknown mean
3817 μ and unknown variance σ^2 , and use a maximum likelihood fit to deter-
3818 mine the best parameters (μ, σ^2) . We will see an example of this when
3819 considering linear regression in Chapter 9. A third level of abstraction is
3820 to consider families of distributions, and in this book, we consider the ex-
3821ponential family. The univariate Gaussian is an example of a member of
3822 the exponential family. Many of the widely used statistical models, includ-

ing 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). ◇

An *exponential family* is a family of probability distributions, parameterized by $\theta \in \mathbb{R}^D$, of the form

$$p(\mathbf{x} | \boldsymbol{\theta}) = h(\mathbf{x}) \exp (\langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle - A(\boldsymbol{\theta})) , \quad (6.110)$$

where $\phi(\mathbf{x})$ is the vector of sufficient statistics. In general, any inner product (Section 3.2) can be used in (6.110), and for concreteness we will use the standard dot product here ($\langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle = \boldsymbol{\theta}^\top \phi(\mathbf{x})$). Note that the form of the exponential family is essentially a particular expression of $g_\theta(\phi(\mathbf{x}))$ in the Fisher-Neyman theorem (Theorem 6.14).

The factor $h(\mathbf{x})$ can be absorbed into the dot product term by adding another entry ($\log h(\mathbf{x})$) to the vector of sufficient statistics $\phi(\mathbf{x})$, and constraining the corresponding parameter $\theta_0 = 1$. 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

$$p(\mathbf{x} | \boldsymbol{\theta}) \propto \exp (\boldsymbol{\theta}^\top \phi(\mathbf{x})) . \quad (6.111)$$

For this form of parameterization, the parameters $\boldsymbol{\theta}$ are called the *natural parameters*. 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 based on the fact that we can capture information about data in $\phi(\mathbf{x})$.

Example 6.13 (Gaussian as Exponential Family)

Consider the univariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Let $\phi(\mathbf{x}) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$. Then by using the definition of the exponential family,

$$p(\mathbf{x} | \boldsymbol{\theta}) \propto \exp(\theta_1 x + \theta_2 x^2) . \quad (6.112)$$

Setting

$$\boldsymbol{\theta} = \left[\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2} \right]^\top \quad (6.113)$$

and substituting into (6.112) we obtain

$$p(x | \boldsymbol{\theta}) \propto \exp\left(\frac{\mu x}{\sigma^2} - \frac{x^2}{2\sigma^2}\right) \propto \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right). \quad (6.114)$$

Therefore, the univariate Gaussian distribution is a member of the exponential family with sufficient statistic $\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$.

Example 6.14 (Bernoulli as Exponential Family)

Recall the Bernoulli distribution from Example 6.8

$$p(x | \mu) = \mu^x (1 - \mu)^{1-x}, \quad x \in \{0, 1\}. \quad (6.115)$$

This can be written in exponential family form

$$p(x | \mu) = \exp[\log(\mu^x (1 - \mu)^{1-x})] \quad (6.116)$$

$$= \exp[x \log \mu + (1 - x) \log(1 - \mu)] \quad (6.117)$$

$$= \exp[x \log \mu - x \log(1 - \mu) + \log(1 - \mu)] \quad (6.118)$$

$$= \exp\left[x \log \frac{\mu}{1 - \mu} + \log(1 - \mu)\right]. \quad (6.119)$$

The last line (6.119) can be identified as being in exponential family form (6.110) by observing that

$$h(x) = 1 \quad (6.120)$$

$$\theta = \log \frac{\mu}{1 - \mu} \quad (6.121)$$

$$\phi(x) = x \quad (6.122)$$

$$A(\theta) = -\log(1 - \mu) = \log(1 + \exp(\theta)). \quad (6.123)$$

The relationship between θ and μ is invertible,

$$\mu = \frac{1}{1 + \exp(-\theta)}. \quad (6.124)$$

This relation is used to obtain the right equality of (6.123).

³⁸⁴³ *Remark.* The relationship between the original Bernoulli parameter μ and the natural parameter θ is known as the *sigmoid* or logistic function. Observe that $\mu \in (0, 1)$ but $\theta \in \mathbb{R}$, and therefore the sigmoid function squeezes a real value into the range $(0, 1)$. This property is useful in machine learning, for example it is used in logistic regression (Bishop, 1995, Section 4.3.2), as well as as a nonlinear activation functions in neural networks (Goodfellow et al., 2016, Chapter 6). ◇

sigmoid

It is often not obvious how to find the parametric form of the conju-

gate distribution of a particular distribution. Exponential families provide a convenient way to find conjugate pairs of distributions. Consider the random variable \mathbf{x} distributed as an exponential family (6.110)

$$p(\mathbf{x} | \boldsymbol{\theta}) = h(\mathbf{x}) \exp (\langle \boldsymbol{\theta}, \phi(\mathbf{x}) \rangle - A(\boldsymbol{\theta})) . \quad (6.125)$$

Every exponential family has a conjugate prior (Brown, 1986)

$$p(\boldsymbol{\theta} | \boldsymbol{\gamma}) = h_c(\boldsymbol{\theta}) \exp \left(\left\langle \begin{bmatrix} \boldsymbol{\gamma}_1 \\ \boldsymbol{\gamma}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\theta} \\ -A(\boldsymbol{\theta}) \end{bmatrix} \right\rangle - A_c(\boldsymbol{\gamma}) \right) , \quad (6.126)$$

where $\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1 \\ \boldsymbol{\gamma}_2 \end{bmatrix}$ has dimension $\dim(\boldsymbol{\theta}) + 1$. The sufficient statistics of the conjugate prior are $\begin{bmatrix} \boldsymbol{\theta} \\ -A(\boldsymbol{\theta}) \end{bmatrix}$. By using the knowledge of the general form of conjugate priors for exponential families, we can derive functional forms of conjugate priors corresponding to particular distributions.

Example 6.15

Recall the exponential family form of the Bernoulli distribution (6.119),

$$p(x | \mu) = \exp \left[x \log \frac{\mu}{1-\mu} + \log(1-\mu) \right] . \quad (6.127)$$

The canonical conjugate prior therefore has the same form

$$p(\mu | \boldsymbol{\gamma}, n_0) = \exp \left[n_0 \boldsymbol{\gamma} \log \frac{\mu}{1-\mu} + n_0 \log(1-\mu) - A_c(\boldsymbol{\gamma}, n_0) \right] , \quad (6.128)$$

which simplifies to

$$p(\mu | \boldsymbol{\gamma}, n_0) = \exp [n_0 \boldsymbol{\gamma} \log \mu + n_0 (1-\boldsymbol{\gamma}) \log(1-\mu) - A_c(\boldsymbol{\gamma}, n_0)] . \quad (6.129)$$

Putting this in non-exponential family form

$$p(\mu | \boldsymbol{\gamma}, n_0) \propto \mu^{n_0 \boldsymbol{\gamma}} (1-\mu)^{n_0 (1-\boldsymbol{\gamma})} \quad (6.130)$$

which is of the same form as the Beta distribution (6.101), with minor manipulations to get the original parametrization (Example 6.12).

Observe that in this example we have derived the form of the Beta distribution by looking at the conjugate prior of the exponential family.

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

3861 Gaussian). From an optimization perspective, the log-likelihood function
3862 is concave allowing for efficient optimization approaches to be applied
3863 (Chapter 7).

3864 6.7 Change of Variables/Inverse Transform

3865 It may seem that there are very many known distributions, but in re-
3866 ality the set of distributions for which we have names is quite limited.
3867 Therefore, it is often useful to understand how transformed random vari-
3868 ables are distributed. For example, assume that x is a random variable
3869 distributed according to the univariate normal distribution $\mathcal{N}(0, 1)$, what
3870 is the distribution of x^2 ? Another example, which is quite common in ma-
3871 chine learning, is: given that x_1 and x_2 are univariate standard normal,
3872 what is the distribution of $\frac{1}{2}(x_1 + x_2)$?

3873 One option to work out the distribution of $\frac{1}{2}(x_1 + x_2)$ is to calculate the
3874 mean and variance of x_1 and x_2 and then combine them. As we saw in
3875 Section 6.4.4, we can calculate the mean and variance of resulting ran-
3876 dom variables when we consider affine transformations of random vari-
3877 ables. However, we may not be able to obtain the functional form of the
3878 distribution under transformations. Furthermore, we may be interested
3879 in nonlinear transformations of random variables for which closed-form
3880 expressions are not readily available.

3881 *Remark* (Notation). In this section, we will be explicit about random vari-
3882 ables and the values they take. Hence, we will use small letters x, y to
3883 denote random variables and small capital letters X, Y to denote the values
3884 that the random variables take. We will explicitly write probability mass
3885 functions (pmf) of discrete random variables x as $P(x = x)$. For contin-
3886 uous random variables x , the probability density function (pdf) is written
3887 as $f(x)$ and the cumulative distribution function (cdf) is written as
3888 $F_x(x \leq x)$. \diamond

3889 We will look at two approaches for obtaining distributions of transfor-
3890 mations of random variables: a direct approach using the definition of a
3891 cumulative distribution function and a change-of-variable approach that
3892 uses the chain rule of calculus (Section 5.2.2). The change-of-variable ap-
3893 proach is widely used because it provides a “recipe” for attempting to
3894 compute the resulting distribution due to a transformation. We will ex-
3895 plain the techniques for univariate random variables, and will only briefly
3896 provide the results for the general case of multivariate random variables.

3897 As mentioned in Section 6.1, random variables and probability distri-
3898 butions are closely associated with each other. It is worth carefully teasing
3899 apart the two ideas, and in doing so we will motivate why we need to
3900 transform random variables.

Moment generating functions can also be used to study transformations of random variables (Casella and Berger, 2002, Chapter 2).

Example 6.16

Consider a medical test that returns the number of cancerous cells that can be found in the biopsy. The state space is 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 ?

Transformations of discrete random variables can be understood directly. Given a discrete random variable x with probability mass function (pmf) $P(x = x)$ (Section 6.2.1), and an invertible function $U(x)$. Consider the transformed random variable $y := U(x)$, with pmf $P(y = y)$. Then

$$P(y = y) = P(U(x) = y) \quad \text{transformation of interest} \quad (6.131a)$$

$$= P(x = U^{-1}(y)) \quad \text{inverse} \quad (6.131b)$$

- 3901 where we can observe that $x = U^{-1}(y)$. Therefore for discrete random
 3902 variables, transformations directly change the individual events (with the
 3903 probabilities appropriately transformed).

6.7.1 Distribution Function Technique

- 3904
 3905 The distribution function technique goes back to first principles, and uses
 3906 the definition of a cumulative distribution function (cdf) $F_x(x) = P(x \leq x)$ and the fact that its differential is the probability density function (pdf)
 3907 $f(x)$ (Wasserman, 2004, Chapter 2). For a random variable x and a func-
 3908 tion U , we find the pdf of the random variable $y := U(x)$ by
 3909

- 1 Finding the cdf:

$$F_y(y) = P(y \leq y) \quad (6.132)$$

- 2 Differentiating the cdf $F_y(y)$ to get the pdf $f(y)$.

$$f(y) = \frac{d}{dy} F_y(y). \quad (6.133)$$

- 3910 We also need to keep in mind that the domain of the random variable may
 3911 have changed due to the transformation by U .

Example 6.17

Let x be a continuous random variable with probability density function on $0 \leq x \leq 1$

$$f(x) = 3x^2. \quad (6.134)$$

We are interested in finding the pdf of $y = x^2$.

The function f is an increasing function of x , and the resulting value of y lies in the interval $[0, 1]$. We obtain

$$F_y(Y) = P(y \leq Y) \quad \text{definition of cdf} \quad (6.135a)$$

$$= P(x^2 \leq Y) \quad \text{transformation of interest} \quad (6.135b)$$

$$= P(x \leq Y^{\frac{1}{2}}) \quad \text{inverse} \quad (6.135c)$$

$$= F_x(Y^{\frac{1}{2}}) \quad \text{definition of cdf} \quad (6.135d)$$

$$= \int_0^{Y^{\frac{1}{2}}} 3t^2 dt \quad \text{cdf as a definite integral} \quad (6.135e)$$

$$= [t^3]_{t=0}^{t=Y^{\frac{1}{2}}} \quad \text{result of integration} \quad (6.135f)$$

$$= Y^{\frac{3}{2}}, \quad 0 \leq Y \leq 1. \quad (6.135g)$$

Therefore, the cdf of y is

$$F_y(Y) = Y^{\frac{3}{2}} \quad (6.136)$$

for $0 \leq Y \leq 1$. To obtain the pdf, we differentiate the cdf

$$f(y) = \frac{d}{dy} F_y(Y) = \frac{3}{2} y^{\frac{1}{2}} \quad (6.137)$$

for $0 \leq y \leq 1$.

3912 In Example 6.17, we considered a strictly monotonically increasing function $f(x) = 3x^2$. This means that we could compute an inverse function.
3913 3914 In general, we require that the function of interest $y = U(x)$ has an inverse $x = U^{-1}(y)$. A useful result can be obtained by considering the cumulative distribution function $F_x(x)$ of a random variable x , and using it as the transformation $U(x)$. This leads to the following theorem which is called the probability integral transform. The result is the basis of generating samples from distributions whose cdfs are known by first generating a sample from a uniform distribution and then transforming it by the inverse cdf.

Functions that have inverses are called injective functions (Section 2.7).

Theorem 6.15. *This is Theorem 2.1.10 in Casella and Berger (2002). Let x be a continuous random variable with a strictly monotonic cumulative distribution function $F_x(\cdot)$. Then the random variable y defined as*

$$y = F_x(x), \quad (6.138)$$

3922 *has a uniform distribution.*

3923 *Proof* We need to show that the cumulative distribution function of y defines a distribution of a uniform random variable. Recall that by the axioms of probability (Section 6.1) probabilities must be non-negative and

3926 sum/integrate to one. Therefore, the range of possible values of $y = F_x(x)$
 3927 is the interval $[0, 1]$. For any $F_x(\cdot)$, the inverse $F_x^{-1}(\cdot)$ exists because we
 3928 assumed that $F_x(\cdot)$ is strictly monotonically increasing, which we will use
 3929 in the following.

Given any continuous random variable x , the definition of a cdf gives

$$F_y(y) = P(y \leq Y) \quad (6.139a)$$

$$= P(F_x(x) \leq Y) \quad \text{transformation of interest} \quad (6.139b)$$

$$= P(x \leq F_x^{-1}(Y)) \quad \text{inverse exists} \quad (6.139c)$$

$$= F_x(F_x^{-1}(Y)) \quad \text{definition of cdf} \quad (6.139d)$$

$$= Y, \quad (6.139e)$$

3930 where the last line is due to the fact that $F_x(\cdot)$ composed with its inverse
 3931 results in an identity transformation. The statement $F_y(y) = y$ along with
 3932 the fact that y lies in the interval $[0, 1]$ means that $F_y(\cdot)$ is the cdf of the
 3933 uniform random variable on the unit interval. \square

probability integral
transform 3934
3935 Theorem 6.15 is known as the *probability integral transform*, and it is used to derive algorithms for sampling from distributions by transforming the result of sampling from a uniform random variable (Bishop, 2006). 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).

6.7.2 Change of Variables

3942 The distribution function technique in Section 6.7.1 is derived from first
 3943 principles, based on the definitions of cdfs and using properties of in-
 3944 verses, differentiation and integration. This argument from first principles
 3945 relies on two facts:

- 3946 1 We can transform the cdf of y into an expression that is a cdf of x .
 3947 2 We can differentiate the cdf to obtain the pdf.

3948 Let us break down the reasoning step by step, with the goal of understand-
 3949 ing the more general change of variables approach in Theorem 6.16.

Remark. The name change of variables comes from the idea of changing the variable of integration when faced with a difficult integral. For univariate functions, we use the substitution rule of integration,

$$\int f(g(x))g'(x)dx = \int f(u)du \quad \text{where } u = g(x). \quad (6.140)$$

3950 The derivation of this rule is based on the chain rule of calculus (5.31) and
 3951 by applying twice the fundamental theorem of calculus. The fundamental
 3952 theorem of calculus formalizes the fact that integration and differentiation

are somehow “inverses” of each other. An intuitive understanding of the rule can be obtained by thinking (loosely) about small changes (differentials) to the equation $u = g(x)$. That is by considering $\Delta u = g'(x)\Delta x$ as a differential of $u = g(x)$. By substituting $u = g(x)$, the argument inside the integral on the right hand side of (6.140) becomes $f(g(x))$. By pretending that the term du can be approximated by $du \approx \Delta u = g'(x)\Delta x$, and that $dx \approx \Delta x$, we obtain (6.140). \diamond

Consider a function of a random variable $y = U(x)$, where $x \in [a, b]$. By the definition of the cdf, we have

$$F_y(Y) = P(y \leq Y). \quad (6.141)$$

We are interested in a function U of the random variable

$$P(y \leq Y) = P(U(x) \leq Y), \quad (6.142)$$

where we assume that the function U is invertible. By applying the inverse U^{-1} to the arguments of $P(U(x) \leq Y)$, we obtain

$$P(U(x) \leq Y) = P(U^{-1}(U(x)) \leq U^{-1}(Y)) = P(x \leq U^{-1}(Y)), \quad (6.143)$$

which is an expression of the cdf of x . Recall the definition of the cdf in terms of the pdf

$$P(x \leq U^{-1}(Y)) = \int_a^{U^{-1}(Y)} f(x) dx. \quad (6.144)$$

Now we have an expression of the cdf of y in terms of x :

$$F_y(Y) = \int_a^{U^{-1}(Y)} f(x) dx. \quad (6.145)$$

To obtain the pdf, we differentiate (6.145) with respect to y .

$$f(y) = \frac{d}{dy} F_y(Y) = \frac{d}{dy} \int_a^{U^{-1}(Y)} f(x) dx \quad (6.146)$$

Note that the integral on the right hand side is with respect to x , but we need an integral with respect to y because we are differentiating with respect to y . In particular we use (6.140) to get the substitution

$$\int f(U^{-1}(y)) U^{-1}'(y) dy = \int f(x) dx \quad \text{where } x = U^{-1}(y). \quad (6.147)$$

Using (6.147) on the right hand side of (6.146) gives us

$$f(y) = \frac{d}{dy} \int_a^{U^{-1}(Y)} f_x(U^{-1}(y)) U^{-1}'(y) dy. \quad (6.148)$$

We then recall that differentiation is a linear operator and we use the

subscript x to remind ourselves that $f_x(U^{-1}(y))$ is a function of x and not y . Invoking the fundamental theorem of calculus again gives us

$$f(y) = f_x(U^{-1}(y)) \times \left| \frac{d}{dy} U^{-1}(y) \right|. \quad (6.149)$$

change of variable³⁹⁶⁰ This is called the *change of variable* technique. The term $\left| \frac{d}{dy} U^{-1}(y) \right|$ ³⁹⁶¹ measures how much a unit volume changes when applying U (see also³⁹⁶² the Remark on page 151. In (6.149) we introduced the absolute value of³⁹⁶³ the differential. For decreasing functions, it turns out that an additional³⁹⁶⁴ negative sign is needed, and instead of having two types of change-of-³⁹⁶⁵ variable rules, the absolute value unifies both of them.

³⁹⁶⁶ *Remark.* In comparison to the discrete case in (6.131b), we have an addi-³⁹⁶⁷ tional factor $\left| \frac{d}{dy} U^{-1}(y) \right|$. The continuous case requires more care because³⁹⁶⁸ $P(y = y) = 0$ for all y . The probability density function $f(y)$ does not³⁹⁶⁹ have a description as a probability of an event involving y . \diamond

³⁹⁷⁰ So far in this section we have been studying univariate change of vari-³⁹⁷¹ ables. The case for multivariate random variables is analogous, but com-³⁹⁷² plicated by fact that the absolute value cannot be used for multivariate³⁹⁷³ functions. Instead we use the determinant of the Jacobian matrix. Recall³⁹⁷⁴ from (5.56) that the Jacobian is a matrix of partial derivatives, and that³⁹⁷⁵ the existence of a non-zero determinant shows that we can invert the Ja-³⁹⁷⁶ cobian. Recall the discussion in Section 4.1 that the determinant arises³⁹⁷⁷ because our differentials (cubes of volume) are transformed into paral-³⁹⁷⁸ lelepipeds by the Jacobian. Let us summarize the discussion above in the³⁹⁷⁹ following theorem, which gives us a recipe for multivariate change of vari-³⁹⁸⁰ ables.

Theorem 6.16. [Theorem 17.2 in Billingsley (1995)] Let $f(\mathbf{x})$ be the value of the probability density of the multivariate continuous random variable \mathbf{x} . If the vector-valued function $\mathbf{y} = U(\mathbf{x})$ is differentiable and invertible for all values within the domain of \mathbf{x} , then for corresponding values of \mathbf{y} , the probability density of $\mathbf{y} = U(\mathbf{x})$ is given by

$$f(\mathbf{y}) = f_{\mathbf{x}}(U^{-1}(\mathbf{y})) \times \left| \det \left(\frac{\partial}{\partial \mathbf{y}} U^{-1}(\mathbf{y}) \right) \right|. \quad (6.150)$$

³⁹⁸¹ The theorem looks intimidating at first glance, but the key point is that³⁹⁸² a change of variable of a multivariate random variable follows the pro-³⁹⁸³ cedure of the univariate change of variable. First we need to work out³⁹⁸⁴ the inverse transform, and substitute that into the density of \mathbf{x} . Then we³⁹⁸⁵ calculate the determinant of the Jacobian and multiply the result. The³⁹⁸⁶ following example illustrates the case of a bivariate random variable.

Example 6.18

Consider a bivariate random variable $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ with probability density function

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2\pi} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right). \quad (6.151)$$

We use the change-of-variable technique from Theorem 6.16 to derive the effect of a linear transformation (Section 2.7) of the random variable. Consider a matrix $\mathbf{A} \in \mathbb{R}^{2 \times 2}$ defined as

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \quad (6.152)$$

We are interested in finding the probability density function of the transformed bivariate random variable $\mathbf{y} = \mathbf{Ax}$.

Recall that for change of variables we require the inverse transformation of \mathbf{x} as a function of \mathbf{y} . Since we consider linear transformations, the inverse transformation is given by the matrix inverse (see Section 2.2.2). For 2×2 matrices, we can explicitly write out the formula, given by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}. \quad (6.153)$$

Observe that $ad - bc$ is the determinant (Section 4.1) of \mathbf{A} . The corresponding probability density function is given by

$$f(\mathbf{x}) = f(\mathbf{A}^{-1}\mathbf{y}) = \frac{1}{2\pi} \exp\left(-\frac{1}{2} \mathbf{y}^\top \mathbf{A}^{-\top} \mathbf{A}^{-1} \mathbf{y}\right). \quad (6.154)$$

The partial derivative of a matrix times a vector with respect to the vector is the matrix itself (Section 5.5) and, therefore,

$$\frac{\partial}{\partial \mathbf{y}} \mathbf{A}^{-1} \mathbf{y} = \mathbf{A}^{-1}. \quad (6.155)$$

Recall from Section 4.1 that the determinant of the inverse is the inverse of the determinant so that the determinant of the Jacobian matrix is given by

$$\det\left(\frac{\partial}{\partial \mathbf{y}} \mathbf{A}^{-1} \mathbf{y}\right) = \frac{1}{ad - bc}. \quad (6.156)$$

We are now able to apply the change-of-variable formula from Theorem 6.16 by multiplying (6.154) with (6.156), which yields

$$f(\mathbf{y}) = f(\mathbf{x}) \left| \det\left(\frac{\partial}{\partial \mathbf{y}} \mathbf{A}^{-1} \mathbf{y}\right) \right| \quad (6.157a)$$

$$= \frac{1}{2\pi} \exp\left(-\frac{1}{2} \mathbf{y}^\top \mathbf{A}^{-\top} \mathbf{A}^{-1} \mathbf{y}\right) |ad - bc|^{-1}. \quad (6.157b)$$

3987 While Example 6.18 is based on a bivariate random variable, which
 3988 allows to easily compute the matrix inverse, the relation above holds for
 3989 higher dimensions.

3990 *Remark.* We saw in Section 6.5 that the density $f(\mathbf{x})$ above is actually
 3991 the standard Gaussian distribution, and the transformed density $f(\mathbf{y})$ is a
 3992 bivariate Gaussian with covariance $\Sigma = \mathbf{A}\mathbf{A}^\top$. \diamond

3993 6.8 Further Reading

3994 This chapter is rather terse at times, Grinstead and Snell (1997) and
 3995 Walpole et al. (2011) provides more relaxed presentations that are suit-
 3996 able for self study. Readers interested in more philosophical aspects of
 3997 probability should consider Hacking (2001), whereas a more software
 3998 engineering approach is presented by Downey (2014). An overview of
 3999 exponential families can be found in Barndorff-Nielsen (2014). We will
 4000 see more about how to use probability distributions to model machine
 4001 learning tasks in Chapter 8. Ironically the recent surge in interest in neu-
 4002 ral networks has resulted in a broader appreciation of probabilistic mod-
 4003 els. For example the idea of normalizing flows (Rezende and Mohamed,
 4004 2015) relies on change of variables for transforming random variables.
 4005 An overview of methods for variational inference as applied to neural net-
 4006 works is described in Chapters 16 to 20 of Goodfellow et al. (2016).

4007 We side stepped a large part of the difficulty in continuous random vari-
 4008 ables by avoiding measure theoretic questions (Billingsley, 1995; Pollard,
 4009 2002), and by assuming without construction that we have real numbers,
 4010 and ways of defining sets on real numbers as well as their appropriate fre-
 4011 quency of occurrence. These details do matter, for example in the specifi-
 4012 cation of conditional probability $p(y | x)$ for continuous random variables
 4013 x, y (Proschan and Presnell, 1998). The lazy notation hides the fact that
 4014 we want to specify that $x = \mathbf{x}$ (which is a set of measure zero). Further-
 4015 more we are interested in the probability density function of y . A more
 4016 precise notation would have to say $\mathbb{E}_y[f(y) | \sigma(x)]$, where we take the
 4017 expectation over y of a test function f conditioned on the σ -algebra of
 4018 x . A more technical audience interested in the details of probability the-
 4019 ory have many options (Jacod and Protter, 2004; Jaynes, 2003; MacKay,
 4020 2003b; Grimmett and Welsh, 2014) including some very technical dis-
 4021 cussions (Çinlar, 2011; Dudley, 2002; Shirayev, 1984; Lehmann and
 4022 Casella, 1998; Bickel and Doksum, 2006). As machine learning allows
 4023 us to model more intricate distributions on ever more complex types of
 4024 data, a developer of probabilistic machine learning models would have to
 4025 understand these more technical aspects. Machine learning books with a
 4026 probabilistic modelling focus includes MacKay (2003b); Bishop (2006);
 4027 Murphy (2012); Barber (2012); Rasmussen and Williams (2006).

4028

Exercises

- 6.1 Consider a mixture of two Gaussian distributions (illustrated in Figure 6.4)

$$\mathcal{N}\left(\begin{bmatrix} 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) + 1.5\mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & 2.9 \\ -1.1 & 0.7 \end{bmatrix}\right).$$

4029

- 1 Compute the marginal distributions for each dimension
- 2 Compute the mean, mode and median for each marginal distribution
- 3 Compute the mean and mode for the 2 dimensional distribution

4030

- 6.2 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 :

4031

$$p(x|\mu) = \mu^x(1-\mu)^{1-x}, \quad x \in \{0, 1\}$$

4032

Choose a conjugate prior for the Bernoulli likelihood and compute the posterior distribution $p(\mu|x_1, \dots, x_N)$.

4033

- 6.3 Consider the following time-series model:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{A}\mathbf{x}_t + \mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \\ \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \end{aligned}$$

4034

where \mathbf{w}, \mathbf{v} are i.i.d. Gaussian noise variables. Further, assume that $p(\mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

4035

- 1 What is the form of $p(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T)$? Justify your answer (you do not have to explicitly compute the joint distribution). (1–2 sentences)
- 2 Assume that $p(\mathbf{x}_t|\mathbf{y}_1, \dots, \mathbf{y}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$.

4036

a) Compute $p(\mathbf{x}_{t+1}|\mathbf{y}_1, \dots, \mathbf{y}_t)$

4037

b) Compute $p(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}|\mathbf{y}_1, \dots, \mathbf{y}_t)$

4038

c) At time $t+1$, we observe the value $\mathbf{y}_{t+1} = \hat{\mathbf{y}}$. Compute $p(\mathbf{x}_{t+1}|\mathbf{y}_1, \dots, \mathbf{y}_{t+1})$.

4039

- 6.4 Prove the relationship in Equation 6.44, which relates the standard definition of the variance to the raw score expression for the variance.

4040

- 6.5 Prove the relationship in Equation 6.45, which relates the pairwise difference between examples in a dataset with the raw score expression for the variance.

4041

- 6.6 Express the Bernoulli distribution in the natural parameter form of the exponential family (Equation (6.110)).

4042

- 6.7 Express the Binomial distribution as an exponential family distribution. Also express the Beta distribution as an exponential family distribution. Show that the product of the Beta and the Binomial distribution is also a member of the exponential family.

4043

- 6.8 Derive the relationship in Section 6.5.2 in two ways:

4044

1 By completing the square

4045

2 By expressing the Gaussian in its exponential family form

The *product* of two Gaussians $\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})\mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{B})$ is an unnormalized Gaussian distribution $c\mathcal{N}(\mathbf{x}|\mathbf{c}, \mathbf{C})$ with

$$\mathbf{C} = (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1} \tag{6.158}$$

$$\mathbf{c} = \mathbf{C}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}) \quad (6.159)$$

$$c = (2\pi)^{-\frac{D}{2}} |\mathbf{A} + \mathbf{B}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{a} - \mathbf{b})^\top (\mathbf{A} + \mathbf{B})^{-1}(\mathbf{a} - \mathbf{b})\right). \quad (6.160)$$

Note that the normalizing constant c itself can be considered a (normalized) Gaussian distribution either in \mathbf{a} or in \mathbf{b} with an “inflated” covariance matrix $\mathbf{A} + \mathbf{B}$, i.e., $c = \mathcal{N}(\mathbf{a} | \mathbf{b}, \mathbf{A} + \mathbf{B}) = \mathcal{N}(\mathbf{b} | \mathbf{a}, \mathbf{A} + \mathbf{B})$.

6.9 Iterated Expectations.

Consider two random variables x, y with joint distribution $p(x, y)$. Show that:

$$\mathbb{E}_x[x] = \mathbb{E}_y[\mathbb{E}_x[x|y]]$$

Here, $\mathbb{E}_x[x|y]$ denotes the expected value of x under the conditional distribution $p(x|y)$.

6.10 Manipulation of Gaussian Random Variables.

Consider a Gaussian random variable $\mathbf{x} \sim \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$, where $\mathbf{x} \in \mathbb{R}^D$. Furthermore, we have

$$\mathbf{y} = \mathbf{Ax} + \mathbf{b} + \mathbf{w}, \quad (6.161)$$

where $\mathbf{y} \in \mathbb{R}^E$, $\mathbf{A} \in \mathbb{R}^{E \times D}$, $\mathbf{b} \in \mathbb{R}^E$, and $\mathbf{w} \sim \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{Q})$ is independent Gaussian noise. “Independent” implies that \mathbf{x} and \mathbf{w} are independent random variables and that \mathbf{Q} is diagonal.

1 Write down the likelihood $p(\mathbf{y}|\mathbf{x})$.

2 The distribution $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}$ is Gaussian.¹ Compute the mean $\boldsymbol{\mu}_y$ and the covariance $\boldsymbol{\Sigma}_y$. Derive your result in detail.

3 The random variable \mathbf{y} is being transformed according to the measurement mapping

$$\mathbf{z} = \mathbf{Cy} + \mathbf{v}, \quad (6.162)$$

where $\mathbf{z} \in \mathbb{R}^F$, $\mathbf{C} \in \mathbb{R}^{F \times E}$, and $\mathbf{v} \sim \mathcal{N}(\mathbf{v} | \mathbf{0}, \mathbf{R})$ is independent Gaussian (measurement) noise.

- Write down $p(\mathbf{z}|\mathbf{y})$.

- Compute $p(\mathbf{z})$, i.e., the mean $\boldsymbol{\mu}_z$ and the covariance $\boldsymbol{\Sigma}_z$. Derive your result in detail.

4 Now, a value $\hat{\mathbf{y}}$ is measured. Compute the posterior distribution $p(\mathbf{x}|\hat{\mathbf{y}})$.²

Hint for solution: Start by explicitly computing the joint Gaussian $p(\mathbf{x}, \mathbf{y})$.

This also requires to compute the cross-covariances $\text{Cov}_{\mathbf{x}, \mathbf{y}}[\mathbf{x}, \mathbf{y}]$ and $\text{Cov}_{\mathbf{y}, \mathbf{x}}[\mathbf{y}, \mathbf{x}]$.

Then, apply the rules for Gaussian conditioning.

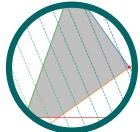
6.11 Probability integral transformation

Given a continuous random variable x , with cdf $F_x(x)$. Show that the random variable $y = F_x(x)$ is uniformly distributed.

¹An affine transformation of the Gaussian random variable \mathbf{x} into $\mathbf{Ax} + \mathbf{b}$ preserves Gaussianity. Furthermore, the sum of this Gaussian random variable and the independent Gaussian random variable \mathbf{w} is Gaussian.

²This posterior is also Gaussian, i.e., we need to determine only its mean and covariance matrix.

Continuous Optimization



4064 Since machine learning algorithms are implemented on a computer, the
 4065 mathematical formulations are expressed as numerical optimization meth-
 4066 ods. This chapter describes the basic numerical methods for training ma-
 4067 chine learning models. Training a machine learning model often boils
 4068 down to finding a good set of parameters. The notion of “good” is de-
 4069 termined by the objective function or the probabilistic model, which we
 4070 will see examples of in the second part of this book. Given an objective
 4071 function finding the best value is done using optimization algorithms.

Since we consider
 4071 data and models in
 4072 \mathbb{R}^D the
 4073 optimization
 4074 problems we face
 4075 are continuous
 4076 optimization
 4077 problems, as
 4078 opposed to
 4079 combinatorial
 4080 optimization
 4081 problems for
 4082 discrete variables.
 4083
 4084
 4085
 4086

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.

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

global minimum

local minimum

Stationary points
 are the real roots of
 the derivative, that
 is points that have
 zero gradient.

$$\ell(x) = x^4 + 7x^3 + 5x^2 - 17x + 3. \quad (7.1)$$

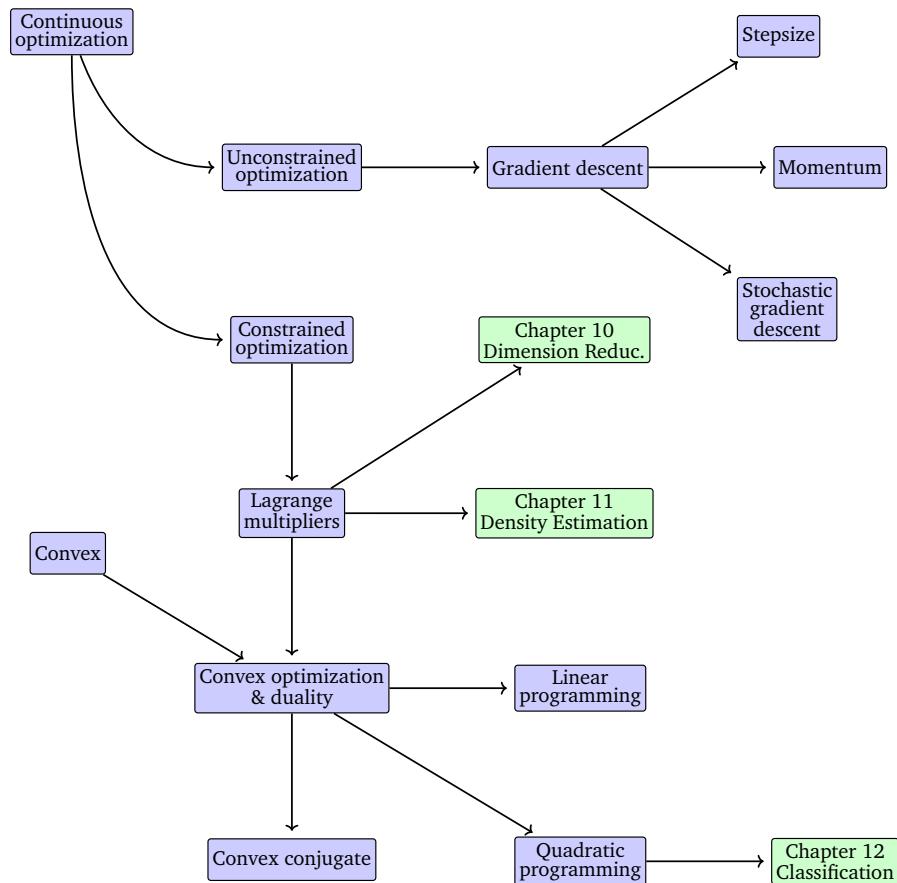


Figure 7.1 A mind map of the concepts related to optimization, as presented in this chapter. There are two main ideas: gradient descent and convex optimization.

Its gradient is given by

$$\frac{d\ell(x)}{dx} = 4x^3 + 21x^2 + 10x - 17. \quad (7.2)$$

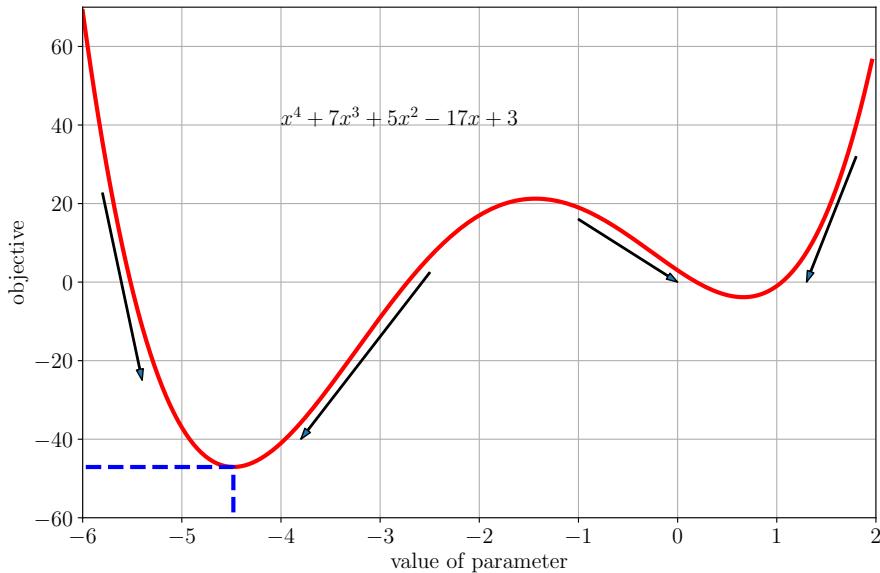
Since this is a cubic equation, it has in general three solutions when set to zero. In the example, two of them are minima and one is a maximum (around $x = -1.4$). Recall that to check whether a stationary point is a minimum or maximum we need to take the derivative a second time and check whether the second derivative is positive or negative at the stationary point.

$$\frac{d^2\ell(x)}{dx^2} = 12x^2 + 42x + 10 \quad (7.3)$$

⁴⁰⁸⁷ By substituting our visually estimated values of $x = -4.5, -1.4, 0.7$ we
⁴⁰⁸⁸ will observe that as expected the middle point is a maximum ($\frac{d^2\ell(x)}{dx^2} < 0$)
⁴⁰⁸⁹ and the other two stationary points are minima.

⁴⁰⁹⁰ Note that we have avoided analytically solving for values of x in the
⁴⁰⁹¹ previous discussion, although for low order polynomials such as the above

Figure 7.2 Example objective function. Gradients are indicated by arrows, and the global minimum is indicated by the dashed blue line.



According to the Abel-Ruffini theorem, there is in general no algebraic solution for polynomials of degree 5 or more. we could do so. In general, we are unable to find analytic solutions, and hence we 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.

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

$$\min_x f(\mathbf{x}) \quad (7.4)$$

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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 Section 5.1 that the gradient points in the direction of the steepest ascent. Another useful intuition is to consider the set of lines where the function is at a certain value ($f(\mathbf{x}) = c$ for some value $c \in \mathbb{R}$), which are known as the contour lines. The gradient points in a direction that 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 $f(\mathbf{x})$) with a ball starting at a particular location \mathbf{x}_0 . When the ball is released, it will move downhill in the direction of steepest descent. Gradient descent exploits the fact that $f(\mathbf{x}_0)$ decreases fastest if one moves from \mathbf{x}_0 in the direction of the negative gradient $-((\nabla f)(\mathbf{x}_0))^\top$ of f at \mathbf{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

$$\mathbf{x}_1 = \mathbf{x}_0 - \gamma ((\nabla f)(\mathbf{x}_0))^\top \quad (7.5)$$

for a small step size $\gamma \geq 0$ then $f(\mathbf{x}_1) \leq f(\mathbf{x}_0)$. 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 $f(\mathbf{x}_*)$ of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$, we start with an initial guess \mathbf{x}_0 of the parameters we wish to optimize and then iterate according to

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i ((\nabla f)(\mathbf{x}_i))^\top. \quad (7.6)$$

For suitable step size γ_i , the sequence $f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq \dots$ converges to a local minimum.

Example 7.1

Consider a quadratic function in two dimensions

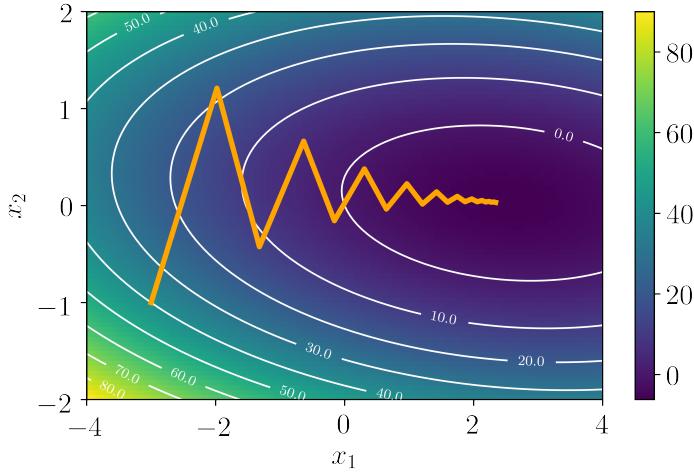
$$f \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (7.7)$$

with gradient

$$\nabla f \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}. \quad (7.8)$$

Starting at the initial location $\mathbf{x}_0 = [-3, -1]^\top$, 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

Figure 7.3 Gradient descent on a 2 dimensional quadratic surface (shown as a heatmap). See Example 7.1 for a description.



plugging x_0 into (7.8)) that the gradient at x_0 points north and east, leading to $x_1 = [-1.98, 1.21]^\top$. Repeating that argument gives us $x_2 = [-1.32, -0.42]^\top$, and so on.

4128 **Remark.** Gradient descent can be relatively slow close to the minimum:
4129 Its asymptotic rate of convergence is inferior to many other methods. Us-
4130 ing the ball rolling down the hill analogy, when the surface is a long thin
4131 valley the problem is poorly conditioned (Trefethen and Bau III, 1997).
4132 For poorly conditioned convex problems, gradient descent increasingly
4133 ‘zigzags’ as the gradients point nearly orthogonally to the shortest direc-
4134 tion to a minimum point, see Fig. 7.3. ◇

7.1.1 Stepsize

4135 As mentioned earlier, choosing a good stepsize is important in gradient
4136 descent. If the stepsize is too small, gradient descent can be slow. If the
4137 stepsize is chosen too large, gradient descent can overshoot, fail to con-
4138 verge, or even diverge. We will discuss the use of momentum in the next
4139 section. It is a method that smoothes out erratic behavior of gradient up-
4140 dates and dampens oscillations.

4141 Adaptive gradient methods rescale the stepsize at each iteration, de-
4142 pending on local properties of the function. There are two simple heuris-
4143 tics (Toussaint, 2012):

- 4145 • When the function value increases after a gradient step, the step size
4146 was too large. Undo the step and decrease the stepsize.
- 4147 • When the function value decreases the step could have been larger. Try
4148 to increase the stepsize.

4149 Although the “undo” step seems to be a waste of resources, using this
 4150 heuristic guarantees monotonic convergence.

Example 7.2 (Solving a Linear Equation System)

When we solve linear equations of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, in practice we solve $\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$ approximately by finding \mathbf{x}_* that minimizes the squared error

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = (\mathbf{A}\mathbf{x} - \mathbf{b})^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) \quad (7.9)$$

if we use the Euclidean norm. The gradient of (7.9) with respect to \mathbf{x} is

$$\nabla_{\mathbf{x}} = 2(\mathbf{A}\mathbf{x} - \mathbf{b})^\top \mathbf{A}. \quad (7.10)$$

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 $\mathbf{A}\mathbf{x} = \mathbf{b}$. We will see more on solving squared error problems in Chapter 9.

4151 *Remark.* When applied to the solution of linear systems of equations $\mathbf{A}\mathbf{x} =$
 4152 \mathbf{b} gradient descent may converge slowly. The speed of convergence of gra-
 4153 dient descent is dependent on the *condition number* $\kappa = \frac{\sigma(\mathbf{A})_{\max}}{\sigma(\mathbf{A})_{\min}}$, which
 4154 is the ratio of the maximum to the minimum singular value (Section 4.5)
 4155 of \mathbf{A} . The condition number essentially measures the ratio of the most
 4156 curved direction versus the least curved direction, which corresponds to
 4157 our imagery that poorly conditioned problems are long thin valleys: they
 4158 are very curved in one direction, but very flat in the other. Instead of di-
 4159 rectly solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, one could instead solve $\mathbf{P}^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{0}$, where
 4160 \mathbf{P} is called the *preconditioner*. The goal is to design \mathbf{P}^{-1} such that $\mathbf{P}^{-1}\mathbf{A}$
 4161 has a better condition number, but at the same time \mathbf{P}^{-1} is easy to com-
 4162 pute. For further information on gradient descent, pre-conditioning and
 4163 convergence we refer to (Boyd and Vandenberghe, 2004, Chapter 9). ◇

condition number

preconditioner

7.1.2 Gradient Descent with Momentum

4164 As illustrated in Figure 7.3, the convergence of gradient descent may be
 4165 very slow if the curvature of the optimization surface is such that the there
 4166 are regions which are poorly scaled. The curvature is such that the gra-
 4167 dient descent steps hops between the walls of the valley, and approaches
 4168 the optimum in small steps. The proposed tweak to improve convergence
 4169 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

Goh (2017) wrote an intuitive blog post on gradient descent with momentum.

emulates the phenomenon of a heavy ball which is reluctant to change directions. The idea is to have a gradient update with memory to implement a moving average. The momentum-based method remembers the update $\Delta \mathbf{x}_i$ at each iteration i and determines the next update as a linear combination of the current and previous gradients

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i ((\nabla f)(\mathbf{x}_i))^{\top} + \alpha \Delta \mathbf{x}_i \quad (7.11)$$

$$\Delta \mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1} = -\gamma_{i-1} ((\nabla f)(\mathbf{x}_{i-1}))^{\top}, \quad (7.12)$$

4171 where $\alpha \in [0, 1]$. Sometimes we will only know the gradient approxi-
4172 mately. In such cases the momentum term is useful since it averages out
4173 different noisy estimates of the gradient. One particularly useful way to
4174 obtain an approximate gradient is using a stochastic approximation, which
4175 we discuss next.

4176 7.1.3 Stochastic Gradient Descent

4177 Computing the gradient can be very time consuming. However, often it is
4178 possible to find a “cheap” approximation of the gradient. Approximating
4179 the gradient is still useful as long as it points in roughly the same direction
4180 as the true gradient.

4181 *Stochastic gradient descent* (often shortened to SGD) is a stochastic ap-
4182 proximation of the gradient descent method for minimizing an objective
4183 function that is written as a sum of differentiable functions. The word
4184 stochastic here refers to the fact that we acknowledge that we do not
4185 know the gradient precisely, but instead only know a noisy approxima-
4186 tion to it. By constraining the probability distribution of the approximate
4187 gradients, we can still theoretically guarantee that SGD will converge.

In machine learning given $n = 1, \dots, N$ 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

$$L(\boldsymbol{\theta}) = \sum_{n=1}^N L_n(\boldsymbol{\theta}) \quad (7.13)$$

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,

$$L(\boldsymbol{\theta}) = - \sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) \quad (7.14)$$

4188 where $\mathbf{x}_n \in \mathbb{R}^D$ are the training inputs, y_n are the training targets and $\boldsymbol{\theta}$
4189 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

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma_i (\nabla L(\boldsymbol{\theta}_i))^\top = \boldsymbol{\theta}_i - \gamma_i \sum_{n=1}^N (\nabla L_n(\boldsymbol{\theta}_i))^\top \quad (7.15)$$

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.

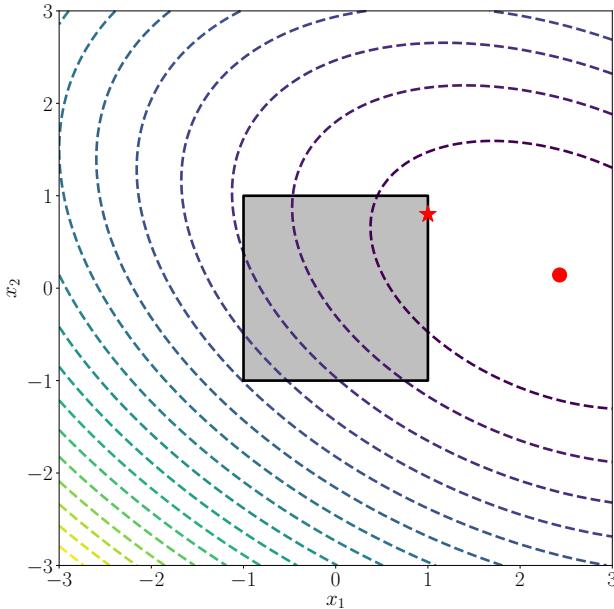
Consider the term $\sum_{n=1}^N (\nabla L_n(\boldsymbol{\theta}_i))$ in (7.15) above: we can reduce the amount of computation by taking a sum over a smaller set of L_n . In contrast to batch gradient descent, which uses all L_n for $n = 1, \dots, N$, we randomly choose a subset of L_n for mini-batch 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.

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). \diamond

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 (Section 6.4.1). Large mini-batch sizes will provide accurate estimates of the gradient, reducing 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. The reduction in variance leads to more stable convergence, but each gradient calculation will be more expensive.

In contrast small mini-batches are quick to estimate. 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. In machine learning, optimization methods are used for training by minimizing an objective function on the training data, but the overall goal is to improve generalization performance (Chapter 8). Since the goal in machine learning does not necessarily need a precise estimate of the minimum of the objective function, approximate gradients using mini-batch

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 ($-1 \leq x \leq 1$ and $-1 \leq y \leq 1$) require that the optimal solution are within the box, resulting in an optimal value indicated by the star.



approaches have been widely used. 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).

4234

7.2 Constrained Optimization and Lagrange Multipliers

In the previous section, we considered the problem of solving for the minimum of a function

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (7.16)$$

4235

where $f : \mathbb{R}^D \rightarrow \mathbb{R}$.

In this section we have additional constraints. That is for real valued functions $g_i : \mathbb{R}^D \rightarrow \mathbb{R}$ for $i = 1, \dots, m$ we consider the constrained optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (7.17)$$

$$\text{subject to } g_i(\mathbf{x}) \leq 0 \quad \text{for all } i = 1, \dots, m$$

4236

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

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})) \quad (7.18)$$

where $\mathbf{1}(z)$ is an infinite step function

$$\mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}. \quad (7.19)$$

4238 This gives infinite penalty if the constraint is not satisfied, and hence
 4239 would provide the same solution. However, this infinite step function is
 4240 equally difficult to optimize. We can overcome this difficulty by introducing
 4241 *Lagrange multipliers*. The idea of Lagrange multipliers is to replace the
 4242 step function with a linear function.

We associate to problem (7.17) the *Lagrangian* by introducing the Lagrange multipliers $\lambda_i \geq 0$ corresponding to each inequality constraint respectively (Boyd and Vandenberghe, 2004, Chapter 4).

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}) \end{aligned} \quad (7.20)$$

4243 where in the last line we have a concatenated all constraints $g_i(\mathbf{x})$ into a
 4244 vector $\mathbf{g}(\mathbf{x})$, and all the Lagrange multipliers into a vector $\boldsymbol{\lambda} \in \mathbb{R}^m$.

4245 We now introduce the idea of Lagrangian duality. In general, duality
 4246 in optimization is the idea of converting an optimization problem in one
 4247 set of variables \mathbf{x} (called the primal variables), into another optimization
 4248 problem in a different set of variables $\boldsymbol{\lambda}$ (called the dual variables). We
 4249 introduce two different approaches to duality: in this section we discuss
 4250 Lagrangian duality, and in Section 7.3.3 we discuss Legendre-Fenchel du-
 4251 ality.

Definition 7.1. The problem in (7.17)

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to $g_i(\mathbf{x}) \leq 0$ for all $i = 1, \dots, m$

is known as the *primal problem*, corresponding to the primal variables x .
 The associated *Lagrangian dual problem* is given by

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \mathfrak{D}(\boldsymbol{\lambda}) \quad (7.21)$$

subject to $\boldsymbol{\lambda} \geq 0$,

4252 where $\boldsymbol{\lambda}$ are the dual variables and $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$.

4253 *Remark.* In the discussion of Definition 7.1 below, we use two concepts
 4254 which are also of independent interest (Boyd and Vandenberghe, 2004).

Lagrange multipliers

Lagrangian

primal problem
Lagrangian dual
problem

First the *minimax inequality*, which says that for any function with two arguments $\varphi(\mathbf{x}, \mathbf{y})$, the maximin is less than the minimax, that is

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \varphi(\mathbf{x}, \mathbf{y}) \leq \min_{\mathbf{x}} \max_{\mathbf{y}} \varphi(\mathbf{x}, \mathbf{y}). \quad (7.23)$$

This inequality can be proved by considering the inequality

$$\text{For all } \mathbf{x}, \mathbf{y} \quad \min_{\mathbf{x}} \varphi(\mathbf{x}, \mathbf{y}) \leq \max_{\mathbf{y}} \varphi(\mathbf{x}, \mathbf{y}). \quad (7.24)$$

Observe that taking the maximum over \mathbf{y} of the left hand side of (7.24) maintains the inequality since the inequality is true for all \mathbf{y} . Similarly we can take the minimum over \mathbf{x} of the right hand side of (7.24) to obtain (7.23).

The second concept is *weak duality*, which uses (7.23) to show that primal values are always greater than or equal to dual values. This is described in more detail in (7.27). \diamond

Recall that the difference between $J(\mathbf{x})$ in (7.18) and the Lagrangian in (7.20) is that we have relaxed the indicator function to a linear function. Therefore when $\boldsymbol{\lambda} \geq 0$, the Lagrangian $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ is a lower bound of $J(\mathbf{x})$. Hence the maximum of $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is $J(\mathbf{x})$

$$J(\mathbf{x}) = \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}). \quad (7.25)$$

Recall that the original problem was minimising $J(\mathbf{x})$,

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}). \quad (7.26)$$

By the minimax inequality (7.23) it turns out that, for any function swapping the order of the minimum and maximum above results in a smaller value.

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq 0} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \quad (7.27)$$

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 definition follows.

In contrast to the original optimization problem which has constraints, $\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ is an unconstrained optimization problem for a given value of $\boldsymbol{\lambda}$. If solving $\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ 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 \mathbf{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.

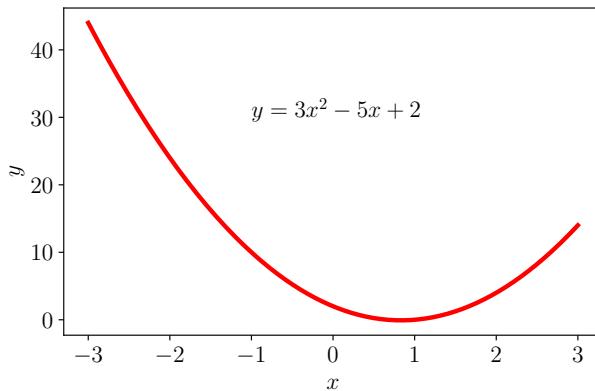


Figure 7.5 Example of a convex function.

Remark (Equality constraints). Consider (7.17) with additional equality constraints

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (7.28)$$

subject to $g_i(\mathbf{x}) \leq 0$ for all $i = 1, \dots, m$
 $h_j(\mathbf{x}) = 0$ for all $j = 1, \dots, n$

We can model equality constraints by replacing them with two inequality constraints. That is for each equality constraint $h_j(\mathbf{x}) = 0$ we equivalently replace it by two constraints $h_j(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) \geq 0$. It turns out that the resulting Lagrange multipliers are then unconstrained.

Therefore we constrain the Lagrange multipliers corresponding to the inequality constraints in (7.28) 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

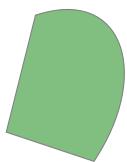
Definition 7.2. A set C is a *convex set* if for any $x, y \in C$ and for any scalar θ with $0 \leq \theta \leq 1$, we have

$$\theta x + (1 - \theta)y \in C \quad (7.29)$$

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

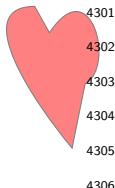


Convex functions are functions such that a straight line between any two points of the function lie above the function. Figure 7.2 shows a non-convex function and Figure 7.3 shows a convex function. Another convex function is shown in Figure 7.5.

Definition 7.3. Let function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ be a function whose domain is a convex set. The function f is a *convex function* if for all \mathbf{x}, \mathbf{y} in the domain of f , and for any scalar θ with $0 \leq \theta \leq 1$, we have

$$f(\theta\mathbf{x} + (1 - \theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1 - \theta)f(\mathbf{y}). \quad (7.30)$$

Figure 7.7 Example of a nonconvex set



convex function
concave function

Remark. A *concave function* is the negative of a convex function. \diamond

The constraints involving $g(\cdot)$ and $h(\cdot)$ in (7.28) 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.

If a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable, we can specify convexity in terms of its gradient $\nabla_{\mathbf{x}}f(\mathbf{x})$ (Section 5.2). A function $f(\mathbf{x})$ is convex if and only if for any two points \mathbf{x}, \mathbf{y} ,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla_{\mathbf{x}}f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}). \quad (7.31)$$

If we further know that a function $f(\mathbf{x})$ is twice differentiable, that is the Hessian (5.146) exists for all values in the domain of \mathbf{x} , then the function $f(\mathbf{x})$ is convex if and only if $\nabla_{\mathbf{x}}^2f(\mathbf{x})$ is positive semi-definite (Boyd and Vandenberghe, 2004).

Example 7.3

The negative entropy $f(x) = x \log_2 x$ is convex for $x > 0$. A visualization of the function is shown 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 7.3. Consider a point midway between the two points (that is $\theta = 0.5$), then the left hand side is $f(0.5 \cdot 2 + 0.5 \cdot 4) = 3 \log_2 3 \approx 4.75$. 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 (7.31). Calculating the derivative of $f(x)$, we obtain

$$\nabla_x(x \log_2 x) = 1 \cdot \log_2 x + x \cdot \frac{1}{x \log_e 2} \quad (7.32)$$

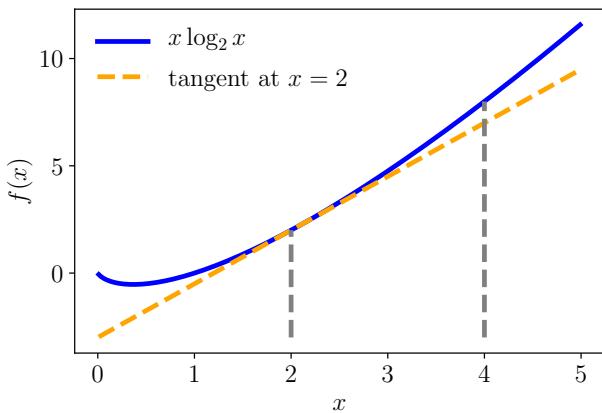


Figure 7.8 The negative entropy function (which is convex), and its tangent at $x = 2$.

$$= \log_2 x + \frac{1}{\log_e 2}. \quad (7.33)$$

Using the same two test points $x = 2$ and $x = 4$, the left hand side of (7.31) is given by $f(4) = 8$. The right hand side is

$$f(\mathbf{x}) + \nabla_{\mathbf{x}}^{\top} (\mathbf{y} - \mathbf{x}) = f(2) + \nabla f(2) \cdot (4 - 2) \quad (7.34)$$

$$= 2 + \left(1 + \frac{1}{\log_e 2}\right) \cdot 2 \approx 6.9. \quad (7.35)$$

4311 We can check that a function or set is convex from first principles by
4312 recalling the definitions. In practice we often rely on operations that pre-
4313 serve convexity to check that a particular function or set is convex. Al-
4314 though the details are vastly different, this is again the idea of closure
4315 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 $\alpha \geq 0$ is a nonnegative scalar, then the function αf is convex. We can see this by multiplying α to both sides of equation in Definition 7.3, 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

$$f_1(\theta \mathbf{x} + (1 - \theta) \mathbf{y}) \leq \theta f_1(\mathbf{x}) + (1 - \theta) f_1(\mathbf{y}) \quad (7.36)$$

$$f_2(\theta \mathbf{x} + (1 - \theta) \mathbf{y}) \leq \theta f_2(\mathbf{x}) + (1 - \theta) f_2(\mathbf{y}). \quad (7.37)$$

Summing up both sides gives us

$$f_1(\theta \mathbf{x} + (1 - \theta) \mathbf{y}) + f_2(\theta \mathbf{x} + (1 - \theta) \mathbf{y})$$

$$\leq \theta f_1(\mathbf{x}) + (1 - \theta)f_1(\mathbf{y}) + \theta f_2(\mathbf{x}) + (1 - \theta)f_2(\mathbf{y}) \quad (7.38)$$

where the right hand side can be rearranged to

$$\theta(f_1(\mathbf{x}) + f_2(\mathbf{x})) + (1 - \theta)(f_1(\mathbf{y}) + f_2(\mathbf{y})) \quad (7.39)$$

completing the proof that the sum of convex functions is convex.

Combining the two facts above, we see that $\alpha f_1(\mathbf{x}) + \beta f_2(\mathbf{x})$ is convex for $\alpha, \beta \geq 0$. This closure property can be extended using a similar argument for nonnegative weighted sums of more than two convex functions.

4316 *Remark.* The inequality defining convex functions, see 7.30, is sometimes
Jensen's inequality
4317 called *Jensen's inequality*. In fact a whole class of inequalities for taking
4318 nonnegative weighted sums of convex functions are all called Jensen's
4319 inequality. \diamond

convex optimization
problem In summary, a constrained optimization problem is called a *convex optimization problem* if

$$\begin{aligned} & \min_{\mathbf{x}} f(\mathbf{x}) \\ & \text{subject to } g_i(\mathbf{x}) \leq 0 \quad \text{for all } i = 1, \dots, m \\ & \quad h_j(\mathbf{x}) = 0 \quad \text{for all } j = 1, \dots, n \end{aligned} \quad (7.40)$$

4320 where all the functions $f(\mathbf{x})$ and $g_i(\mathbf{x})$ are convex functions, and all
4321 $h_j(\mathbf{x}) = 0$ are convex sets. In the following two subsections, we will de-
4322 scribe two classes of convex optimization problems that are widely used
4323 and well understood.

4324 7.3.1 Linear Programming

Consider the special case when all the functions above are linear, that is

$$\begin{aligned} & \min_{\mathbf{x} \in \mathbb{R}^d} \mathbf{c}^\top \mathbf{x} \\ & \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned} \quad (7.41)$$

Linear programs are
one of the most
widely used
approaches in
industry. where $\mathbf{A} \in \mathbb{R}^{m \times d}$ and $\mathbf{b} \in \mathbb{R}^m$. 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.

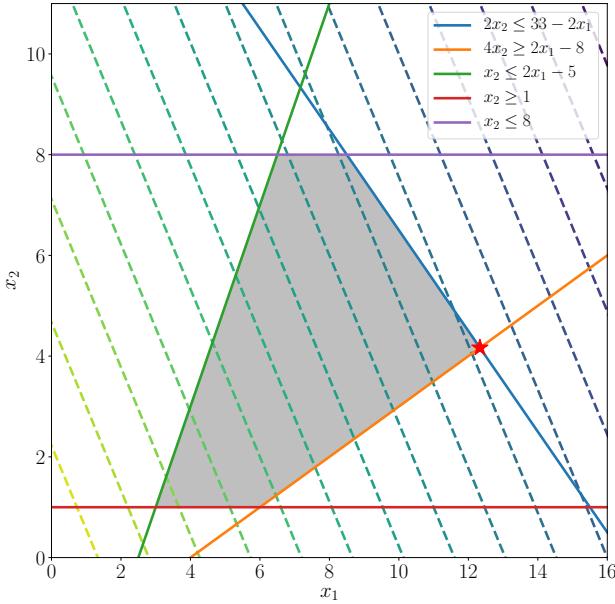


Figure 7.9
 Illustration of a linear program. The unconstrained problem (indicated by the contour lines) has a minimum on the right side. The optimal value given the constraints are shown by the star.

$$\min_{\mathbf{x} \in \mathbb{R}^2} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (7.42)$$

$$\text{subject to } \begin{bmatrix} 2 & 2 \\ 2 & -4 \\ -2 & 1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 33 \\ 8 \\ 5 \\ -1 \\ 8 \end{bmatrix} \quad (7.43)$$

The Lagrangian is given by

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

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 \mathbf{x} .

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

Taking the derivative of $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ with respect to \mathbf{x} and setting it to zero gives us

$$\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda} = \mathbf{0}.$$

Therefore the dual Lagrangian is $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^\top \mathbf{b}$. Recall we would like to maximize $\mathfrak{D}(\boldsymbol{\lambda})$. In addition to the constraint due to the derivative of

It is convention to minimize the primal and maximize the dual.

$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ being zero, we also have the fact that $\boldsymbol{\lambda} \geq \mathbf{0}$, resulting in the following dual optimization problem

$$\begin{aligned} & \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} -\mathbf{b}^\top \boldsymbol{\lambda} \\ & \text{subject to } \mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda} = \mathbf{0} \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{7.44}$$

This is also a linear program, but with m variables. We have the choice of solving the primal (7.41) or the dual (7.44) 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.

4331

7.3.2 Quadratic Programming

Consider when the objective function is a convex quadratic function, and the constraints are affine,

$$\begin{aligned} & \min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} + \mathbf{c}^\top \mathbf{x} \\ & \text{subject to } \mathbf{A} \mathbf{x} \leq \mathbf{b} \end{aligned} \tag{7.45}$$

4332

where $\mathbf{A} \in \mathbb{R}^{m \times d}$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^d$. The square symmetric matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$ 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 \mathbf{Q} , resulting in elliptical contour lines. The optimal value must lie in the shaded (feasible) region, and is indicated by the star.

$$\min_{\mathbf{x} \in \mathbb{R}^2} \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{7.46}$$

$$\text{subject to } \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{7.47}$$

The Lagrangian is given by

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} + \mathbf{c}^\top \mathbf{x} + \boldsymbol{\lambda}^\top (\mathbf{A} \mathbf{x} - \mathbf{b}) \\ &= \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} + (\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda})^\top \mathbf{x} - \boldsymbol{\lambda}^\top \mathbf{b} \end{aligned}$$

where again we have rearranged the terms. Taking the derivative of $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ with respect to \mathbf{x} and setting it to zero gives

$$\mathbf{Q}\mathbf{x} + (\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda}) = 0$$

Assuming that \mathbf{Q} is invertible, we get

$$\mathbf{x} = -\mathbf{Q}^{-1}(\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda}) \quad (7.48)$$

Substituting (7.48) into the primal Lagrangian $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ we get the dual Lagrangian

$$\mathcal{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda})\mathbf{Q}^{-1}(\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \mathbf{b}$$

Therefore the dual optimization problem is given by

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} -\frac{1}{2}(\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda})\mathbf{Q}^{-1}(\mathbf{c} + \mathbf{A}^\top \boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \mathbf{b} \quad (7.49)$$

$$\text{subject to } \boldsymbol{\lambda} \geq 0. \quad (7.50)$$

4336 We will see an application of Quadratic Programming in machine learning
4337 in Chapter 12.

7.3.3 Legendre-Fenchel Transform and Convex Conjugate

4339 Let us revisit the idea of duality, which we saw in Section 7.2, without
4340 considering constraints. One useful fact about convex sets is that a convex
4341 set can be equivalently described by its supporting hyperplanes. A hyper-
4342 plane is called a *supporting hyperplane* of a convex set if it intersects the
4343 convex set and the convex set is contained on just one side of it. Recall
4344 that for a convex function, we can fill it up to obtain the epigraph which
4345 is a convex set. Therefore we can also describe convex functions in terms
4346 of their supporting hyperplanes. Furthermore observe that the supporting
4347 hyperplane just touches the convex function, and is in fact the tangent to
4348 the function at that point. And recall that the tangent of a function $f(\mathbf{x})$ at
4349 a given point \mathbf{x}_0 is the evaluation of the gradient of that function at that
4350 point $\left. \frac{df(\mathbf{x})}{d\mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_0}$. In summary, because convex sets can be equivalently
4351 described by its supporting hyperplanes, convex functions can be equiv-
4352 alently described by a function of their gradient. The *Legendre transform*
4353 formalizes this concept.

supporting
hyperplane

4354 We begin with the most general definition which unfortunately has a
4355 counterintuitive form, and look at special cases to try to relate the defini-
4356 tion to the intuition above. The *Legendre-Fenchel transform* is a transfor-
4357 mation (in the sense of a Fourier transform) from a convex differentiable
4358 function $f(\mathbf{x})$ to a function that depends on the tangents $s(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x})$.
4359 It is worth stressing that this is a transformation of the function $f(\cdot)$ and
4360 not the variable \mathbf{x} or the function evaluated at a value. The Legendre-
4361 Fenchel transform is also known as the convex conjugate (for reasons

Legendre transform
 Physics students are often introduced to the Legendre transform as relating the Lagrangian and the Hamiltonian in classical mechanics. Legendre-Fenchel transform

4362 we will see soon) and is closely related to duality (Hiriart-Urruty and
4363 Lemaréchal, 2001, Chapter 5).

convex conjugate

Definition 7.4. The *convex conjugate* of a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is a function f^* defined by

$$f^*(\mathbf{s}) = \sup_{\mathbf{x} \in \mathbb{R}^D} (\langle \mathbf{s}, \mathbf{x} \rangle - f(\mathbf{x})) \quad (7.51)$$

4364 Note that the convex conjugate definition above does not need the func-
4365 tion f to be convex nor differentiable. In the definition above, we have
4366 used a general inner product (Section 3.2) but in the rest of this section
4367 we will consider the standard dot product between finite dimensional vec-
4368 tors ($\langle \mathbf{s}, \mathbf{x} \rangle = \mathbf{s}^\top \mathbf{x}$) to avoid too many technical details.

This derivation is easiest to understand by drawing the reasoning as it progresses.

To understand the above definition in a geometric fashion, consider a 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

$$y - f(x_0) = s(x - x_0) \quad (7.52)$$

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

$$\inf_{x_0} -sx_0 + f(x_0). \quad (7.53)$$

4369 The convex conjugate above is by convention defined to be the negative
4370 of this. The reasoning in this paragraph did not rely on the fact that we
4371 chose a one dimensional convex and differentiable function, and holds for
4372 $f : \mathbb{R}^D \rightarrow \mathbb{R}$ 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

$$f(x_0) = sx_0 + c. \quad (7.54)$$

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

$$-c = sx_0 - f(x_0). \quad (7.55)$$

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 $f^*(s)$.

$$f^*(s) = sx_0 - f(x_0). \quad (7.56)$$

4373 Compare (7.56) with Definition 7.4, and observe that (7.56) is a special
4374 case (without the supremum). \diamond

4375 The conjugate function has nice properties, for example for convex
4376 functions, applying the Legendre transform again gets us back to the origi-
4377 nal function. In the same way that the slope of $f(x)$ is s , the slope of $f^*(s)$
4378 is x . The following two examples show common uses of convex conjugates
4379 in machine learning.

Example 7.7

To illustrate the application of convex conjugates, consider the quadratic function based on a positive definite matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$. We denote the primal variable to be $\mathbf{y} \in \mathbb{R}^n$ and the dual variable to be $\boldsymbol{\alpha} \in \mathbb{R}^n$.

$$f(\mathbf{y}) = \frac{\lambda}{2} \mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y} \quad (7.57)$$

Applying Definition 7.4, we obtain the function

$$f^*(\boldsymbol{\alpha}) = \sup_{\mathbf{y} \in \mathbb{R}^n} \langle \mathbf{y}, \boldsymbol{\alpha} \rangle - \frac{\lambda}{2} \mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}. \quad (7.58)$$

Observe that the function is differentiable, and hence we can find the maximum by taking the derivative and with respect to \mathbf{y} setting it to zero.

$$\frac{\partial [\langle \mathbf{y}, \boldsymbol{\alpha} \rangle - \frac{\lambda}{2} \mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}]}{\partial \mathbf{y}} = (\boldsymbol{\alpha} - \lambda \mathbf{K}^{-1} \mathbf{y})^\top \quad (7.59)$$

and hence when the gradient is zero we have $\mathbf{y} = \frac{1}{\lambda} \mathbf{K} \boldsymbol{\alpha}$. Substituting into (7.58) yields

$$f^*(\boldsymbol{\alpha}) = \frac{1}{\lambda} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} - \frac{\lambda}{2} \left(\frac{1}{\lambda} \mathbf{K} \boldsymbol{\alpha} \right)^\top \mathbf{K}^{-1} \left(\frac{1}{\lambda} \mathbf{K} \boldsymbol{\alpha} \right) \quad (7.60)$$

$$= \frac{1}{2\lambda} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}. \quad (7.61)$$

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 $\ell : \mathbb{R} \rightarrow \mathbb{R}$. This also illustrates the applica-

tion of the convex conjugate to the vector case. Let $\mathcal{L}(\mathbf{t}) = \sum_{i=1}^n \ell_i(t_i)$,

$$\mathcal{L}^*(\mathbf{z}) = \sup_{\mathbf{t} \in \mathbb{R}^n} \langle \mathbf{z}, \mathbf{t} \rangle - \sum_{i=1}^n \ell_i(t_i) \quad (7.62)$$

$$= \sup_{\mathbf{t} \in \mathbb{R}^n} \sum_{i=1}^n z_i t_i - \ell_i(t_i) \quad \text{definition of dot product} \quad (7.63)$$

$$= \sum_{i=1}^n \sup_{\mathbf{t} \in \mathbb{R}^n} z_i t_i - \ell_i(t_i) \quad (7.64)$$

$$= \sum_{i=1}^n \ell_i^*(z_i) \quad \text{definition of conjugate} \quad (7.65)$$

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 $f(\mathbf{y})$ and $g(\mathbf{x})$ be convex functions, and \mathbf{A} a real matrix of appropriate dimensions such that $\mathbf{Ax} = \mathbf{y}$. Then

$$\min_{\mathbf{x}} f(\mathbf{Ax}) + g(\mathbf{x}) = \min_{\mathbf{Ax}=\mathbf{y}} f(\mathbf{y}) + g(\mathbf{x}). \quad (7.66)$$

By introducing the Lagrange multiplier \mathbf{u} for the constraints $\mathbf{Ax} = \mathbf{y}$,

$$\min_{\mathbf{Ax}=\mathbf{y}} f(\mathbf{y}) + g(\mathbf{x}) = \min_{\mathbf{x}, \mathbf{y}} \max_{\mathbf{u}} f(\mathbf{y}) + g(\mathbf{x}) + (\mathbf{Ax} - \mathbf{y})^\top \mathbf{u} \quad (7.67)$$

$$= \max_{\mathbf{u}} \min_{\mathbf{x}, \mathbf{y}} f(\mathbf{y}) + g(\mathbf{x}) + (\mathbf{Ax} - \mathbf{y})^\top \mathbf{u} \quad (7.68)$$

where the last step of swapping max and min is due to the fact that $f(\mathbf{y})$ and $g(\mathbf{x})$ are convex functions. By splitting up the dot product term and collecting \mathbf{x} and \mathbf{y} ,

$$\max_{\mathbf{u}} \min_{\mathbf{x}, \mathbf{y}} f(\mathbf{y}) + g(\mathbf{x}) + (\mathbf{Ax} - \mathbf{y})^\top \mathbf{u} \quad (7.69)$$

$$= \max_{\mathbf{u}} \left[\min_{\mathbf{y}} -\mathbf{y}^\top \mathbf{u} + f(\mathbf{y}) \right] + \left[\min_{\mathbf{x}} (\mathbf{Ax})^\top \mathbf{u} + g(\mathbf{x}) \right] \quad (7.70)$$

$$= \max_{\mathbf{u}} \left[\min_{\mathbf{y}} -\mathbf{y}^\top \mathbf{u} + f(\mathbf{y}) \right] + \left[\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}^\top \mathbf{u} + g(\mathbf{x}) \right] \quad (7.71)$$

Recall the convex conjugate (Definition 7.4) and the fact that dot products are symmetric,

$$\max_{\mathbf{u}} \left[\min_{\mathbf{y}} -\mathbf{y}^\top \mathbf{u} + f(\mathbf{y}) \right] + \left[\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{A}^\top \mathbf{u} + g(\mathbf{x}) \right] \quad (7.72)$$

$$= \max_{\mathbf{u}} -f^*(\mathbf{u}) - g^*(-\mathbf{A}^\top \mathbf{u}). \quad (7.73)$$

For general inner products, \mathbf{A}^\top is replaced by the adjoint \mathbf{A}^* .

Therefore we have shown that

$$\min_{\mathbf{x}} f(\mathbf{A}\mathbf{x}) + g(\mathbf{x}) = \max_{\mathbf{u}} -f^*(\mathbf{u}) - g^*(-\mathbf{A}^\top \mathbf{u}). \quad (7.74)$$

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. The idea of momentum can be generalized to a general class of acceleration methods (Nesterov, 2018). 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

4418 on different approaches for numerically solving continuous optimization
 4419 problems, including algorithms for constrained optimization problems. A
 4420 good starting point to appreciate this literature are Luenberger (1969)
 4421 and Bonnans et al. (2006). A recent survey of continuous optimization is
 4422 Bubeck (2015).

4423 Modern applications of machine learning often mean that the size of
 4424 datasets prohibit the use of batch gradient descent, and hence stochastic
 4425 gradient descent is the current workhorse of large scale machine learning
 4426 methods. Recent surveys of the literature include (Hazan, 2015; Bottou
 4427 et al., 2018).

4428 For duality and convex optimization, the book by Boyd and Vandenberghe
 4429 (Boyd and Vandenberghe, 2004) includes lectures and slides online.
 4430 A more mathematical treatment is provided by Bertsekas (2009), and
 4431 recent book by one of the key researchers in the area of optimization is
 4432 Nesterov (2018). Convex optimization is based upon convex analysis, and
 4433 the reader interested in more foundational results about convex functions
 4434 is referred to Hiriart-Urruty and Lemaréchal (2001); Rockafellar (1970);
 4435 Borwein and Lewis (2006). Legendre-Fenchel transforms are also covered
 4436 in the above books on convex analysis, but more beginner friendly presentations
 4437 are available at Zia et al. (2009); Gonçalves (2014). The role of
 4438 Legendre-Fenchel transforms in the analysis of convex optimization algo-
 4439 rithms is surveyed in Polyak (2016).

4440 Exercises

7.1 Consider the univariate function

$$f(x) = x^3 + 2x^2 + 5x - 3.$$

4441 Find its stationary points and indicate whether they are maximum, mini-
 4442 mum or saddle points.

4443 7.2 Consider the update equation for stochastic gradient descent (Equation (7.15)).

4444 Write down the update when we use a mini-batch size of one.

4445 7.3 Express the following optimization problem as a standard linear program in
 matrix notation

$$\max_{\mathbf{x} \in \mathbb{R}^2, \xi \in \mathbb{R}} \mathbf{p}^\top \mathbf{x} + \xi$$

4446 subject to the constraints that $\xi \geq 0$, $x_0 \leq 0$ and $x_1 \leq 3$.

4447 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

$$L^*(\beta) + \frac{\gamma}{2}\beta^2$$

where γ is a given hyperparameter.

When Models meet Data

4412 In the first part of the book, we introduced the mathematics that form the
 4413 foundations of many machine learning methods. The hope is that a reader
 4414 would be able to learn the rudimentary forms of the language of mathe-
 4415 matics, which we will now use to describe and discuss machine learning.
 4416 The second part of the book introduces four pillars of machine learning:

- 4417 • Regression (Chapter 9)
- 4418 • Dimensionality reduction (Chapter 10)
- 4419 • Density estimation (Chapter 11)
- 4420 • Classification (Chapter 12)

4421 Recall from Table 1.1 that these problems illustrate two supervised and
 4422 two unsupervised learning methods — one discrete and another continu-
 4423 ous. The main aim of this part of the book is to illustrate how the mathe-
 4424 matical concepts introduced in the first part of the book can be used to
 4425 design machine learning algorithms that can be used to solve tasks within
 4426 the remit of the four pillars. We do not intend to introduce advanced ma-
 4427 chine learning concepts, but instead to provide a set of practical methods
 4428 that allow the reader to apply the knowledge they gained from the first
 4429 part of the book. It also provides a gateway to the wider machine learning
 4430 literature for readers already familiar with the mathematics.

4431 It is worth at this point to pause and consider the problem that a ma-
 4432 chine learning algorithm is designed to solve. As discussed in Chapter 1,
 4433 there are three major components of a machine learning system: data,
 4434 models and learning. The main question of machine learning is “what do
 4435 we mean by good models?”. That is, we are interested to find models that
 4436 perform well on future data. The word *model* has many subtleties and we
 4437 will revisit it multiple times in this chapter. It is also not entirely obvious
 4438 how to objectively define the word “good”. One of the guiding principles
 4439 of machine learning is that good models should perform well on unseen
 4440 data. This requires us to define some performance metrics, such as accu-
 4441 racy or distance from ground truth, as well as figuring out ways to do well
 4442 under these performance metrics.

model

4443 This chapter covers a few necessary bits and pieces of mathematical
 4444 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

learning models. By doing so, we briefly outline the current best practices for training a model such that the resulting predictor does 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 (Figure 8.1), where we think of each row of the table as representing a particular instance or example, and each column to be a particular feature. In recent years machine learning has been applied to many types of data that do not obviously come in the tabular numerical format, for example genomic sequences, text and image contents of a webpage, and social media graphs. We do not discuss the important and challenging aspects of identifying good 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).

Even when we have data in tabular format, there are still choices to be made to obtain a numerical representation. 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 considered for units, scaling, and constraints. Without additional information, one should shift and scale all columns of the dataset such that they mean

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

Table 8.2 Example data from a fictitious human resource database (see Table 8.1), converted to a numerical format.

0 and variance 1. For the purposes of this book we assume that a domain expert already converted data appropriately, i.e., each input \mathbf{x}_n is a D -dimensional vector of real numbers, which are called *features*, *attributes* or *covariates*. We consider a dataset to be of the form as illustrated by Figure 8.2.

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 $n = 1, \dots, N$. We assume that we are given a set of numerical data, represented as an array of vectors (Figure 8.2). Each row is a particular individual \mathbf{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 $d = 1, \dots, D$. Recall that data is represented as vectors, which means that each example (each data point) is a D dimensional vector. The orientation of the table originates from the database community, but for some machine learning algorithms (for example in Chapter 10) the it is more convenient to represent examples as column vectors.

For supervised learning problems we have a *label* y_n associated with each example \mathbf{x}_n . The label y_n has various other names including: target, response variable and annotation. A dataset is written as a set of example-label pairs $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n), \dots, (\mathbf{x}_N, y_N)\}$. The table of examples $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is often concatenated, and written as $\mathbf{X} \in \mathbb{R}^{N \times D}$. Figure 8.1 illustrates an example of a one dimensional input x and corresponding labels y .

Representing data as vectors \mathbf{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.

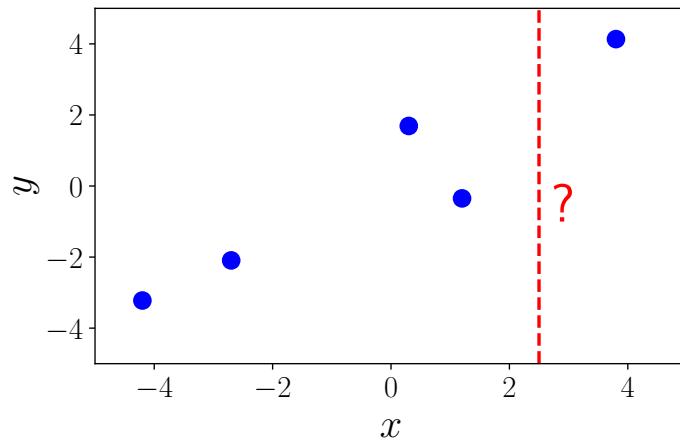
Since we have vector representations of data, we can manipulate data to find potentially better representations of it. We will discuss finding good representations in two ways: finding lower-dimensional approximations of the original feature vector, and using nonlinear higher-dimensional

features
attributes
covariates

example
data point
feature

label

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 $x = 2.5$ (vertical dashed red line), which is not part of the training data.



4519 combinations of the original feature vector. In Chapter 10 we will see an
 4520 example of finding a low-dimensional approximation of the original data
 4521 space by finding the principal components. Finding principal components
 4522 is closely related to concepts of eigenvalue and singular value decomposi-
 4523 tion as introduced in Chapter 4. For the high-dimensional representation,
 4524 we will see an explicit *feature map* $\phi(\cdot)$ that allows us to represent inputs
 4525 x_n using a higher dimensional representation $\phi(x_n)$. The main motiva-
 4526 tion for higher dimensional representations is that we can construct new
 4527 features as non-linear combinations of the original features, which in turn
 4528 may make the learning problem easier. We will discuss the feature map
 4529 in Section 9.2 and show how this feature map leads to a *kernel* in Sec-
 4530 tion 12.3.3. In recent years deep learning methods (Goodfellow et al.,
 4531 2016) have shown promise in using the data itself to learn the features,
 4532 and has been very successful in areas such as computer vision, speech
 4533 recognition and natural language processing. We will not cover neural
 4534 networks in this part of the book, but the reader is referred to Section 5.6
 4535 for the mathematical description of backpropagation, a key concept for
 4536 training neural networks.

4537 *Models are Functions*
 4538 predictor Once we have data in an appropriate vector representation, we can get to
 4539 the business of constructing a predictive function (known as a *predictor*).
 4540 In Chapter 1 we did not yet have the language to be precise about models.
 4541 Using the concepts from the first part of the book, we can now introduce
 4542 what "model" means. We present two major approaches in this book: a
 4543 predictor as a function, and a predictor as a probabilistic model. We de-
 4544 scribe 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

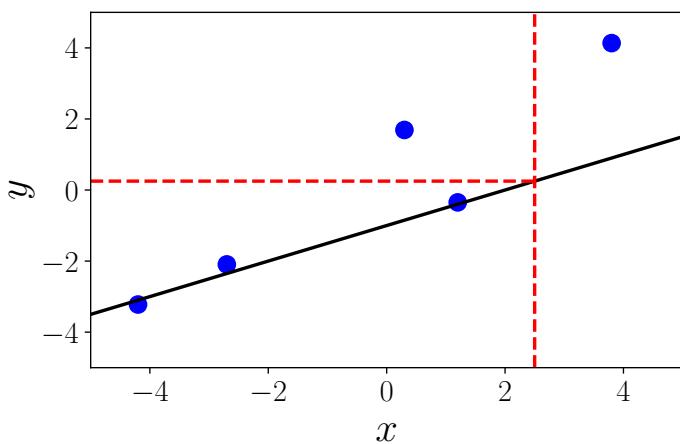


Figure 8.2 Example function (black solid diagonal line) and its prediction at $x = 2.5$. That is $f(2.5) = 0.25$.

the output to be a single number, i.e., a real-valued scalar output. This can be written as

$$f : \mathbb{R}^D \rightarrow \mathbb{R}, \quad (8.1)$$

where the input vector x is D -dimensional (has D features), and the function f then applied to it (written as $f(x)$) 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

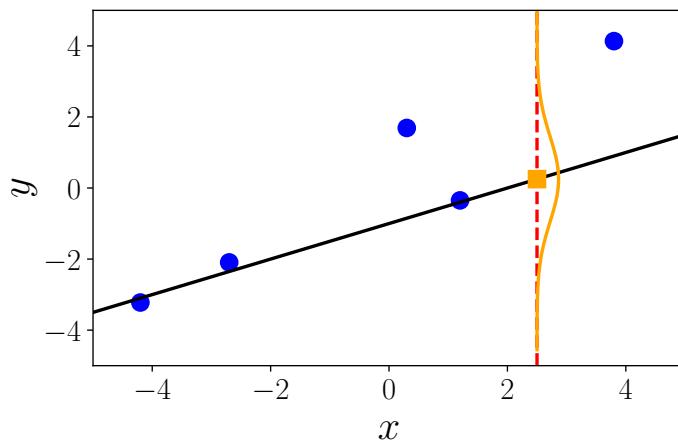
$$f(x) = \boldsymbol{\theta}^\top \mathbf{x} + \theta_0. \quad (8.2)$$

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.

Figure 8.3 Example function (black solid diagonal line) and its predictive uncertainty at $x = 2.5$ (drawn as a Gaussian).



4565 Instead of considering a predictor as a single function, we could con-
 4566 sider predictors to be probabilistic models, i.e., models describing the dis-
 4567 tribution of possible functions. We limit ourselves in this book to the spe-
 4568 cial case of distributions with finite dimensional parameters, which allows
 4569 us to describe probabilistic models without needing stochastic processes
 4570 and random measures. For this special case we can think about probabilis-
 4571 tic models as multivariate probability distributions, which already allow
 4572 for a rich class of models.

4573 We will introduce how to use concepts from probability (Chapter 6) to
 4574 define machine learning models in Section 8.3, and introduce a graphical
 4575 language for describing probabilistic models in a compact way in Sec-
 4576 tion 8.4.

4577 *Learning is Finding Parameters*

4578 The goal of learning is to find a model and its corresponding parame-
 4579 ters such that the resulting predictor will perform well on unseen data.
 4580 There are conceptually three distinct algorithmic phases when discussing
 4581 machine learning algorithms:

- 4582 1 prediction or inference
- 4583 2 training or parameter estimation
- 4584 3 hyperparameter tuning or model selection

4585 The prediction phase is when we use a trained predictor on previously un-
 4586 seen test data. In other words, the parameters and model choice is already
 4587 fixed and the predictor is applied to new vectors representing new input
 4588 data points. As outlined in Chapter 1 and the previous subsection, we will
 4589 consider two schools of machine learning in this book, corresponding to
 4590 whether the predictor is a function or a probabilistic model. When we
 4591 have a probabilistic model (discussed further in Section 8.3) the predic-
 4592 tion phase is called inference.

4593 *Remark.* Unfortunately there is no agreed upon naming for the different
 4594 algorithmic phases. The word inference is sometimes also used to mean
 4595 parameter estimation of a probabilistic model, and less often may be also
 4596 used to mean prediction for non-probabilistic models. ◇

4597 The training or parameter estimation phase is when we adjust our pre-
 4598 dictive model based on training data. We would like to find good predic-
 4599 tors given training data, and there are two main strategies for doing so:
 4600 finding the best predictor based on some measure of quality (sometimes
 4601 called finding a point estimate), or using Bayesian inference. Finding a
 4602 point estimate can be applied to both types of predictors, but Bayesian in-
 4603 ference requires probabilistic models. For the non-probabilistic model, we
 4604 follow the principle of *empirical risk minimization*, which we describe in
 4605 Section 8.1. Empirical risk minimization directly provides an optimization
 4606 problem for finding good parameters. With a statistical model the princi-
 4607 ple of *maximum likelihood* is used to find a good set of parameters (Sec-
 4608 tion 8.2). We can additionally model the uncertainty of parameters using
 4609 a probabilistic model, which we will look at in more detail in Section 8.3.

empirical risk
minimization

maximum likelihood

4610 We use numerical methods to find good parameters that “fit” the data,
 4611 and most training methods can be thought of as hill climbing approaches
 4612 to find the maximum of an objective, for example the maximum of a like-
 4613 lihood. To apply hill-climbing approaches we use the gradients described
 4614 Chapter 5 and implement numerical optimization approaches from Chap-
 4615 ter 7.

The convention in
optimization is to
minimize objectives.
Hence, there is often
an extra minus sign
in machine learning
objectives.

4616 As mentioned in Chapter 1, we are interested in learning a model based
 4617 on data such that it performs well on future data. It is not enough for
 4618 the model to only fit the training data well, the predictor needs to per-
 4619 form well on unseen data. We simulate the behaviour of our predictor on
 4620 future unseen data using cross validation (Section 8.1.4). As we will see
 4621 in this chapter, to achieve the goal of performing well on unseen data,
 4622 we will need to balance between fitting well on training data and finding
 4623 “simple” explanations of the phenomenon. This trade off is achieved us-
 4624 ing regularization (Section 8.1.3) or by adding a prior (Section 8.2.2). In
 4625 philosophy, this is considered to be neither induction or deduction, and
 4626 is called *abduction*. According to the Stanford Encyclopedia of Philosophy,
 4627 abduction is the process of inference to the best explanation (Douven,
 4628 2017).

abduction
A good movie title is
“AI abduction”.

4629 We often need to make high level modeling decisions about the struc-
 4630 ture of the predictor, such as the number of components to use or the
 4631 class of probability distributions to consider. The choice of the number of
 4632 components is an example of a *hyperparameter*, and this choice can af-
 4633 fect the performance of the model significantly. The problem of choosing
 4634 between different models is called *model selection*, which we describe in
 4635 Section 8.5. For non-probabilistic models, model selection is often done

hyperparameter
model selection

4636 using *cross validation*, which is described in Section 8.1.4. We also use cross validation
 4637 model selection to choose hyperparameters of our model.

4638 *Remark.* The distinction between parameters and hyperparameters is some-
 4639 what arbitrary, and is mostly driven by the distinction between what can
 4640 be numerically optimized versus what needs to use search techniques.
 4641 Another way to consider the distinction is to consider parameters as the
 4642 explicit parameters of a probabilistic model, and to consider hyperparam-
 4643 eters (higher level parameters) as parameters that control the distribution
 4644 of these explicit parameters. ◇

4645 In the following sections, we will look at three flavours of machine
 4646 learning: empirical risk minimization (Section 8.1), the principle of maxi-
 4647 mum likelihood (Section 9.2), and probabilistic modeling (Section 8.3).

4648 8.1 Empirical Risk Minimization

4649 After having all the mathematics under our belt, we are now in a posi-
 4650 tion to introduce what it means to learn. The “learning” part of machine
 4651 learning boils down to estimating parameters based on training data.

4652 In this section we consider the case of a predictor that is a function,
 4653 and consider the case of probabilistic models in Section 8.2. We describe
 4654 the idea of empirical risk minimization, which was originally popularized
 4655 by the proposal of the support vector machine (described in Chapter 12).
 4656 However, its general principles are widely applicable and allows us to ask
 4657 the question of what is learning without explicitly constructing probabilis-
 4658 tic models. There are four main design choices, which we will cover in
 4659 detail in the following subsections:

- 4660 **Section 8.1.1** What is the set of functions we allow the predictor to take?
- 4661 **Section 8.1.2** How do we measure how well the predictor performs on
 4662 the training data?
- 4663 **Section 8.1.3** How do we construct predictors from only training data
 4664 that performs well on unseen test data?
- 4665 **Section 8.1.4** What is the procedure for searching over the space of mod-
 4666 els?

4667 8.1.1 Hypothesis Class of Functions

Assume we are given N examples $\mathbf{x}_n \in \mathbb{R}^D$ and corresponding scalar la-
 bels $y_n \in \mathbb{R}$. We consider the supervised learning setting, where we obtain
 pairs $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$. Given this data, we would like to estimate a
 predictor $f(\cdot, \boldsymbol{\theta}) : \mathbb{R}^D \rightarrow \mathbb{R}$, parameterized by $\boldsymbol{\theta}$. We hope to be able to
 find a good parameter $\boldsymbol{\theta}^*$ such that we fit the data well

$$f(\mathbf{x}_n, \boldsymbol{\theta}^*) \approx y_n \quad \text{for all } n = 1, \dots, N. \quad (8.3)$$

4668 In this section, we use the notation $\hat{y}_n = f(\mathbf{x}_n, \boldsymbol{\theta}^*)$ to represent the output
4669 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,

$$f(\mathbf{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}_n + \theta_0. \quad (8.4)$$

Observe that the predictor takes the vector of features representing a single example \mathbf{x}_n as input and produces a real valued output. That is $f : \mathbb{R}^D \rightarrow \mathbb{R}$. 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 \mathbf{x}_n , that is $\tilde{\mathbf{x}}_n = \begin{bmatrix} \mathbf{x}_n \\ 1 \end{bmatrix}$. 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

$$f(\tilde{\mathbf{x}}_n, \tilde{\boldsymbol{\theta}}) = \tilde{\boldsymbol{\theta}}^\top \tilde{\mathbf{x}}_n. \quad (8.5)$$

We will often overload the notation in this book to have tidier presentation: both \mathbf{x}_n as well as $\boldsymbol{\theta}_n$ are used to refer to the new concatenated vectors.

Instead of a linear function, we may wish to consider non-linear functions as predictors. Recent advances in neural network allows for efficient computation of more complex non-linear function classes.

4670 *Remark.* For ease of presentation we will describe empirical risk minimiza-
4671 tion in terms of supervised learning. This simplifies the definition of the
4672 hypothesis class and the loss function. ◇

4673 Given the class of functions we want to search for a good predictor,
4674 we now move on to the second ingredient of empirical risk minimization:
4675 how to measure how well the predictor fits the training data.

8.1.2 Loss Function for Training

4677 Consider the label y_n for a particular example; and the corresponding
4678 prediction \hat{y}_n that we make based on \mathbf{x}_n . To define what it means to fit the
4679 data well, we need to specify a *loss function* $\ell(y_n, \hat{y}_n)$ that takes two values
4680 as input and produces a non-negative number (referred to as the loss)
4681 representing how much error we have made on this particular prediction.

loss function

4682 Our goal for finding a good parameter vector $\boldsymbol{\theta}^*$ is to minimize the average
4683 loss on the set of N training examples.

independent and
identically
distributed

training set

The word error is often used to mean loss.

One assumption that is commonly made in machine learning is that the set of examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ are *independent and identically distributed*. The word independent (Section 6.4.5) means that two data points $(\mathbf{vec}\mathbf{x}_i, y_i)$ and $(\mathbf{vec}\mathbf{x}_j, y_j)$ 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* $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ which we collect into an example matrix \mathbf{X} and label vector \mathbf{y} , the average loss is given by

$$\mathbf{R}_{\text{emp}}(f, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, \hat{y}_n) \quad (8.6)$$

4684 empirical risk
4685 empirical risk
4686 minimization
4687

where $\hat{y}_n = f(\mathbf{x}_n, \boldsymbol{\theta}^*)$. Equation (8.6) is called the *empirical risk*. Note that the empirical risk depends on three arguments, the predictor f and the data \mathbf{X}, \mathbf{y} . This general strategy for learning is called *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

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \boldsymbol{\theta}))^2, \quad (8.7)$$

where we have substituted the predictor $\hat{y}_n = f(\mathbf{x}_n, \boldsymbol{\theta})$. By using our choice of a linear predictor $f(\mathbf{x}_n, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}_n$ we obtain the optimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^N (y_n - \boldsymbol{\theta}^\top \mathbf{x}_n)^2. \quad (8.8)$$

This equation can be equivalently expressed in matrix form by collecting the labels into a vector $\mathbf{y} := [y_1, \dots, y_N]^\top \in \mathbb{R}^N$ and collecting the dataset into a matrix $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$.

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2. \quad (8.9)$$

This is known as the least squares problem. There exists 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

$$\mathbf{R}_{\text{true}}(f) = \mathbb{E}_{\mathbf{x}, y} \ell(y, f(\mathbf{x})) \quad (8.10)$$

where y is the ground truth label, and $f(\mathbf{x})$ is the prediction based on the data \mathbf{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 often a 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. \diamond

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. It is important for the user to not cycle back to a new round of training after having observed the test set. 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

Even knowing only the performance of the predictor on the test set leaks information (Blum and Hardt, 2015).

It turns out that empirical risk minimization can lead to *overfitting*, that

overfitting

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 $\mathbf{R}_{\text{emp}}(f, \mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ 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 $\mathbf{R}_{\text{emp}}(f, \mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ if the test risk is much larger than the training risk, this is an indication of 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.

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

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2. \quad (8.11)$$

with the “regularized” problem by adding a penalty term involving only $\boldsymbol{\theta}$

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2. \quad (8.12)$$

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 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 probabilistic models as the prior probability of the parameters. Recall from Section 6.6 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

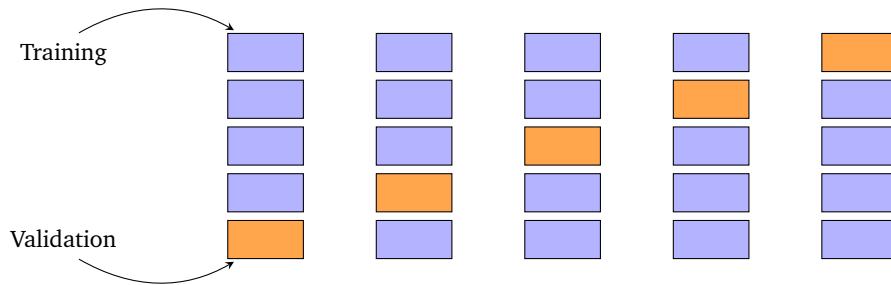


Figure 8.4 K -fold cross validation. The data set is divided into $K = 5$ chunks, $K - 1$ of which serve as the training set (blue) and one as the validation set (orange).

4749 sometimes referred to as the *validation set*. The validation set is a subset
 4750 of the available training data that we keep aside. A practical issue with
 4751 this approach is that the amount of data is limited, and ideally we would
 4752 use as much of the data available to train the model. This would require to
 4753 keep our validation set \mathcal{V} small, which then would lead to a noisy estimate
 4754 (with high variance) of the predictive performance. One solution to these
 4755 contradictory objectives (large training set, large validation set) is to use
 4756 *cross validation*. K -fold cross validation effectively partitions the data into
 4757 K chunks, $K - 1$ of which form the training set $\tilde{\mathcal{D}}$, and the last chunk
 4758 serves as the validation set \mathcal{V} (similar to the idea outlined above). Cross-
 4759 validation iterates through (ideally) all combinations of assignments of
 4760 chunks to $\tilde{\mathcal{D}}$ and \mathcal{V} , see Figure 8.4. This procedure is repeated for all K
 4761 choices for the validation set, and the performance of the model from the
 4762 K runs is averaged.

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) of the trained model on the validation set). More precisely, for each partition k the training data $\tilde{\mathcal{D}}^{(k)}$ produces a predictor $f^{(k)}$, which is then applied to validation set $\mathcal{V}^{(k)}$ to compute the empirical risk $R(f^{(k)}, \mathcal{V}^{(k)})$. We cycle through all possible partitionings of validation and training sets and compute the average generalization error of the predictor. Cross-validation approximates the expected generalization error

$$\mathbb{E}_{\mathcal{V}}[R(f, \mathcal{V})] \approx \frac{1}{K} \sum_{k=1}^K R(f^{(k)}, \mathcal{V}^{(k)}), \quad (8.13)$$

4763 where $R(f^{(k)}, \mathcal{V}^{(k)})$ is the risk (e.g., RMSE) on the validation set $\mathcal{V}^{(k)}$ for
 4764 predictor $f^{(k)}$. The approximation has two sources: first due to the finite
 4765 training set which results in not the best possible $f^{(k)}$ and second due to
 4766 the finite validation set which results in an inaccurate estimation of the
 4767 risk $R(f^{(k)}, \mathcal{V}^{(k)})$. A potential disadvantage of K -fold cross validation is
 4768 the computational cost of training the model K times, which can be bur-
 4769 densome if the training cost is computationally expensive. In practice, it

validation set

cross validation

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.

In this section we saw that empirical risk minimization is based on the following concepts: the hypothesis class of functions, the loss function and regularization. In Section 8.2 we will see the effect of using a probability distribution to replace the idea of loss functions and regularization.

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* (Hastie et al., 2001; 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).

The idea of regularization has its roots in the solution of ill-posed inverse problems (Neumaier, 1998). The approach presented here is called Tikhonov regularization, and there is a closely related constrained version called Ivanov regularization. Tikhonov regularization 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. In Section 8.2.1 we introduce the likelihood, which is analogous to the concept of loss functions (Section 8.1.2) in empirical risk minimization. The concept of priors (Section 8.2.2) is analogous to the concept of regularization (Section 8.1.3).

8.2.1 Maximum Likelihood Estimation

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 a random variable \mathbf{x} and for a family of probability densities $p(\mathbf{x} | \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, the *negative log likelihood* is given by

$$\mathcal{L}_{\mathbf{x}}(\boldsymbol{\theta}) = -\log p(\mathbf{x} | \boldsymbol{\theta}). \quad (8.14)$$

The notation $\mathcal{L}_{\mathbf{x}}(\boldsymbol{\theta})$ emphasizes the fact that the parameter $\boldsymbol{\theta}$ is varying and the data \mathbf{x} is fixed. We very often drop the reference to \mathbf{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 interpret what the probability density $p(\mathbf{x} | \boldsymbol{\theta})$ 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 \mathbf{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 how likely a particular setting of $\boldsymbol{\theta}$ is for the observations \mathbf{x} . Based on this second view, the maximum likelihood estimator gives us the most likely parameter $\boldsymbol{\theta}$ for the set of data.

We consider the supervised learning setting, where we obtain pairs $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ with $\mathbf{x}_n \in \mathbb{R}^D$ and labels $y_n \in \mathbb{R}$. We are interested in constructing a predictor that takes a feature vector \mathbf{x}_n as input and produces a prediction y_n (or something close to it). That is given a vector \mathbf{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 6.5) with zero mean, $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$. We further assume that the linear model $\mathbf{x}_n^\top \boldsymbol{\theta}$ is used for prediction. This means we specify a Gaussian likelihood for each example label pair \mathbf{x}_n, y_n ,

$$p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) = \mathcal{N}(y_n | \mathbf{x}_n^\top \boldsymbol{\theta}, \sigma^2). \quad (8.15)$$

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.

independent and identically distributed

We assume that the set of examples $(x_1, y_1), \dots, (x_N, y_N)$ are *independent and identically distributed*. The word independent (Section 6.4.5) implies that the likelihood of the whole dataset ($\mathbf{y} = [y_1, \dots, y_N]^\top$ and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$) factorizes into a product of the likelihoods of each individual example

$$p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) = \prod_{n=1}^N p(y_n | \mathbf{x}_n, \boldsymbol{\theta}), \quad (8.16)$$

where $p(y_n | \mathbf{x}_n, \boldsymbol{\theta})$ 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

$$\mathcal{L}(\boldsymbol{\theta}) = -\log p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) = -\sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \boldsymbol{\theta}). \quad (8.17)$$

4831 While it is tempting to interpret the fact that $\boldsymbol{\theta}$ is on the right of the conditioning in $p(y_n | \mathbf{x}_n, \boldsymbol{\theta})$ (8.15), and hence should be interpreted as observed
4832 and fixed, this interpretation is incorrect. The negative log likelihood $\mathcal{L}(\boldsymbol{\theta})$
4833 is a function of $\boldsymbol{\theta}$.

Therefore, to find a good parameter vector $\boldsymbol{\theta}$ that explains the data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ well, we look for a $\boldsymbol{\theta}$ that minimizes the negative log likelihood

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \quad (8.18)$$

4835 *Remark.* The negative sign in (8.17) is a historical artefact that is due
4836 to the convention that we want to maximize likelihood, but numerical
4837 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

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = -\sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) \quad (8.19)$$

$$= -\sum_{n=1}^N \log \mathcal{N}(y_n | \mathbf{x}_n^\top \boldsymbol{\theta}, \sigma^2) \quad (8.20)$$

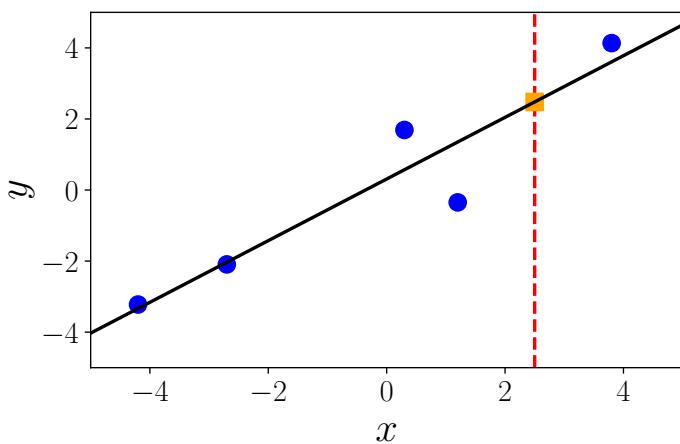


Figure 8.5 For the given data, the maximum likelihood estimate of the parameters results in the black diagonal line. The orange square shows the value of the maximum likelihood prediction at $x = 2.5$.

$$= -\sum_{n=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - \mathbf{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2}\right) \quad (8.21)$$

$$= -\sum_{n=1}^N \log \exp\left(-\frac{(y_n - \mathbf{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2}\right) - \sum_{n=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}} \quad (8.22)$$

$$= \sum_{n=1}^N \frac{(y_n - \mathbf{x}_n^\top \boldsymbol{\theta})^2}{2\sigma^2} - \sum_{n=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}}. \quad (8.23)$$

(8.24)

Observe that the first term in the last equation above is the least squares problem.

4838 It turns out that for Gaussian likelihoods the resulting optimization
 4839 problem corresponding to maximum likelihood estimation has a closed-
 4840 form solution. We will see more details on this in Chapter 9. Maximum
 4841 likelihood estimation may suffer from overfitting, analogous to unregu-
 4842 larized empirical risk minimization (Section 9.2.3). For other likelihood
 4843 functions, i.e., if we model our noise with non-Gaussian distributions,
 4844 maximum likelihood estimation may not have a closed-form analytic solu-
 4845 tion. In this case, we resort to numerical optimization methods discussed
 4846 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

posterior
prior
likelihood

a given prior, after observing some data \mathbf{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 \mathbf{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 $p(\boldsymbol{\theta} | \mathbf{x})$ (the more specific knowledge) on the parameters $\boldsymbol{\theta}$ from general *prior* statements (prior distribution) $p(\boldsymbol{\theta})$ and the function $p(\mathbf{x} | \boldsymbol{\theta})$ that links the parameters $\boldsymbol{\theta}$ and the observed data \mathbf{x} (called the *likelihood*):

$$p(\boldsymbol{\theta} | \mathbf{x}) = \frac{p(\mathbf{x} | \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{x})}. \quad (8.25)$$

Recall that we are interested in finding the parameter $\boldsymbol{\theta}$ that maximizes likelihood, and the distribution $p(\mathbf{x})$ 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,

$$p(\boldsymbol{\theta} | \mathbf{x}) \propto p(\mathbf{x} | \boldsymbol{\theta})p(\boldsymbol{\theta}). \quad (8.26)$$

maximum a posteriori estimation

4848 The proportion relation above hides the density of the data $p(\mathbf{x})$ which
4849 may be difficult to estimate. Instead of estimating the minimum of the
4850 negative log likelihood, we now estimate the minimum of the negative log
4851 posterior, which is referred to as *maximum a posteriori estimation* (MAP).

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 6.5). Note that the conjugate prior of a Gaussian is also a Gaussian (Section 6.6.1) 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.

4852 The idea of including prior knowledge about where good parameters
4853 lie is widespread in machine learning. An alternative view which we saw
4854 in Section 8.1.3 is the idea of regularization, which introduces an addi-
4855 tional term that biases the resulting parameters to be close to the origin.
4856 Maximum a posteriori estimation can be considered to bridge the non-
4857 probabilistic and probabilistic worlds as it explicitly acknowledges the
4858 need for a prior distribution but it still only produces a point estimate
4859 of the parameters.

4860 *Remark.* The maximum likelihood estimate $\boldsymbol{\theta}_{\text{ML}}$ possesses the following
4861 properties (Lehmann and Casella, 1998; Efron and Hastie, 2016):

- 4862 • Asymptotic consistency: The MLE converges to the true value in the

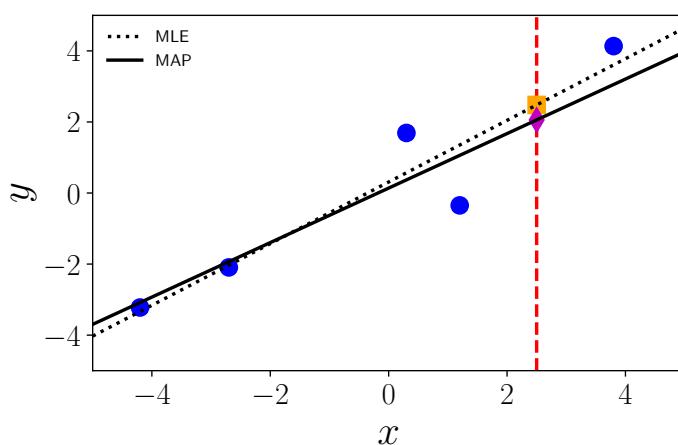


Figure 8.6
Comparing the Maximum Likelihood estimate and the Maximum A Posteriori estimate and their predictions at $x = 2.5$. The prior biases the slope to be less steep and the intercept to be closer to zero.

4863 limit of infinitely many observations, plus a random error that is approximately normal.
 4864

- 4865 • The size of the samples necessary to achieve these properties can be quite large.
- 4866 • The error's variance decays in $1/N$ where N is the number of data points.
- 4867 • Especially, in the “small” data regime, maximum likelihood estimation
 4868 can lead to *overfitting*.

4871 ◇

4872 The principle of maximum likelihood estimation (and maximum apos-
 4873 teriori estimation) uses probabilistic modeling to reason about the un-
 4874 certainty in the data and model parameters. However we have not yet
 4875 taken probabilistic modeling to its full extent. In this section the resulting
 4876 training procedure still produces a point estimate of the predictor, that is
 4877 training returns one single set of parameter values that represent the best
 4878 predictor. In Section 8.3 we will take the view that the parameter values
 4879 should also be treated as random variables.

4880 *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,

$$p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) = \varphi(\boldsymbol{\theta}^\top \mathbf{x}_n) \quad (8.27)$$

4881 we can consider other models for other prediction tasks, such as binary
 4882 classification or modelling count data (McCullagh and Nelder, 1989). An

4883 alternative view of this is to consider likelihoods that are from the ex-
 4884ponential family (Section 6.6). The class of models, which have linear
 4885 dependence between parameters and data, and have potentially nonlin-
 4886ear transformation φ (called a link function) is referred to as generalized
 4887 linear models (Agresti, 2002, Chapter 4).

4888 Maximum likelihood estimation has a rich history, and was originally
 4889 proposed by Sir Ronald Fisher in the 1930s. We will expand upon the idea
 4890 of a probabilistic model in Section 8.3. One debate among researchers
 4891 who use probabilistic models, is the discussion between Bayesian and
 4892 frequentist statistics. As mentioned in Section 6.1.1 it boils down to the
 4893 definition of probability. Recall from Section 6.1 that one can consider
 4894 probability to be a generalization (by allowing uncertainty) of logical rea-
 4895 soning (Cheeseman, 1985; Jaynes, 2003). The method of maximum like-
 4896 lihood estimation is frequentist in nature, and the interested reader is
 4897 pointed to Efron and Hastie (2016) for a balanced view of both Bayesian
 4898 and frequentist statistics.

4899 There are some probabilistic models where maximum likelihood es-
 4900 timation may not be possible. The reader is referred to more advanced
 4901 statistical textbooks, e.g., Casella and Berger (2002), for approaches such
 4902 as method of moments, M -estimation and estimating equations.

4903 8.3 Probabilistic Modeling

4904 In machine learning, we are frequently concerned with the interpretation
 4905 and analysis of data, e.g., for prediction of future events and decision mak-
 4906 ing. To make this task more tractable, we often build models that describe
 4907 the process that generates the data. For example, when we want to de-
 4908 scribe the outcome of a coin-flip experiment, we can describe this process
 4909 using a Bernoulli distribution as described in Chapter 6. In this example,
 4910 we can say that an outcome $x \in \{\text{head, tail}\}$ can be described as the con-
 4911 ditional distribution $p(x | \mu)$ where x is the outcome of the experiment
 4912 and μ is the probability of “heads”. Probabilistic models represent all as-
 4913 pects of our experiment as probability distributions, which is the focus of
 4914 this section. The benefit of using probabilistic models is that we have a
 4915 unified set of tools from probability (Chapter 6) for modeling, inference,
 4916 parameter estimation and model selection.

4917 *Remark.* Thinking about empirical risk minimization (Section 8.1) as “prob-
 4918 ability free” is incorrect. There is an underlying unknown probability dis-
 4919 tribution $p(\mathbf{x}, y)$ that governs the data generation, but the approach of
 4920 empirical risk minimization is agnostic to that choice of distribution. The
 4921 design of the loss function (Section 8.1.2) does however implicitly select
 4922 for particular conditional probability distributions. This is in contrast to
 4923 standard statistical approaches that explicitly require the knowledge of
 4924 $p(\mathbf{x}, y)$. Furthermore, since the distribution is a joint distribution on both

examples \mathbf{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 . \diamond

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

$$y_n = \mathbf{x}_n^\top \boldsymbol{\theta} + \varepsilon \quad \text{where } \varepsilon \sim \mathcal{N}(0, \sigma^2). \quad (8.28)$$

By making this assumption, we obtain the likelihood described in Section 8.2.1 which 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 \mathbf{x}_* we can do this by using $\boldsymbol{\theta}^*$ directly in the likelihood function that connects parameters and data to obtain a prediction $p(\mathbf{x}_* | \boldsymbol{\theta}^*)$. 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

$$p(\mathbf{x}_*) = \mathbb{E}_{\boldsymbol{\theta}}[p(\mathbf{x}_* | \boldsymbol{\theta})] = \int p(\mathbf{x}_* | \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \quad (8.29)$$

⁴⁹⁴⁵ no longer depends on the parameters θ – they have been marginalized/
 Bayesian inference⁴⁹⁴⁶ integrated out. This is referred to as *Bayesian inference*.

⁴⁹⁴⁷ 8.3.2 Latent Variables

⁴⁹⁴⁸ Taking a probabilistic view of machine learning implies that we also want
⁴⁹⁴⁹ to treat predictors (models) as random variables. While data x_1, \dots, x_N
⁴⁹⁵⁰ can be observed, these models themselves possess quantities/parameters
⁴⁹⁵¹ that are not directly observed. In the coin-flip experiment described in the
⁴⁹⁵² introduction to this section, the probability μ of “heads” is typically not
⁴⁹⁵³ known and depends on the coin we use. We describe these kind of un-
⁴⁹⁵⁴ known quantities with a random variable. Given that these random vari-
⁴⁹⁵⁵ ables cannot be observed but only inferred we call them *hidden variables*
⁴⁹⁵⁶ or *latent variables*. While we can make general statements about plausible
⁴⁹⁵⁷ values of these latent variables prior to observing any data (e.g., by using
⁴⁹⁵⁸ a prior distribution), a key challenge in machine learning is to infer more
⁴⁹⁵⁹ about these unobserved variables given a dataset.

⁴⁹⁶⁰ *Remark.* We tend to use the notation θ to represent the vector of un-
⁴⁹⁶¹ observed random variables. We also refer to θ as the *parameters* of the
⁴⁹⁶² model. ◇

⁴⁹⁶³ Recall from Section 8.2.2 that Bayes’ Theorem gives us a principled tool
⁴⁹⁶⁴ to update our probability distribution over θ given observed data. Here
⁴⁹⁶⁵ we go from a prior distribution on the latent variables to a posterior dis-
⁴⁹⁶⁶ tribution after taking the likelihood into account. Since we can write the
⁴⁹⁶⁷ numerator of (8.25) as the joint distribution $p(\theta, x) = p(x | \theta)p(\theta)$ of
⁴⁹⁶⁸ the latent variables and the data, this joint distribution is of central im-
⁴⁹⁶⁹ portance and sufficient to compute the quantities of interest, either by
⁴⁹⁷⁰ conditioning or marginalization (Section 6.2.1). For example, we obtain
⁴⁹⁷¹ the posterior distribution by conditioning so that we can make informed
⁴⁹⁷² statements about the latent variables, and we obtain the *marginal likeli-
⁴⁹⁷³ hood* (or *evidence*) $p(x)$ via marginalization where we integrate out the
⁴⁹⁷⁴ latent variables (8.30).

$$p(x) = \int p(x | \theta)p(\theta)d\theta. \quad (8.30)$$

⁴⁹⁷⁵ The marginal likelihood is very useful for model selection as we will dis-
⁴⁹⁷⁶ cuss in Section 8.5. Hence we define a probabilistic model in machine
⁴⁹⁷⁷ learning as the joint distribution $p(\theta, x)$ of all latent and observed vari-
⁴⁹⁷⁸ ables.

⁴⁹⁷⁹ Unfortunately, when we choose non-conjugate priors (Section 6.6.1),
⁴⁹⁸⁰ we cannot compute the posterior distribution using Bayes’ theorem be-
⁴⁹⁸¹ cause the computations become intractable. A solution to this problem is
⁴⁹⁸² to estimate a single parameter vector θ^* that explains the available data
⁴⁹⁸³ “best”, e.g., by maximum likelihood estimation as discussed in Section 8.2.

4972 Then, we can make a prediction of new data directly via the likelihood
 4973 $p(\mathbf{x} | \boldsymbol{\theta}^*)$ – without needing integration.

4974 *Remark.* In the machine learning literature, there can be a somewhat ar-
 4975bitrary separation between “variables” and “parameters”. While parame-
 4976ters are estimated (e.g., via maximum likelihood) variables are usually
 4977marginalized out as in (8.30). In this book, we are not so strict with this
 4978separation because, in principle, we can place a prior on any parameter
 4979and integrate it out, which would then turn the parameter into a variable
 4980according to the separation above. ◇

4981 There are modelling situations in machine learning where we may wish
 4982 to introduce new random variables \mathbf{z} into the problem. In these scenar-
 4983ios, the direct model of the problem (involving only $\mathbf{x}, \boldsymbol{\theta}$) may be com-
 4984putationally difficult to solve, but introducing a set of latent variables \mathbf{z}
 4985allows us to design an efficient algorithm. We will see an example of this
 4986in Chapter 11, where we introduce the Gaussian mixture model and use
 4987it for density estimation.

4988 So far in this chapter we have seen three different flavours of learning
 4989 from data. First by using a non-probabilistic approach called empirical risk
 4990 minimization 8.1. Second by introducing the idea of likelihood and prior,
 4991 and maximizing them 8.2. Third by taking probabilistic modeling to its
 4992 full extent in this section by considering the parameters of the likelihood
 4993 and prior to also be random variables.

4994 We can exploit the fact that all the elements of a probabilistic model
 4995 are random variables to have a unified language for representing them.
 4996 In Section 8.4 we will see a concise graphical language for representing
 4997 probabilistic models. We will use this graphical language to describe the
 4998 probabilistic models in Chapter 9, 10 and 11.

4999 Recall from the introduction to this chapter that we often need to make
 5000 high level modeling decisions about the structure of the predictor. These
 5001 modeling decisions (hyperparameters) affect the prediction performance,
 5002 but cannot be selected directly using the approaches we have seen so far.
 5003 We will describe the problem of model selection in Section 8.5.

5004 *Further Reading*

5005 Probabilistic models in machine learning Bishop (2006); Barber (2012);
 5006 Murphy (2012) provide a way for users to capture uncertainty about
 5007 data and predictive models in a principled fashion. Ghahramani (2015)
 5008 presents a short review of probabilistic models in machine learning. Given
 5009 a probabilistic model, we may be lucky enough to be able to compute pa-
 5010 rameters of interest analytically. However, in general, analytic solutions
 5011 are rare and computational methods such as sampling (Brooks et al.,
 5012 2011) and variational inference (Blei et al., 2017) are used.

5013 In recent years, there have been several proposed programming lan-
 5014guages that aim to treat the variables defined in software as random vari-

ables 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:

```
5020 Stan http://mc-stan.org/
5021 Edward http://edwardlib.org/
5022 PyMC https://docs.pymc.io/
5023 Pyro http://pyro.ai/
5024 Tensorflow Probability https://github.com/tensorflow/probability
5025 Infer.NET http://infernet.azurewebsites.net/
5026 MXFusion https://github.com/amzn/MXFusion
```

This section may be
skipped at first
reading.
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
graphical model

5027 In this section we introduce a graphical language for specifying probabilistic models, called the *directed graphical model*. They provide 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 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 6.4.5.

5028 In a *graphical model*, nodes are random variables; in Figure 8.7(a), the
5029 nodes of the random variables a, b, c represent their respective (marginal)
5030 probabilities $p(a), p(b)$ and $p(c)$. Edges represent probabilistic relations
5031 between variables, e.g., conditional probabilities.

5032 *Remark.* Not every distribution can be represented in a particular choice of
5033 graphical model. A discussion of this can be found in Bishop (2006). ◇

5034 Probabilistic graphical models have some convenient properties:

- 5035 • They are a simple way to visualize the structure of a probabilistic model
- 5036 • They can be used to design or motivate new kind of statistical models
- 5037 • Inspection of the graph alone gives us insight into properties, e.g., con-
5038 ditional independence
- 5039 • Complex computations for inference and learning in statistical models
5040 can be expressed in terms of graphical manipulations.

5056

8.4.1 Graph Semantics

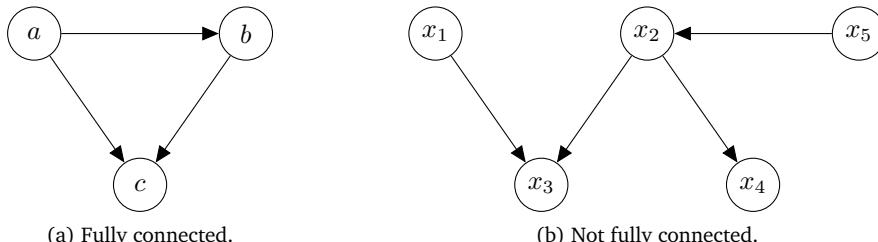


Figure 8.7
Examples of
directed graphical
models.

5057 *Directed graphical models/Bayesian networks* are a method for repre-
 5058 senting conditional dependencies in a probabilistic model. They provide
 5059 a visual description of the conditional probabilities, hence, providing a
 5060 simple language for describing complex interdependence. The modular
 5061 description also entails computational simplification. Directed links (ar-
 5062 rows) between two nodes (random variables) are conditional probabili-
 5063 ties. For example, the arrow between a and b in Figure 8.7(a) gives the
 5064 conditional probability $p(b | a)$ of b given a .

5065 Directed graphical models can be derived from joint distributions if we
 5066 know something about their factorization.

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.

Example 8.7

Consider the joint distribution

$$p(a, b, c) = p(c | a, b)p(b | a)p(a) \quad (8.31)$$

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).

5067 In general, we can construct the corresponding directed graphical model
 5068 from a factorized joint distribution as follows:

- 5069 1 Create a node for all random variables
 5070 2 For each conditional distribution, we add a directed link (arrow) to
 5071 the graph from the nodes corresponding to the variables on which the
 5072 distribution is conditioned on
- 5073 The graph layout depends on the choice of factorization of the joint dis-
 5074 tribution.

The graph layout
depends on the
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 $p(x_1, \dots, x_5)$ 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

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_5)p(x_2 | x_5)p(x_3 | x_1, x_2)p(x_4 | x_2). \quad (8.32)$$

In general, the joint distribution $p(\mathbf{x}) = p(x_1, \dots, x_K)$ is given as

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{Pa}_k) \quad (8.33)$$

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 (Example 6.8) where the probability that the outcome x of this experiment is “heads” is

$$p(x | \mu) = \text{Ber}(\mu). \quad (8.34)$$

We now repeat this experiment N times and observe outcomes x_1, \dots, x_N so that we obtain the joint distribution

$$p(x_1, \dots, x_N | \mu) = \prod_{n=1}^N p(x_n | \mu). \quad (8.35)$$

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 6.4.5 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 , $n = 1, \dots, N$. A

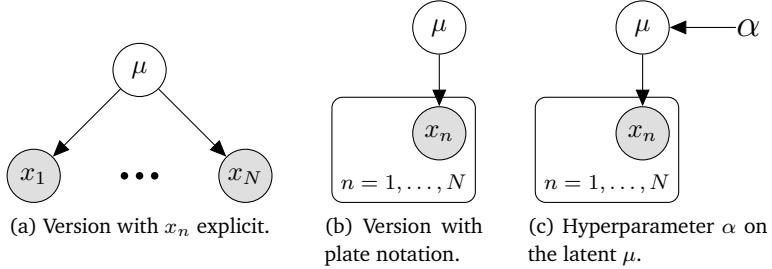


Figure 8.8
Graphical models
for a repeated
Bernoulli
experiment.

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 . A *hyper-prior* is a second layer of prior distributions on the parameters of the first layer of priors. 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.

8.4.2 Conditional Independence and D-Separation

Directed graphical models allow us to find conditional independence (Section 6.4.5) relationship properties of the joint distribution only by looking at the graph. A concept called *d-separation* (Pearl, 1988) is key to this.

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

$$\mathcal{A} \perp\!\!\!\perp \mathcal{B} | \mathcal{C}, \quad (8.36)$$

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 $\mathcal{A} \perp\!\!\!\perp \mathcal{B} | \mathcal{C}$.

Figure 8.9
D-separation
example.

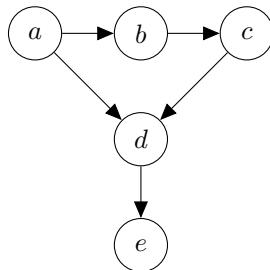
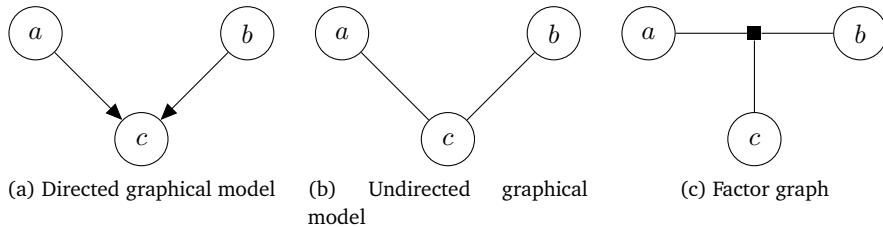


Figure 8.10 Three types of graphical models: (a) Directed graphical models (Bayesian network); (b) Undirected graphical models (Markov random field); (c) Factor graphs.



Example 8.9 (Conditional Independence)

Consider the graphical model in Figure 8.9. By visual inspection we see that

$$b \perp\!\!\!\perp d \mid a, c, \quad (8.37a)$$

$$a \perp\!\!\!\perp c \mid b, \quad (8.37b)$$

$$b \not\perp\!\!\!\perp d \mid c, \quad (8.37c)$$

$$a \not\perp\!\!\!\perp c \mid b, e. \quad (8.37d)$$

Directed graphical models allow a compact representation of probabilistic models, and we will see examples of directed graphical models in Chapter 9, 10 and 11. The representation along with the concept of conditional independence, allows us to factorize the respective probabilistic models into expressions that are easier to optimize. The graphical representation of the probabilistic model allows us to visually see the impact of design choices we have made on the structure of the model. We will discuss different ways to choose the structure in Section 8.5.

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

- *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)

Undirected
graphical models
Factor graphs

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).

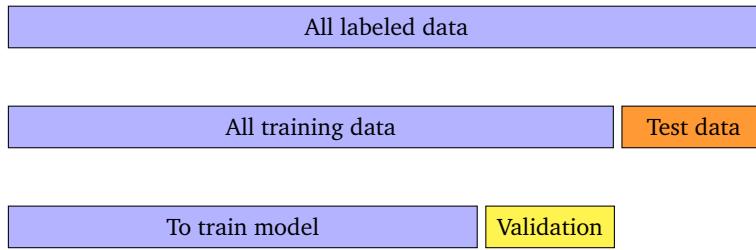
8.5 Model Selection

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 functional form of the likelihood) 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 datasets. For instance, a polynomial of degree 1 (a line $y = a_0 + a_1x$) 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.

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 dataset 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

Note that a polynomial $y = a_0 + a_1x + a_2x^2$ can also describe linear functions by setting $a_2 = 0$, i.e., it is strictly more expressive than a first-order polynomial.

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 level¹⁶⁷ is used to estimate¹⁶⁸ generalization performance for the best choice of model chosen by the inner¹⁶⁹ loop.



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 called the *test set* and the set used for choosing the best model is called the *validation set*.

$$\mathbb{E}_{\mathcal{V}}[G(\mathcal{V}) \mid M] \approx \frac{1}{K} \sum_{k=1}^K G(\mathcal{V}^{(k)} \mid M), \quad (8.38)$$

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 error¹⁸⁴ is defined as $\frac{\sigma}{\sqrt{K}}$,¹⁸⁵ where K is the number of experiments and σ the standard deviation.¹⁸⁶

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. We assume that simpler models are less prone to overfitting than complex models, and hence the objective of model selection is to find

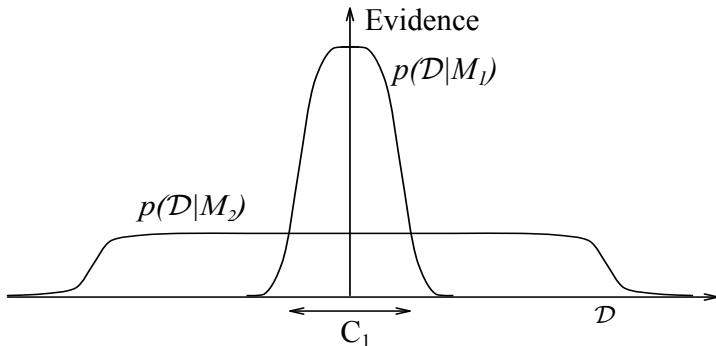


Figure 8.12
Bayesian inference embodies Occam's razor (MacKay, 2003a), see text for description.

5193 the simplest model that explains the data reasonably well. This concept is
 5194 also known as *Occam's Razor*.

Occam's Razor

5195 *Remark.* If we treat model selection as a hypothesis testing problem, we
 5196 are looking for the simplest hypothesis that is consistent with the data (Mur-
 5197 phy, 2012). \diamond

5198 One may consider placing a prior on models that favors simpler mod-
 5199 els. However, it is not necessary to do this: An ‘‘automatic Occam’s Ra-
 5200 zor’’ is quantitatively embodied in the application of Bayesian probabili-
 5201 ty (Spiegelhalter and Smith, 1980; MacKay, 1992; Jefferys and Berger,
 5202 1992). Figure 8.12 from MacKay (2003a) gives us the basic intuition why
 5203 complex and very expressive models may turn out to be a less probable
 5204 choice for modeling a given dataset \mathcal{D} . Let us think of the horizontal axis
 5205 representing the space of all possible datasets \mathcal{D} . If we are interested in
 5206 the posterior probability $p(M_i | \mathcal{D})$ of model M_i given the data \mathcal{D} , we can
 5207 employ Bayes’ theorem. Assuming a uniform prior $p(M)$ over all mod-
 5208 els, Bayes’ theorem rewards models in proportion to how much they pre-
 5209 dicted the data that occurred. This probability of the data given model
 5210 M_i , $p(\mathcal{D} | M_i)$, is called the *evidence* for M_i . A simple model M_1 can only
 5211 predict a small number of datasets, which is shown by $p(\mathcal{D} | M_1)$; a more
 5212 powerful model M_2 that has, e.g., more free parameters than M_1 , is able
 5213 to predict a greater variety of datasets. This means, however, that M_2
 5214 does not predict the datasets in region C_1 as strongly as M_1 . Suppose that
 5215 equal prior probabilities have been assigned to the two models. Then, if
 5216 the dataset falls in region C_1 , the less powerful model M_1 is the more
 5217 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

5218 Above, we argued that models need to be able to explain the data, i.e.,
 5219 there should be a way to generate data from a given model. Furthermore
 5220 if the model has been appropriately learned from the data, then we expect
 5221 that the generated data should be similar to the empirical data. For this,
 5222 it is helpful to phrase model selection as a hierarchical inference problem,
 5223 which allows us to compute the posterior distribution over models.

Let us consider a finite number of models $M = \{M_1, \dots, M_K\}$, where

Bayesian model selection generative process

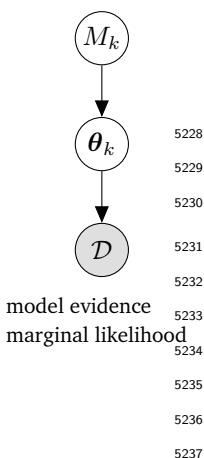
each model M_k possesses parameters θ_k . In *Bayesian model selection*, we place a prior $p(M)$ on the set of models. The corresponding *generative process* that allows us to generate data from this model is

$$M_k \sim p(M) \quad (8.39)$$

$$\theta_k | M_k \sim p(\theta_k) \quad (8.40)$$

$$\mathcal{D} | \theta_k \sim p(\mathcal{D} | \theta_k) \quad (8.41)$$

Figure 8.13 5224
 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 $p(\theta_k | M_k)$ on the corresponding model parameters, which are then used to generate the data \mathcal{D} . 5225
5226
5227



and illustrated in Figure 8.13.

Given a training set \mathcal{D} , we apply Bayes' theorem and compute the posterior distribution over models as

$$p(M_k | \mathcal{D}) \propto p(M_k)p(\mathcal{D} | M_k). \quad (8.42)$$

Note that this posterior no longer depends on the model parameters θ_k because they have been integrated out in the Bayesian setting since

$$p(\mathcal{D} | M_k) = \int p(\mathcal{D} | \theta_k)p(\theta_k | M_k)d\theta_k, \quad (8.43)$$

where $p(\theta_k | M_k)$ is the prior distribution of the model parameters θ_k of model M_k . The term (8.43) is referred to as the *model evidence* or *marginal likelihood*.

From the posterior in (8.42), we determine the MAP estimate as

$$M^* = \arg \max_{M_k} p(M_k | \mathcal{D}). \quad (8.44)$$

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 (8.43).

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 dataset \mathcal{D} . If we compute the posteriors $p(M_1 | \mathcal{D})$ and $p(M_2 | \mathcal{D})$, we can compute the ratio of the posteriors (*posterior odds*)

$$\underbrace{\frac{p(M_1 | \mathcal{D})}{p(M_2 | \mathcal{D})}}_{\text{posterior odds}} = \underbrace{\frac{\frac{p(\mathcal{D} | M_1)p(M_1)}{p(\mathcal{D})}}{\frac{p(\mathcal{D} | M_2)p(M_2)}{p(\mathcal{D})}}}_{\text{prior odds}} = \underbrace{\frac{p(M_1)}{p(M_2)}}_{\text{prior odds}} \underbrace{\frac{p(\mathcal{D} | M_1)}{p(\mathcal{D} | M_2)}}_{\text{Bayes factor}} \quad (8.45)$$

5239 The first fraction on the right-hand-side (prior odds) measures how much
 5240 our prior (initial) beliefs favor M_1 over M_2 . The ratio of the marginal
 5241 likelihoods (second fraction on the right-hand-side) is called the *Bayes*
 5242 *factor* and measures how well the data \mathcal{D} is predicted by M_1 compared to
 5243 M_2 .

Bayes factor

5244 *Remark.* The *Jeffreys-Lindley paradox* states that the “Bayes factor always
 5245 favors the simpler model since the probability of the data under a complex
 5246 model with a diffuse prior will be very small” (Murphy, 2012). Here, a
 5247 diffuse prior refers to a prior that does not favor specific models, i.e.,
 5248 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 (8.45) is 1, i.e., the posterior odds is the ratio of the marginal likelihoods (Bayes factor)

$$\frac{p(\mathcal{D} | M_1)}{p(\mathcal{D} | M_2)}. \quad (8.46)$$

5249 If the Bayes factor is greater than 1, we choose model M_1 , otherwise
 5250 model M_2 . In a similar way to frequentist statistics, there are guidelines
 5251 on the size of the ratio that one should consider before “significance” of
 5252 the result (Jeffreys, 1961).

5253 *Remark* (Computing the Marginal Likelihood). The marginal likelihood
 5254 plays an important role in model selection: We need to compute Bayes
 5255 factors (8.45) and posterior distributions over models (8.42).

5256 Unfortunately, computing the marginal likelihood requires us to solve
 5257 an integral (8.43). This integration is generally analytically intractable,
 5258 and we will have to resort to approximation techniques, e.g., numerical
 5259 integration (Stoer and Burlirsch, 2002), stochastic approximations using
 5260 Monte Carlo (Murphy, 2012) or Bayesian Monte Carlo techniques (O’Hagan,
 5261 1991; Rasmussen and Ghahramani, 2003).

5262 However, there are special cases in which we can solve it. In Section 6.6.1,
 5263 we discussed conjugate models. If we choose a conjugate parameter prior
 5264 $p(\boldsymbol{\theta})$, we can compute the marginal likelihood in closed form. In Chapter
 5265 9, we will do exactly this in the context of linear regression. ◇

5266 We have seen a brief introduction to the basic concepts of machine
 5267 learning in this chapter. For the rest of this part of the book we will see
 5268 how the three different flavours of learning in Section 8.1, Section 8.2 and
 5269 Section 8.3 are applied to the four pillars of machine learning (regression,
 5270 dimensionality reduction, density estimation and classification).

5271 Further Reading

5272 We mentioned at the start of the section that there are high level modeling
 5273 choices that influence the performance of the model. Examples include:

- 5274 • The degree of a polynomial in a regression setting

- 5275 • The number of components in a mixture model
- 5276 • The network architecture of a (deep) neural network
- 5277 • The type of kernel in a support vector machine
- 5278 • The dimensionality of the latent space in PCA
- 5279 • The learning rate (schedule) in an optimization algorithm

5280 Rasmussen and Ghahramani (2001) showed that the automatic Occam's razor does not necessarily penalize the number of parameters in
 5281 a model but it is active in terms of the complexity of functions. They also
 5282 showed that the automatic Occam's razor also holds for Bayesian non-
 5283 parametric models with many parameters, e.g., Gaussian processes.

If we focus on the maximum likelihood estimate, there exist a number of heuristics for model selection that discourage overfitting. They are called information criteria, and we choose the model with the largest value. The

Akaike Information Criterion (AIC) (Akaike, 1974)

In parametric
 5282 models, the number
 5283 of parameters is
 5284 often related to the
 complexity of the
 model class.

Akaike Information
 Criterion

$$\log p(\mathbf{x} | \boldsymbol{\theta}) - M \quad (8.47)$$

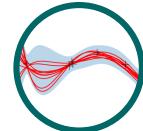
5285 corrects for the bias of the maximum likelihood estimator by addition of
 5286 a penalty term to compensate for the overfitting of more complex models
 5287 with lots of parameters. Here, M is the number of model parameters. The
 5288 AIC estimates the relative information lost by a given model.

The *Bayesian Information Criterion (BIC)* (Schwarz, 1978)

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x} | \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \approx \log p(\mathbf{x} | \boldsymbol{\theta}) - \frac{1}{2} M \log N \quad (8.48)$$

5289 can be used for exponential family distributions. Here, N is the number
 5290 of data points and M is the number of parameters. BIC penalizes model
 5291 complexity more heavily than AIC.

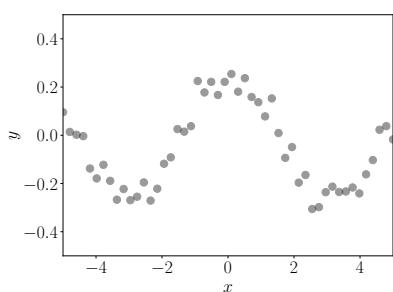
Linear Regression



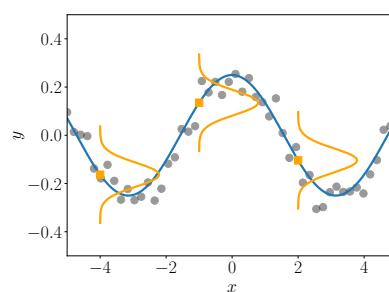
regression

5545 In the following, we will apply the mathematical concepts from Chapters 2, 5, 6 and 7 to solving linear regression (curve fitting) problems.
 5546 In *regression*, we want to find a function f that maps inputs $\mathbf{x} \in \mathbb{R}^D$ to
 5547 corresponding function values $f(\mathbf{x}) \in \mathbb{R}$ given a set of training inputs
 5548 \mathbf{x}_n and corresponding observations $y_n = f(\mathbf{x}_n) + \epsilon$, where ϵ is a random
 5549 variable that comprises measurement noise and unmodeled processes. An
 5550 illustration of such a regression problem is given in Figure 9.1. A typical
 5551 regression problem is given in Figure 9.1(a): For some input values
 5552 x we observe (noisy) function values $y = f(x) + \epsilon$. The task is to infer
 5553 the function f that generated the data. A possible solution is given in
 5554 Figure 9.1(b), where we also show three distributions centered at the
 5555 function values $f(x)$ that represent the noise in the data.

5556 Regression is a fundamental problem in machine learning, and regression
 5557 problems appear in a diverse range of research areas and applications,
 5558 including time-series analysis (e.g., system identification), control
 5559 and robotics (e.g., reinforcement learning, forward/inverse model learning),
 5560 optimization (e.g., line searches, global optimization), and deep
 5561 learning applications (e.g., computer games, speech-to-text translation,
 5562 image recognition, automatic video annotation). Regression is also a key
 5563 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.

5565 Finding a regression function requires solving a variety of problems,
 5566 including

- 5567 • **Choice of the model (type) and the parametrization** of the regres-
 5568 sion function. Given a data set, what function classes (e.g., polynomi-
 5569 als) are good candidates for modeling the data, and what particular
 5570 parametrization (e.g., degree of the polynomial) should we choose?
 5571 Model selection, as discussed in Section 8.5, allows us to compare var-
 5572 ious models to find the simplest model that explains the training data
 5573 reasonably well.
- 5574 • **Finding good parameters.** Having chosen a model of the regression
 5575 function, how do we find good model parameters? Here, we will need to
 5576 look at different loss/objective functions (they determine what a “good”
 5577 fit is) and optimization algorithms that allow us to minimize this loss.
- 5578 • **Overfitting and model selection.** Overfitting is a problem when the
 5579 regression function fits the training data “too well” but does not gen-
 5580 eralize to unseen test data. Overfitting typically occurs if the underly-
 5581 ing model (or its parametrization) is overly flexible and expressive, see
 5582 Section 8.5. We will look at the underlying reasons and discuss ways to
 5583 mitigate the effect of overfitting in the context of linear regression.
- 5584 • **Relationship between loss functions and parameter priors.** Loss func-
 5585 tions (optimization objectives) are often motivated and induced by prob-
 5586 abilistic models. We will look at the connection between loss functions
 5587 and the underlying prior assumptions that induce these losses.
- 5588 • **Uncertainty modeling.** In any practical setting, we have access to only
 5589 a finite, potentially large, amount of (training) data for selecting the
 5590 model class and the corresponding parameters. Given that this finite
 5591 amount of training data does not cover all possible scenarios, we may
 5592 want to describe the remaining parameter uncertainty to obtain a mea-
 5593 sure of confidence of the model’s prediction at test time; the smaller the
 5594 training set the more important uncertainty modeling. Consistent mod-
 5595 eling of uncertainty equips model predictions with confidence bounds.

5596 In the following, we will be using the mathematical tools from Chap-
 5597 ters 3, 5, 6 and 7 to solve linear regression problems. We will discuss
 5598 maximum likelihood and maximum a posteriori (MAP) estimation to find
 5599 optimal model parameters. Using these parameter estimates, we will have
 5600 a brief look at generalization errors and overfitting. Toward the end of
 5601 this chapter, we will discuss Bayesian linear regression, which allows us to
 5602 reason about model parameters at a higher level, thereby removing some
 5603 of the problems encountered in maximum likelihood and MAP estimation.

5604

9.1 Problem Formulation

We consider a regression problem with the likelihood function

$$p(y | \mathbf{x}) = \mathcal{N}(y | f(\mathbf{x}), \sigma^2). \quad (9.1)$$

Here, $\mathbf{x} \in \mathbb{R}^D$ are inputs and $y \in \mathbb{R}$ are noisy function values (targets). With (9.1) the functional relationship between \mathbf{x} and y is given as

$$y = f(\mathbf{x}) + \epsilon, \quad (9.2)$$

5605 where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is independent, identically distributed (i.i.d.) measurement noise. In this particular case, ϵ is Gaussian distributed with mean
 5606 0 and variance σ^2 . Our objective is to find a function that is close (similar)
 5607 to the unknown function f that generated the data.
 5608

In this chapter, we focus on parametric models, i.e., we choose a parametrized function and find parameters θ that “work well” for modeling the data. For the time being, we assume that the noise variance σ^2 is known and focus on learning the model parameters θ . In linear regression, we consider the special case that the parameters appear linearly in our model. An example of linear regression is given by

$$p(y | \mathbf{x}, \theta) = \mathcal{N}(y | \mathbf{x}^\top \theta, \sigma^2) \quad (9.3)$$

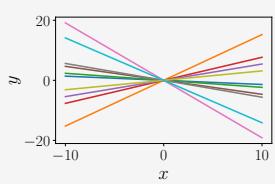
$$\iff y = \mathbf{x}^\top \theta + \epsilon, \quad (9.4)$$

5609 where $\theta \in \mathbb{R}^D$ are the parameters we seek. The class of functions described by (9.4) are straight lines that pass through the origin. In (9.4),
 5610 we chose a parametrization $f(\mathbf{x}) = \mathbf{x}^\top \theta$.
 5611

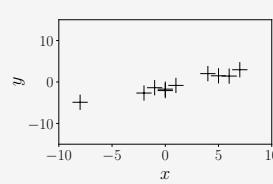
5612 The *likelihood* in (9.3) is the probability density function of y evaluated at $\mathbf{x}^\top \theta$. Note that the only source of uncertainty originates from the
 5613 observation noise (as \mathbf{x} and θ are assumed known in (9.3)). Without ob-
 5614 servation noise the relationship between \mathbf{x} and y would be deterministic
 5615 and (9.3) would be a Delta distribution.
 5616

likelihood

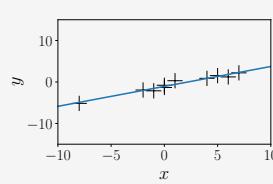
Example 9.1



(a) Example functions (straight lines) that can be described using the linear model in (9.4).



(b) Training set.



(c) Maximum likelihood estimate.

Figure 9.2 Linear regression example.
 (a) Example functions that fall into this category.
 (b) Training set.
 (c) Maximum likelihood estimate.

For $x, \theta \in \mathbb{R}$ the linear regression model in (9.4) describes straight

lines (linear functions), and the parameter θ is the slope of the line. Figure 9.3(a) shows some example functions for different values of θ .

5617 The linear regression model in (9.3)–(9.4) is not only linear in the parameters, but also linear in the inputs x . We will see later that $y = \phi^\top(x)\theta$ 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. Here, a “feature” is a representation $\phi(x)$ of the inputs x .

5618 Linear regression refers to models that are linear in the parameters.

5619

5620

5621

5622

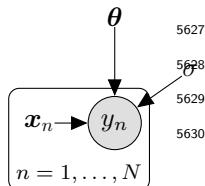
5623 In the following, we will discuss in more detail how to find good parameters θ and how to evaluate whether a parameter set “works well”.

5624

5625 For the time being we assume that the noise variance σ^2 is known.

5626

training set
Figure 9.3
 Probabilistic graphical model for linear regression.
 Observed random variables are shaded,
 deterministic/
 known values are without circles.



Consider the linear regression setting (9.4) and assume we are given a *training set* \mathcal{D} consisting of N inputs $\mathbf{x}_n \in \mathbb{R}^D$ 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 $\mathbf{x}_i, \mathbf{x}_j$ so that the likelihood factorizes according to

$$p(\mathcal{Y} | \mathcal{X}, \theta) = p(y_1, \dots, y_N | \mathbf{x}_1, \dots, \mathbf{x}_N, \theta) \quad (9.5a)$$

$$= \prod_{n=1}^N p(y_n | \mathbf{x}_n, \theta) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{x}_n^\top \theta, \sigma^2), \quad (9.5b)$$

where we defined $\mathcal{X} := \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathcal{Y} := \{y_1, \dots, y_N\}$ as the sets of training inputs and corresponding targets, respectively. The likelihood and the factors $p(y_n | \mathbf{x}_n, \theta)$ are Gaussian due to the noise distribution, see (9.3).

In the following, we will discuss how to find optimal parameters $\theta^* \in \mathbb{R}^D$ for the linear regression model (9.4). Once the parameters θ^* are found, we can predict function values by using this parameter estimate in (9.4) so that at an arbitrary test input \mathbf{x}_* we predict the probability of the corresponding target y_* as

$$p(y_* | \mathbf{x}_*, \theta^*) = \mathcal{N}(y_* | \mathbf{x}_*^\top \theta^*, \sigma^2). \quad (9.6)$$

5631 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 5632 Section 8.2.

5634

9.2.1 Maximum Likelihood Estimation

maximum likelihood estimation A widely used approach to finding the desired parameters θ_{ML} is *maximum likelihood estimation*

likelihood estimation where we find parameters $\boldsymbol{\theta}_{\text{ML}}$ that maximize the likelihood (9.5b). Intuitively, maximizing the likelihood means maximizing the predictive distribution of the training data given the model parameters. We obtain the maximum likelihood parameters as

$$\boldsymbol{\theta}_{\text{ML}} = \arg \max_{\boldsymbol{\theta}} p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}). \quad (9.7)$$

5635 *Remark.* The likelihood $p(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta})$ is not a probability distribution in $\boldsymbol{\theta}$: It
 5636 is simply a function of the parameters $\boldsymbol{\theta}$ but does not integrate to 1 (i.e.,
 5637 it is unnormalized), and may not even be integrable with respect to $\boldsymbol{\theta}$.
 5638 However, the likelihood in (9.7) is a normalized probability distribution
 5639 in \mathbf{y} . \diamond

Maximizing the likelihood means maximizing the predictive distribution of the (training) data given the parameters.

The likelihood is not a probability distribution in the parameters.

5640 To find the desired parameters $\boldsymbol{\theta}_{\text{ML}}$ that maximize the likelihood, we
 5641 typically perform gradient ascent (or gradient descent on the negative
 5642 likelihood). In the case of linear regression we consider here, however,
 5643 a closed-form solution exists, which makes iterative gradient descent un-
 5644 necessary. In practice, instead of maximizing the likelihood directly, we
 5645 apply the log-transformation to the likelihood function and minimize the
 5646 negative log-likelihood.

Since the logarithm is a (strictly) monotonically increasing function, the optimum of a function f is identical to the optimum of $\log f$.

5647 *Remark (Log Transformation).* Since the likelihood (9.5b) is a product
 5648 of N Gaussian distributions, the log-transformation is useful since a) it
 5649 does not suffer from numerical underflow, b) the differentiation rules will
 5650 turn out simpler. More specifically, numerical underflow will be a prob-
 5651 lem when we multiply N probabilities, where N is the number of data
 5652 points, since we cannot represent very small numbers, such as 10^{-256} .
 5653 Furthermore, the log-transform will turn the product into a sum of log-
 5654 probabilities such that the corresponding gradient is a sum of individual
 5655 gradients, instead of a repeated application of the product rule (5.46) to
 5656 compute the gradient of a product of N terms. \diamond

To find the optimal parameters $\boldsymbol{\theta}_{\text{ML}}$ of our linear regression problem, we minimize the negative log-likelihood

$$-\log p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) = -\log \prod_{n=1}^N p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) = -\sum_{n=1}^N \log p(y_n | \mathbf{x}_n, \boldsymbol{\theta}), \quad (9.8)$$

5657 where we exploited that the likelihood (9.5b) factorizes over the number
 5658 of data points due to our independence assumption on the training set.

In the linear regression model (9.4) the likelihood is Gaussian (due to the Gaussian additive noise term), such that we arrive at

$$\log p(y_n | \mathbf{x}_n, \boldsymbol{\theta}) = -\frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \boldsymbol{\theta})^2 + \text{const} \quad (9.9)$$

where the constant includes all terms independent of $\boldsymbol{\theta}$. Using (9.9) in the

negative log-likelihood (9.8) we obtain (ignoring the constant terms)

$$\mathcal{L}(\boldsymbol{\theta}) := -\log p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{x}_n^\top \boldsymbol{\theta})^2 \quad (9.10a)$$

$$= \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2, \quad (9.10b)$$

The negative log-likelihood function is also called *error function*.
design matrix
where we define the *design matrix* $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$ as the collection of training inputs and $\mathbf{y} := [y_1, \dots, y_N]^\top \in \mathbb{R}^N$ as a vector that collects all training targets. Note that the n th row in the design matrix \mathbf{X} corresponds to the training input \mathbf{x}_n .

In (9.10b) we used the fact that the sum of squared errors between the observations y_n and the corresponding model prediction $\mathbf{x}_n^\top \boldsymbol{\theta}$ equals the squared distance between \mathbf{y} and $\mathbf{X}\boldsymbol{\theta}$. Remember from Section 3.1 that $\|\mathbf{x}\|^2 = \mathbf{x}^\top \mathbf{x}$ if we choose the dot product as the inner product.

With (9.10b) we have now a concrete form of the negative log-likelihood function we need to optimize. We immediately see that (9.10b) 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

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{d}{d\boldsymbol{\theta}} \left(\frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \right) \quad (9.11a)$$

$$= \frac{1}{2\sigma^2} \frac{d}{d\boldsymbol{\theta}} (\mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta}) \quad (9.11b)$$

$$= \frac{1}{\sigma^2} (-\mathbf{y}^\top \mathbf{X} + \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X}) \in \mathbb{R}^{1 \times D}. \quad (9.11c)$$

The maximum likelihood estimator $\boldsymbol{\theta}_{\text{ML}}$ solves $\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \mathbf{0}^\top$ (necessary optimality condition) and we obtain

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \mathbf{0}^\top \stackrel{(9.11c)}{\iff} \boldsymbol{\theta}_{\text{ML}}^\top \mathbf{X}^\top \mathbf{X} = \mathbf{y}^\top \mathbf{X} \quad (9.12a)$$

$$\iff \boldsymbol{\theta}_{\text{ML}}^\top = \mathbf{y}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \quad (9.12b)$$

$$\iff \boldsymbol{\theta}_{\text{ML}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \quad (9.12c)$$

Ignoring the possibility of duplicate data points, $\text{rk}(\mathbf{X}) = D$ if $N \geq D$, i.e., we do not have more parameters than data points.

We could right-multiply the first equation by $(\mathbf{X}^\top \mathbf{X})^{-1}$ because $\mathbf{X}^\top \mathbf{X}$ is positive definite if $\text{rk}(\mathbf{X}) = D$, where $\text{rk}(\mathbf{X})$ denotes the rank of \mathbf{X} .

Remark. Setting the gradient to $\mathbf{0}^\top$ is a necessary and sufficient condition and we obtain a global minimum since the Hessian $\nabla_{\boldsymbol{\theta}}^2 \mathcal{L}(\boldsymbol{\theta}) = \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{D \times D}$ is positive definite. \diamond

Example 9.2 (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 9.3(a). For the data set shown in Figure 9.3(b) we find the maximum likelihood estimate of the slope parameter θ using (9.12c) and obtain the maximum likelihood linear function in Figure 9.3(c).

Maximum Likelihood Estimation with Features

So far, we considered the linear regression setting described in (9.4), which allowed us to fit straight lines to data using maximum likelihood estimation. However, straight lines are not sufficiently 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 $\phi(\mathbf{x})$ of the inputs \mathbf{x} and then linearly combine the components of this transformation. The corresponding linear regression model is

$$\begin{aligned} p(y | \mathbf{x}, \boldsymbol{\theta}) &= \mathcal{N}(y | \phi^\top(\mathbf{x})\boldsymbol{\theta}, \sigma^2) \\ \iff y &= \phi^\top(\mathbf{x})\boldsymbol{\theta} + \epsilon = \sum_{k=0}^{K-1} \theta_k \phi_k(\mathbf{x}) + \epsilon, \end{aligned} \quad (9.13)$$

where $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^K$ is a (nonlinear) transformation of the inputs \mathbf{x} and $\phi_k : \mathbb{R}^D \rightarrow \mathbb{R}$ is the k th component of the *feature vector* ϕ . Note that the model parameters $\boldsymbol{\theta}$ still appear only linearly.

Linear regression refers to “linear-in-the-parameters” regression models, but the inputs can undergo any nonlinear transformation.

feature vector

Example 9.3 (Polynomial Regression)

We are concerned with a regression problem $y = \phi^\top(\mathbf{x})\boldsymbol{\theta} + \epsilon$, where $x \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathbb{R}^K$. A transformation that is often used in this context is

$$\phi(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^{K-1} \end{bmatrix} \in \mathbb{R}^K. \quad (9.14)$$

This means, we “lift” the original one-dimensional input space into a K -dimensional feature space consisting of all monomials x^k for $k = 0, \dots, K - 1$. With these features, we can model polynomials of degree $\leq K - 1$ within the framework of linear regression: A polynomial of de-

gree $K - 1$ is

$$f(x) = \sum_{k=0}^{K-1} \theta_k x^k = \phi^\top(x) \boldsymbol{\theta} \quad (9.15)$$

where ϕ is defined in (9.14) and $\boldsymbol{\theta} = [\theta_0, \dots, \theta_{K-1}]^\top \in \mathbb{R}^K$ contains the (linear) parameters θ_k .

feature matrix
design matrix

Let us now have a look at maximum likelihood estimation of the parameters $\boldsymbol{\theta}$ in the linear regression model (9.13). We consider training inputs $\mathbf{x}_n \in \mathbb{R}^D$ and targets $y_n \in \mathbb{R}$, $n = 1, \dots, N$, and define the *feature matrix* (*design matrix*) as

$$\Phi := \begin{bmatrix} \phi^\top(\mathbf{x}_1) \\ \vdots \\ \phi^\top(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \cdots & \phi_{K-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \cdots & \phi_{K-1}(\mathbf{x}_2) \\ \vdots & & \vdots \\ \phi_0(\mathbf{x}_N) & \cdots & \phi_{K-1}(\mathbf{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times K}, \quad (9.16)$$

5681 where $\Phi_{ij} = \phi_j(\mathbf{x}_i)$ and $\phi_j : \mathbb{R}^D \rightarrow \mathbb{R}$.

Example 9.4 (Feature Matrix for Second-order Polynomials)

For a second-order polynomial and N training points $x_n \in \mathbb{R}$, $n = 1, \dots, N$, the feature matrix is

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}. \quad (9.17)$$

maximum likelihood
estimate

With the feature matrix Φ defined in (9.16) the negative log-likelihood for the linear regression model (9.13) can be written as

$$-\log p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2} (\mathbf{y} - \Phi \boldsymbol{\theta})^\top (\mathbf{y} - \Phi \boldsymbol{\theta}) + \text{const.} \quad (9.18)$$

Comparing (9.18) with the negative log-likelihood in (9.10b) for the “feature-free” model, we immediately see we just need to replace \mathbf{X} with Φ . Since both \mathbf{X} and Φ are independent of the parameters $\boldsymbol{\theta}$ that we wish to optimize, we arrive immediately at the *maximum likelihood estimate*

$$\boldsymbol{\theta}_{\text{ML}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y} \quad (9.19)$$

5682 for the linear regression problem with nonlinear features defined in (9.13).

5683 *Remark.* When we were working without features, we required $\mathbf{X}^\top \mathbf{X}$ to

be invertible, which is the case when the rows of \mathbf{X} are linearly independent. In (9.19), we therefore require $\Phi^\top \Phi \in \mathbb{R}^{D \times D}$ to be invertible. This is the case if and only if $\text{rk}(\Phi) = D$. \diamond

Example 9.5 (Maximum Likelihood Polynomial Fit)

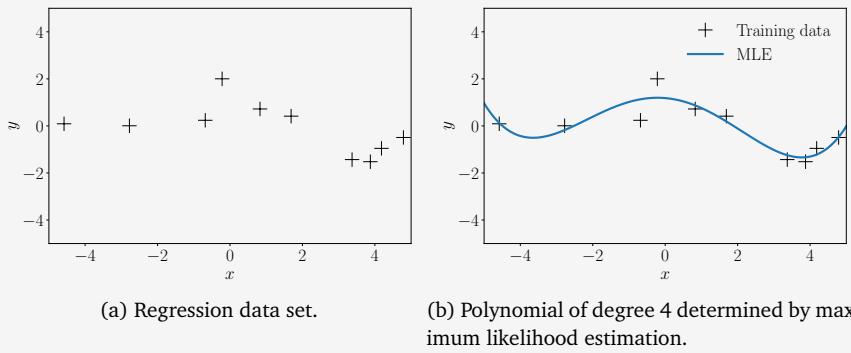


Figure 9.4
Polynomial regression. (a) Data set consisting of (x_n, y_n) pairs, $n = 1, \dots, 10$; (b) Maximum likelihood polynomial of degree 4.

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 (9.19). The maximum likelihood estimate yields function values $\phi^\top(x_*)\boldsymbol{\theta}_{\text{ML}}$ 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 the maximum likelihood estimator σ_{ML}^2 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. The log-likelihood is given by

$$\log p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}, \sigma^2) = \sum_{n=1}^N \log \mathcal{N}(y_n | \phi^\top(\mathbf{x}_n)\boldsymbol{\theta}, \sigma^2) \quad (9.20a)$$

$$= \sum_{n=1}^N \left(-\frac{1}{2} \log(2\pi) - \frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (y_n - \phi^\top(\mathbf{x}_n)\boldsymbol{\theta})^2 \right) \quad (9.20b)$$

$$= -\frac{N}{2} \log \sigma^2 - \underbrace{\frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \phi^\top(\mathbf{x}_n)\boldsymbol{\theta})^2}_{=:s} + \text{const.} \quad (9.20c)$$

The partial derivative of the log-likelihood with respect to σ^2 is then

$$\frac{\partial \log p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}, \sigma^2)}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}s = 0 \quad (9.21a)$$

$$\iff \frac{N}{2\sigma^2} = \frac{s}{2\sigma^4} \quad (9.21b)$$

so that we identify

$$\sigma_{\text{ML}}^2 = \frac{s}{N} = \frac{1}{N} \sum_{n=1}^N (y_n - \boldsymbol{\phi}^\top(\mathbf{x}_n)\boldsymbol{\theta})^2. \quad (9.22)$$

Therefore, the maximum likelihood estimate of the noise variance is the empirical mean of the squared distances between the noise-free function values $\boldsymbol{\phi}^\top(\mathbf{x}_n)\boldsymbol{\theta}$ and the corresponding noisy observations y_n at input locations \mathbf{x}_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 (9.10b), which we minimized to determine the maximum likelihood estimator. Alternatively, given that the noise parameter σ^2 is not a free model parameter, we can ignore the scaling by $1/\sigma^2$, so that we end up with a squared-error-loss function $\|\mathbf{y} - \Phi\boldsymbol{\theta}\|^2$. Instead of using this squared loss, we often use the *root mean squared error (RMSE)*

$$\sqrt{\|\mathbf{y} - \Phi\boldsymbol{\theta}\|^2 / N} = \sqrt{\frac{1}{N} \sum_{n=1}^N (y_n - \boldsymbol{\phi}^\top(\mathbf{x}_n)\boldsymbol{\theta})^2}, \quad (9.23)$$

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, if we fit a model that maps post-codes (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 EUR². If we choose to include the factor σ^2 from the original negative log-likelihood (9.10b) then we end up with a unitless objective, i.e., in the above example our objective would no longer be in EUR or EUR².

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 $0 \leq M \leq N - 1$. For $M \leq N$, the maximum likelihood estimator is unique. For $M > N$ we have more parameters

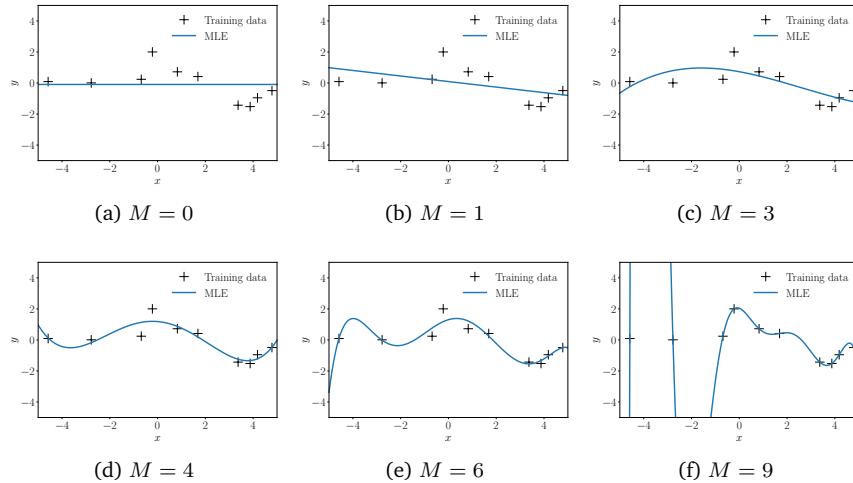


Figure 9.5
Maximum likelihood fits for different polynomial degrees M .

than data points, and would need to solve an underdetermined system of linear equations ($\Phi^\top \Phi$ in (9.19) would also no longer be invertible) so that there are infinitely many possible maximum likelihood estimators.

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 $M = 3, \dots, 5$ 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 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*.

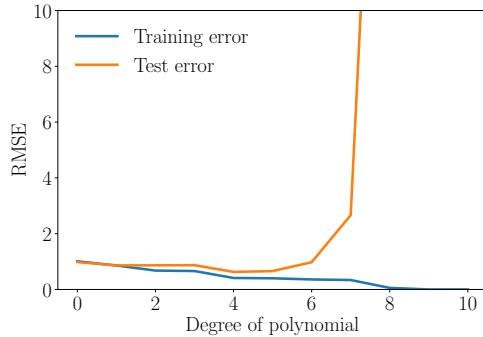
Remember that the goal is to achieve good generalization by making accurate predictions for new (unseen) data. We obtain some quantitative insight into the dependence of the generalization performance on the polynomial of degree M by considering a separate test set comprising 200 data points generated 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 (9.23) for both the training data and the test data.

Looking now at the test error, which is a qualitative 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

overfitting

Note that the noise variance $\sigma^2 > 0$.

Figure 9.6 Training and test error.



training error 5737 maximum likelihood fits in Figure 9.5. Note that the *training error* (blue
 5738 curve in Figure 9.6) never increases when the degree of the polynomial in-
 5739 creases. In our example, the best generalization (the point of the smallest
 5740 *test error*) is obtained for a polynomial of degree $M = 4$.

9.2.3 Maximum *A Posteriori* Estimation

5741 We just saw that maximum likelihood estimation is prone to overfitting.
 5742 We often observe that the magnitude of the parameter values becomes
 5743 relatively large if we run into overfitting (Bishop, 2006).
 5744 To mitigate the effect of huge parameter values, we can place a prior
 5745 distribution $p(\boldsymbol{\theta})$ on the parameters. The prior distribution explicitly en-
 5746 codes what parameter values are plausible (before having seen any data).
 5747 For example, a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ on a single parameter
 5748 θ encodes that parameter values are expected lie in the interval 0 ± 2
 5749 (two standard deviations around the mean value). Once a dataset \mathcal{X}, \mathcal{Y}
 5750 is available, instead of maximizing the likelihood we seek parameters that
 5751 maximize the posterior distribution $p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y})$. This procedure is called
 5752 *maximum a posteriori* (MAP) estimation.
 5753

The posterior over the parameters $\boldsymbol{\theta}$, given the training data \mathcal{X}, \mathcal{Y} , is obtained by applying Bayes' theorem (Section 6.3) as

$$p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{Y} | \mathcal{X})}. \quad (9.24)$$

5754 Since the posterior explicitly depends on the parameter prior $p(\boldsymbol{\theta})$, the
 5755 prior will have an effect on parameter vector we find as the maximizer
 5756 of the posterior. We will see this more explicitly in the following. The pa-
 5757 rameter vector $\boldsymbol{\theta}_{\text{MAP}}$ that maximizes the posterior (9.24) is the maximum
 5758 a posteriori (MAP) estimate.

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

$$\log p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}) = \log p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + \text{const}, \quad (9.25)$$

where the constant comprises the terms that are independent of θ . We see that the log-posterior in (9.25) is the sum of the log-likelihood $p(\mathcal{Y} | \mathcal{X}, \theta)$ and the log-prior $\log p(\theta)$ so that the MAP estimate will be a “compromise” between the prior (our suggestion for plausible parameter values before observing data) and the data-dependent likelihood.

To find the MAP estimate θ_{MAP} , we minimize the negative log-posterior distribution with respect to θ , i.e., we solve

$$\theta_{\text{MAP}} \in \arg \min_{\theta} \{-\log p(\mathcal{Y} | \mathcal{X}, \theta) - \log p(\theta)\}. \quad (9.26)$$

The gradient of the negative log-posterior with respect to θ is

$$-\frac{d \log p(\theta | \mathcal{X}, \mathcal{Y})}{d \theta} = -\frac{d \log p(\mathcal{Y} | \mathcal{X}, \theta)}{d \theta} - \frac{d \log p(\theta)}{d \theta}, \quad (9.27)$$

where we identify the first term on the right-hand-side as the gradient of the negative log-likelihood from (9.11c).

With a (conjugate) Gaussian prior $p(\theta) = \mathcal{N}(\mathbf{0}, b^2 \mathbf{I})$ on the parameters θ , the negative log-posterior for the linear regression setting (9.13), we obtain the negative log posterior

$$-\log p(\theta | \mathcal{X}, \mathcal{Y}) = \frac{1}{2\sigma^2} (\mathbf{y} - \Phi \theta)^T (\mathbf{y} - \Phi \theta) + \frac{1}{2b^2} \theta^T \theta + \text{const.} \quad (9.28)$$

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 θ is then

$$-\frac{d \log p(\theta | \mathcal{X}, \mathcal{Y})}{d \theta} = \frac{1}{\sigma^2} (\theta^T \Phi^T \Phi - \mathbf{y}^T \Phi) + \frac{1}{b^2} \theta^T. \quad (9.29)$$

We will find the MAP estimate θ_{MAP} by setting this gradient to $\mathbf{0}^T$ and solving for θ_{MAP} . We obtain

$$\frac{1}{\sigma^2} (\theta^T \Phi^T \Phi - \mathbf{y}^T \Phi) + \frac{1}{b^2} \theta^T = \mathbf{0}^T \quad (9.30a)$$

$$\iff \theta^T \left(\frac{1}{\sigma^2} \Phi^T \Phi + \frac{1}{b^2} \mathbf{I} \right) - \frac{1}{\sigma^2} \mathbf{y}^T \Phi = \mathbf{0}^T \quad (9.30b)$$

$$\iff \theta^T \left(\Phi^T \Phi + \frac{\sigma^2}{b^2} \mathbf{I} \right) = \mathbf{y}^T \Phi \quad (9.30c)$$

$$\iff \theta^T = \mathbf{y}^T \Phi \left(\Phi^T \Phi + \frac{\sigma^2}{b^2} \mathbf{I} \right)^{-1} \quad (9.30d)$$

$\Phi^T \Phi$ is symmetric, positive semidefinite and the additional term is strictly positive definite so that all eigenvalues of the matrix to be inverted are positive.

so that the MAP estimate is (by transposing both sides of the last equality)

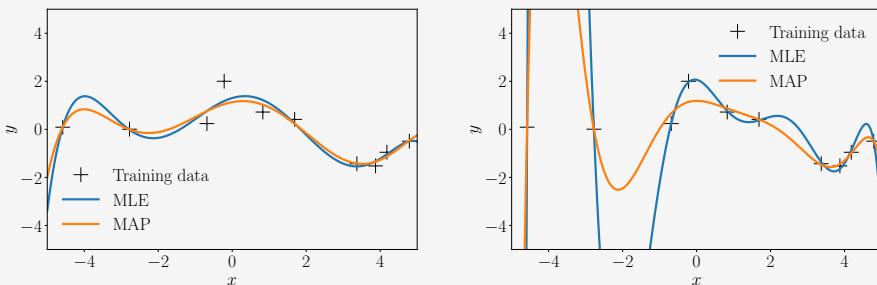
$$\theta_{\text{MAP}} = \left(\Phi^T \Phi + \frac{\sigma^2}{b^2} \mathbf{I} \right)^{-1} \Phi^T \mathbf{y}. \quad (9.31)$$

Comparing the MAP estimate in (9.31) with the maximum likelihood estimate in (9.19) we see that the only difference between both solutions is the additional term $\frac{\sigma^2}{b^2} \mathbf{I}$ in the inverse matrix. This term ensures that

5769 $\Phi^\top \Phi + \frac{\sigma^2}{b^2} I$ is symmetric and strictly positive definite (i.e., its inverse
5770 exists and the MAP estimate is the unique solution of a system of linear
5771 equations) and plays the role of the *regularizer*.

Example 9.6 (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 $p(\theta) = \mathcal{N}(0, I)$ on the parameters θ 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. Although the MAP estimate can push the boundaries of overfitting it is not a general solution to this problem so that we need a more principled approach to tackle overfitting.

9.2.4 MAP Estimation as Regularization

Instead of placing a prior distribution on the parameters θ it is also possible to mitigate the effect of overfitting by penalizing the amplitude of the parameter by means of *regularization*. In *regularized least squares*, we consider the loss function

$$\|\mathbf{y} - \Phi\theta\|^2 + \lambda \|\theta\|_2^2, \quad (9.32)$$

5772 which we minimize with respect to θ (see Section 8.1.3). Here, the first term is a *data-fit term* (also called *misfit term*), which is proportional to the negative log-likelihood, see (9.10b). The second term is called the *regularizer*, and the *regularization parameter* $\lambda \geq 0$ controls the “strictness” of the regularization.

5773 5774 5775 5776 5777 5778 5779 5780 5781 *Remark.* Instead of the Euclidean norm $\|\cdot\|_2$, we can choose any p -norm $\|\cdot\|_p$ in (9.32). In practice, smaller values for p lead to sparser solutions. Here, “sparse” means that many parameter values $\theta_d = 0$, which is also useful for variable selection. For $p = 1$, the regularizer is called *LASSO*

5782 (least absolute shrinkage and selection operator) and was proposed by Tib-
 5783 shirani (1996). \diamond

The regularizer $\lambda \|\boldsymbol{\theta}\|_2^2$ in (9.32) can be interpreted as a negative log-Gaussian prior, which we use in MAP estimation, see (9.26). More specifically, with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2 \mathbf{I})$, we obtain the negative log-Gaussian prior

$$-\log p(\boldsymbol{\theta}) = \frac{1}{2b^2} \|\boldsymbol{\theta}\|_2^2 + \text{const} \quad (9.33)$$

5784 so that for $\lambda = \frac{1}{2b^2}$ the regularization term and the negative log-Gaussian
 5785 prior are identical.

Given that the regularized least-squares loss function in (9.32) consists of terms that are closely related to the negative log-likelihood plus a negative log-prior, it is not surprising that, when we minimize this loss, we obtain a solution that closely resembles the MAP estimate in (9.31). More specifically, minimizing the regularized least-squares loss function yields

$$\boldsymbol{\theta}_{\text{RLS}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I})^{-1} \boldsymbol{\Phi}^\top \mathbf{y}, \quad (9.34)$$

5786 which is identical to the MAP estimate in (9.31) for $\lambda = \frac{\sigma^2}{b^2}$, where σ^2 is
 5787 the noise variance and b^2 the variance of the (isotropic) Gaussian prior
 5788 $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, b^2 \mathbf{I})$.

5789 So far, we covered parameter estimation using maximum likelihood and
 5790 MAP estimation where we found point estimates $\boldsymbol{\theta}^*$ that optimize an ob-
 5791 jective function (likelihood or posterior). We saw that both maximum like-
 5792 lihood and MAP estimation can lead to overfitting. In the next section, we
 5793 will discuss Bayesian linear regression, where we use Bayesian inference
 5794 (Section 8.3) to find a posterior distribution over the unknown parame-
 5795 ters, which we subsequently use to make predictions. More specifically, for
 5796 predictions we will average over all plausible sets of parameters instead
 5797 of focusing on a point estimate.

5798 9.3 Bayesian Linear Regression

5799 Previously, we looked at linear regression models where we estimated the
 5800 model parameters $\boldsymbol{\theta}$, e.g., by means of maximum likelihood or MAP esti-
 5801 mation. We discovered that MLE can lead to severe overfitting, in particu-
 5802 lar, in the small-data regime. MAP addresses this issue by placing a prior
 5803 on the parameters that plays the role of a regularizer.

Bayesian linear
regression

5804 Bayesian linear regression pushes the idea of the parameter prior a step
 5805 further and does not even attempt to compute a point estimate of the pa-
 5806 rameters, but instead the full posterior over the parameters is taken into
 5807 account when making predictions. This means we do not fit any pa-
 5808 rameters, but we compute an average over all plausible parameters settings
 5809 (according to the posterior).

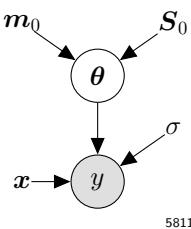
5810

9.3.1 Model

In Bayesian linear regression, we consider the model

$$\begin{aligned} \text{prior} \quad p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0), \\ \text{likelihood} \quad p(y | \mathbf{x}, \boldsymbol{\theta}) &= \mathcal{N}(y | \boldsymbol{\phi}^\top(\mathbf{x})\boldsymbol{\theta}, \sigma^2), \end{aligned} \quad (9.35)$$

Figure 9.8
Graphical model for
Bayesian linear
regression.



5811

where we now explicitly place a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0)$ on $\boldsymbol{\theta}$, which turns the parameter vector into a random variable. This 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. The full probabilistic model, i.e., the joint distribution of observed and unobserved random variables, y and $\boldsymbol{\theta}$, respectively, is

$$p(y, \boldsymbol{\theta} | \mathbf{x}) = p(y | \mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}). \quad (9.36)$$

9.3.2 Prior Predictions

In practice, we are usually not so much interested in the parameter values $\boldsymbol{\theta}$ themselves. 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 \mathbf{x}_* , we integrate out $\boldsymbol{\theta}$ and obtain

$$p(y_* | \mathbf{x}_*) = \int p(y_* | \mathbf{x}_*, \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta}}[p(y_* | \mathbf{x}_*, \boldsymbol{\theta})], \quad (9.37)$$

5812
5813
5814
5815

which we can interpret as the average prediction of $y_* | \mathbf{x}_*, \boldsymbol{\theta}$ 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 \mathbf{x}_* , but no training data.

In our model (9.35), 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 $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0)$, we obtain the predictive distribution as

$$p(y_* | \mathbf{x}_*) = \mathcal{N}(\boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{m}_0, \boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{S}_0\boldsymbol{\phi}(\mathbf{x}_*) + \sigma^2), \quad (9.38)$$

where we exploited that (i) the prediction is Gaussian due to conjugacy (see Section 6.6) and the marginalization property of Gaussians (see Section 6.5), (ii) the Gaussian noise is independent so that

$$\mathbb{V}[y_*] = \mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\theta}] + \mathbb{V}_{\epsilon}[\epsilon], \quad (9.39)$$

5816
5817
5818
5819

(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 (6.51) and (6.52), respectively. In (9.38), the term $\boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{S}_0\boldsymbol{\phi}(\mathbf{x}_*)$ in the predictive variance explicitly accounts for the uncertainty associated

5820 with the parameters θ , whereas σ^2 is the uncertainty contribution due to
 5821 the measurement noise.

If we are interested in predicting noise-free function values $f(\mathbf{x}_*) = \phi^\top(\mathbf{x}_*)\theta$ instead of the noise-corrupted targets y_* we obtain

$$p(f(\mathbf{x}_*)) = \mathcal{N}(\phi^\top(\mathbf{x}_*)\mathbf{m}_0, \phi^\top(\mathbf{x}_*)\mathbf{S}_0\phi(\mathbf{x}_*)) , \quad (9.40)$$

5822 which only differs from (9.38) in the omission of the noise variance σ^2 in
 5823 the predictive variance.

5824 *Remark* (Distribution over Functions). Since we can represent the distri-
 5825 bution $p(\theta)$ using a set of samples θ_i and every sample θ_i gives rise to
 5826 a function $f_i(\cdot) = \theta_i^\top \phi(\cdot)$ it follows that the parameter distribution $p(\theta)$
 5827 induces a distribution $p(f(\cdot))$ over functions. Here we use the notation (\cdot)
 5828 to explicitly denote a functional relationship. ◇

The parameter distribution $p(\theta)$ induces a distribution over functions.

Example 9.7 (Prior over Functions)

Let us consider a Bayesian linear regression problem with polynomials of degree 5. We choose a parameter prior $p(\theta) = \mathcal{N}(\mathbf{0}, \frac{1}{4}\mathbf{I})$. Figure 9.9 visualizes the induced prior distribution over functions (shaded area: dark-gray: 67% confidence bound, light-gray: 95% confidence bound) induced by this parameter prior, including some function samples from this prior.

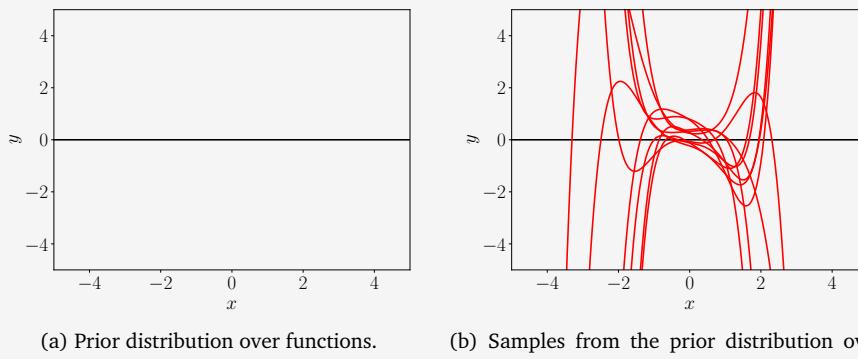


Figure 9.9 Prior over functions.
 (a) Distribution over functions represented by the mean function (black line) and the marginal uncertainties (shaded), representing the 67% and 95% confidence bounds, respectively;
 (b) Samples from the prior over functions, which are induced by the samples from the parameter prior.

A function sample is obtained by first sampling a parameter vector $\theta_i \sim p(\theta)$ and then computing $f_i(\cdot) = \theta_i^\top \phi(\cdot)$. We used 200 input locations $x_* \in [-5, 5]$ to which we apply the feature function $\phi(\cdot)$. The uncertainty (represented by the shaded area) in Figure 9.9 is solely due to the parameter uncertainty because we considered the noise-free predictive distribution (9.40).

5829 So far, we looked at computing predictions using the parameter prior
 5830 $p(\theta)$. However, when we have a parameter posterior (given some train-
 5831 ing data \mathbf{X}, \mathbf{y}), the same principles for prediction and inference hold
 5832 as in (9.37) – we just need to replace the prior $p(\theta)$ with the posterior

5833 $p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y})$. In the following, we will derive the posterior distribution in
5834 detail before using it to make predictions.

5835

9.3.3 Posterior Distribution

Given a training set of inputs $\mathbf{x}_n \in \mathbb{R}^D$ and corresponding observations $y_n \in \mathbb{R}$, $n = 1, \dots, N$, we compute the posterior over the parameters using Bayes' theorem as

$$p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{Y} | \mathcal{X})}, \quad (9.41)$$

where \mathcal{X} is the set of training inputs and \mathcal{Y} the collection of corresponding training targets. Furthermore, $p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ the parameter prior and

$$p(\mathcal{Y} | \mathcal{X}) = \int p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta}}[p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})] \quad (9.42)$$

marginal likelihood/evidence
evidence
The marginal likelihood is the expected likelihood under the parameter prior.
5836
5837
5838
5839

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 $p(\boldsymbol{\theta})$).

Theorem 9.1 (Parameter Posterior). *In our model (9.35), the parameter posterior (9.41) can be computed in closed form as*

$$p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}) = \mathcal{N}(\boldsymbol{\theta} | \mathbf{m}_N, \mathbf{S}_N), \quad (9.43a)$$

$$\mathbf{S}_N = (\mathbf{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}, \quad (9.43b)$$

$$\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\mathbf{y}), \quad (9.43c)$$

5840 where the subscript N indicates the size of the training set.

Proof Bayes' theorem tells us that the posterior $p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y})$ is proportional to the product of the likelihood $p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})$ and the prior $p(\boldsymbol{\theta})$:

$$\text{posterior} \quad p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{Y} | \mathcal{X})} \quad (9.44a)$$

$$\text{likelihood} \quad p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y} | \boldsymbol{\Phi}\boldsymbol{\theta}, \sigma^2\mathbf{I}) \quad (9.44b)$$

$$\text{prior} \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} | \mathbf{m}_0, \mathbf{S}_0). \quad (9.44c)$$

5841 Instead of looking at the product of the prior and the likelihood, we
5842 can transform the problem into log-space and solve for the mean and
5843 covariance of the posterior by completing the squares.

The sum of the log-prior and the log-likelihood is

$$\log \mathcal{N}(\mathbf{y} | \boldsymbol{\Phi}\boldsymbol{\theta}, \sigma^2\mathbf{I}) + \log \mathcal{N}(\boldsymbol{\theta} | \mathbf{m}_0, \mathbf{S}_0) \quad (9.45a)$$

$$= -\frac{1}{2}(\sigma^{-2}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + (\boldsymbol{\theta} - \mathbf{m}_0)^\top\mathbf{S}_0^{-1}(\boldsymbol{\theta} - \mathbf{m}_0)) + \text{const} \quad (9.45b)$$

where the constant contains terms independent of θ . We will ignore the constant in the following. We now factorize (9.45b), which yields

$$\begin{aligned} & -\frac{1}{2}(\sigma^{-2}\mathbf{y}^\top \mathbf{y} - 2\sigma^{-2}\mathbf{y}^\top \Phi\theta + \theta^\top \sigma^{-2}\Phi^\top \Phi\theta + \theta^\top S_0^{-1}\theta \\ & - 2m_0^\top S_0^{-1}\theta + m_0^\top S_0^{-1}m_0) \end{aligned} \quad (9.46a)$$

$$= -\frac{1}{2}(\theta^\top (\sigma^{-2}\Phi^\top \Phi + S_0^{-1})\theta - 2(\sigma^{-2}\Phi^\top \mathbf{y} + S_0^{-1}m_0)^\top \theta) + \text{const}, \quad (9.46b)$$

where the constant contains the black terms in (9.46a), which are independent of θ . The orange terms are terms that are linear in θ , and the blue terms are the ones that are quadratic in θ . Inspecting (9.46b), we find that this equation is quadratic in θ . The fact that the unnormalized log-posterior distribution is a (negative) quadratic form implies that the posterior is Gaussian, i.e.,

$$p(\theta | \mathcal{X}, \mathcal{Y}) = \exp(\log p(\theta | \mathcal{X}, \mathcal{Y})) \propto \exp(\log p(\mathcal{Y} | \mathcal{X}, \theta) + \log p(\theta)) \quad (9.47a)$$

$$\propto \exp\left(-\frac{1}{2}(\theta^\top (\sigma^{-2}\Phi^\top \Phi + S_0^{-1})\theta - 2(\sigma^{-2}\Phi^\top \mathbf{y} + S_0^{-1}m_0)^\top \theta)\right), \quad (9.47b)$$

5844 where we used (9.46b) in the last expression.

The remaining task is it to bring this (unnormalized) Gaussian into the form that is proportional to $\mathcal{N}(\theta | m_N, S_N)$, i.e., we need to identify the mean m_N and the covariance matrix S_N . To do this, we use the concept of *completing the squares*. The desired log-posterior is

$$\log \mathcal{N}(\theta | m_N, S_N) = -\frac{1}{2}(\theta - m_N)^\top S_N^{-1}(\theta - m_N) + \text{const} \quad (9.48a)$$

$$= -\frac{1}{2}(\theta^\top S_N^{-1}\theta - 2m_N^\top S_N^{-1}\theta + m_N^\top S_N^{-1}m_N). \quad (9.48b)$$

completing the squares

Here, we factorized the quadratic form $(\theta - m_N)^\top S_N^{-1}(\theta - m_N)$ into a term that is quadratic in θ alone (blue), a term that is linear in θ (orange), and a constant term (black). This allows us now to find S_N and m_N by matching the colored expressions in (9.46b) and (9.48b), which yields

$$S_N^{-1} = \Phi^\top \sigma^{-2} I \Phi + S_0^{-1} \iff S_N = (\sigma^{-2}\Phi^\top \Phi + S_0^{-1})^{-1}, \quad (9.49)$$

$$m_N^\top S_N^{-1} = (\sigma^{-2}\Phi^\top \mathbf{y} + S_0^{-1}m_0)^\top \iff m_N = S_N(\sigma^{-2}\Phi^\top \mathbf{y} + S_0^{-1}m_0). \quad (9.50)$$

□

5845

Remark (General Approach to Completing the Squares). If we are given an equation

$$\mathbf{x}^\top A\mathbf{x} - 2\mathbf{a}^\top \mathbf{x} + \text{const}_1, \quad (9.51)$$

where \mathbf{A} is symmetric and positive definite, which we wish to bring into the form

$$(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma (\mathbf{x} - \boldsymbol{\mu}) + \text{const}_2, \quad (9.52)$$

we can do this by setting

$$\Sigma := \mathbf{A}, \quad (9.53)$$

$$\boldsymbol{\mu} := \Sigma^{-1} \mathbf{a} \quad (9.54)$$

5846 and $\text{const}_2 = \text{const}_1 - \boldsymbol{\mu}^\top \Sigma \boldsymbol{\mu}$. \diamond

We can see that the terms inside the exponential in (9.47b) are of the form (9.51) with

$$\mathbf{A} := \sigma^{-2} \Phi^\top \Phi + \mathbf{S}_0^{-1}, \quad (9.55)$$

$$\mathbf{a} := \sigma^{-2} \Phi^\top \mathbf{y} + \mathbf{S}_0^{-1} \mathbf{m}_0. \quad (9.56)$$

5847 Since \mathbf{A} , \mathbf{a} can be difficult to identify in equations like (9.46a), it is often helpful to bring these equations into the form (9.51) that decouples quadratic term, linear terms and constants, which simplifies finding the desired solution.

5851 9.3.4 Posterior Predictions

In (9.37), we computed the predictive distribution of y_* at a test input \mathbf{x}_* using the parameter prior $p(\boldsymbol{\theta})$. In principle, predicting with the parameter posterior $p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y})$ 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

$$p(y_* | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int p(y_* | \mathbf{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}) d\boldsymbol{\theta} \quad (9.57a)$$

$$= \int \mathcal{N}(y_* | \boldsymbol{\phi}^\top(\mathbf{x}_*) \boldsymbol{\theta}, \sigma^2) \mathcal{N}(\boldsymbol{\theta} | \mathbf{m}_N, \mathbf{S}_N) d\boldsymbol{\theta} \quad (9.57b)$$

$$= \mathcal{N}(y_* | \boldsymbol{\phi}^\top(\mathbf{x}_*) \mathbf{m}_N, \boldsymbol{\phi}^\top(\mathbf{x}_*) \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}_*) + \sigma^2). \quad (9.57c)$$

5852 The term $\boldsymbol{\phi}^\top(\mathbf{x}_*) \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}_*)$ reflects the posterior uncertainty associated 5853 with the parameters $\boldsymbol{\theta}$. Note that \mathbf{S}_N depends on the training inputs 5854 through Φ , see (9.43b). The predictive mean $\boldsymbol{\phi}^\top(\mathbf{x}_*) \mathbf{m}_N$ coincides with 5855 the MAP estimate.

5856 *Remark* (Marginal Likelihood and Posterior Predictive Distribution). By 5857 replacing the integral in (9.57a) the predictive distribution can be equivalently written as the expectation $\mathbb{E}_{\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}}[p(y_* | \mathbf{x}_*, \boldsymbol{\theta})]$, where the expectation is taken with respect to the parameter posterior $p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y})$.

5860 Writing the posterior predictive distribution in this way highlights a 5861 close resemblance to the marginal likelihood (9.42). The key difference

5862 between the marginal likelihood and the posterior predictive distribution
 5863 are (i) the marginal likelihood can be thought of predicting the training
 5864 targets \mathbf{y} and not the test targets y_* , (ii) the marginal likelihood averages
 5865 with respect to the parameter prior and not the parameter posterior. \diamond

Remark (Mean and Variance of Noise-Free Function Values). In many cases, we are not interested in the predictive distribution $p(y_* | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$ of a (noisy) observation y_* . Instead, we would like to obtain the distribution of the (noise-free) function values $f(\mathbf{x}_*) = \boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\theta}$. We determine the corresponding moments by exploiting the properties of means and variances, which yields

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}_*) | \mathcal{X}, \mathcal{Y}] &= \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}] = \boldsymbol{\phi}^\top(\mathbf{x}_*)\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}] \\ &= \boldsymbol{\phi}^\top(\mathbf{x}_*)\mathbf{m}_N = \mathbf{m}_N^\top\boldsymbol{\phi}(\mathbf{x}_*),\end{aligned}\quad (9.58)$$

$$\begin{aligned}\mathbb{V}_{\boldsymbol{\theta}}[f(\mathbf{x}_*) | \mathcal{X}, \mathcal{Y}] &= \mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\mathbf{x}_*)\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}] \\ &= \boldsymbol{\phi}^\top(\mathbf{x}_*)\mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y}]\boldsymbol{\phi}(\mathbf{x}_*) \\ &= \boldsymbol{\phi}^\top(\mathbf{x}_*)\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}_*)\end{aligned}\quad (9.59)$$

5866 We see that the predictive mean is the same as the predictive mean for
 5867 noisy observations as the noise has mean 0, and the predictive variance
 5868 only differs by σ^2 , which is the variance of the measurement noise: When
 5869 we predict noisy function values, we need to include σ^2 as a source of
 5870 uncertainty, but this term is not needed for noise-free predictions. Here,
 5871 the only remaining uncertainty stems from the parameter posterior. \diamond

5872 *Remark* (Distribution over Functions). The fact that we integrate out the
 5873 parameters $\boldsymbol{\theta}$ induces a distribution over functions: If we sample $\boldsymbol{\theta}_i \sim$
 5874 $p(\boldsymbol{\theta} | \mathcal{X}, \mathcal{Y})$ from the parameter posterior, we obtain a single function re-
 5875 alization $\boldsymbol{\theta}_i^\top\boldsymbol{\phi}(\cdot)$. The *mean function*, i.e., the set of all expected function
 5876 values $\mathbb{E}_{\boldsymbol{\theta}}[f(\cdot) | \boldsymbol{\theta}, \mathcal{X}, \mathcal{Y}]$, of this distribution over functions is $\mathbf{m}_N^\top\boldsymbol{\phi}(\cdot)$.
 5877 The (marginal) variance, i.e., the variance of the function $f(\cdot)$, is given by
 5878 $\boldsymbol{\phi}^\top(\cdot)\mathbf{S}_N\boldsymbol{\phi}(\cdot)$. \diamond

Integrating out
parameters induces
a distribution over
functions.

mean function

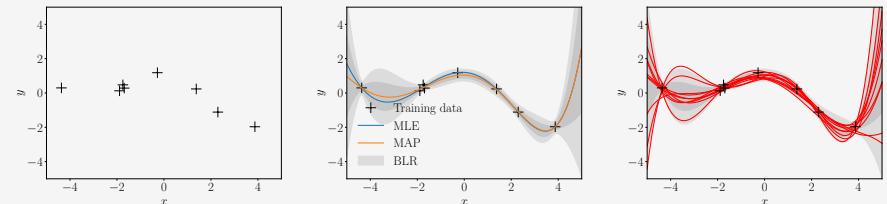
Example 9.8 (Posterior over Functions)

Let us revisit the Bayesian linear regression problem with polynomials of degree 5. We choose a parameter prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \frac{1}{4}\mathbf{I})$. 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 panel (a); Panel (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. Panel (c)

shows some plausible realizations (samples) of functions under that posterior over functions.

Figure 9.10
Bayesian linear regression and posterior distribution over functions. (a) Training data; (b) posterior distribution over functions; (c) Samples from the posterior over functions.

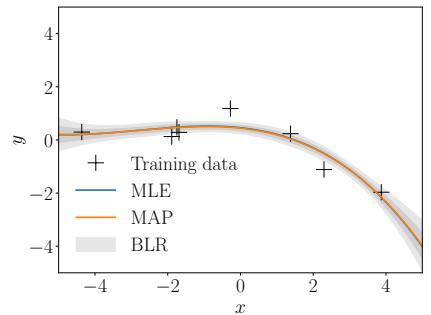


(a) Training data.

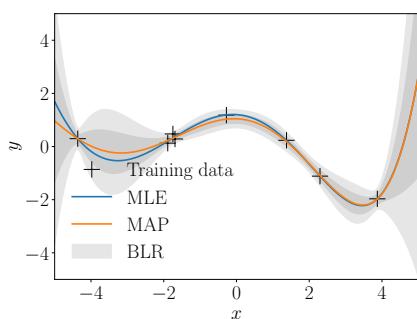
(b) Posterior over functions represented by the marginal uncertainty (shaded) showing the 95% predictive confidence bounds, the maximum likelihood estimate (MLE) and the MAP estimate (MAP), which is identical to the posterior mean function.

(c) Samples from the posterior over functions, which are induced by the samples from the 95% predictive confidence parameter posterior.

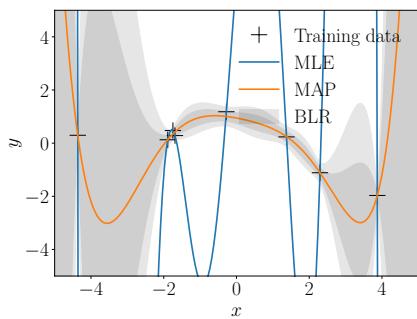
5879 Figure 9.11 shows some posterior distributions over functions induced
 5880 by the parameter posterior. For different polynomial degrees M the left
 5881 panels show the maximum likelihood function $\theta_{\text{ML}}^\top \phi(\cdot)$, the MAP function
 5882 $\theta_{\text{MAP}}^\top \phi(\cdot)$ (which is identical to the posterior mean function) and the 67%
 5883 and 95% predictive confidence bounds obtained by Bayesian linear re-
 5884 gression, represented by the shaded areas. The right panels show samples
 5885 from the posterior over functions: Here, we sampled parameters θ_i from
 5886 the parameter posterior and computed the function $\phi^\top(x_*)\theta_i$, which is a
 5887 single realization of a function under the posterior distribution over func-
 5888 tions. For low-order polynomials, the parameter posterior does not allow
 5889 the parameters to vary much: The sampled functions are nearly identical.
 5890 When we make the model more flexible by adding more parameters (i.e.,
 5891 we end up with a higher-order polynomial), these parameters are not suf-
 5892 ficiently constrained by the posterior, and the sampled functions can be
 5893 easily visually separated. We also see in the corresponding panels on the
 5894 left how the uncertainty increases, especially at the boundaries. Although
 5895 for a 7th-order polynomial the MAP estimate yields a reasonable fit, the
 5896 Bayesian linear regression model additionally tells us that the posterior
 5897 uncertainty is huge. This information can be critical when we use these
 5898 predictions in a decision-making system, where bad decisions can have
 5899 significant consequences (e.g., in reinforcement learning or robotics).



(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 67% (dark-gray) and 95% (light-gray) 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.

5900

9.3.5 Computing the Marginal Likelihood

5901 In Section 8.5.2, we highlighted the importance of the marginal likelihood
 5902 for Bayesian model selection. In the following, we compute the marginal
 5903 likelihood for Bayesian linear regression with a conjugate Gaussian prior
 5904 on the parameters, i.e., exactly the setting we have been discussing in this
 5905 chapter.

Just to re-cap, we consider the following generative process:

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0) \quad (9.60a)$$

$$y_n | \mathbf{x}_n, \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{x}_n^\top \boldsymbol{\theta}, \sigma^2), \quad (9.60b)$$

The marginal likelihood can be interpreted as the expected likelihood under the prior, i.e., $\mathbb{E}_{\boldsymbol{\theta}}[p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta})]$.

$n = 1, \dots, N$. The marginal likelihood is given by

$$p(\mathcal{Y} | \mathcal{X}) = \int p(\mathcal{Y} | \mathcal{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \quad (9.61a)$$

$$= \int \mathcal{N}(\mathbf{y} | \mathbf{X}\boldsymbol{\theta}, \sigma^2 \mathbf{I}) \mathcal{N}(\boldsymbol{\theta} | \mathbf{m}_0, \mathbf{S}_0) d\boldsymbol{\theta}, \quad (9.61b)$$

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 \mathbf{y}); Second, we compute the mean and covariance of this Gaussian.

- 5906 1 The marginal likelihood is Gaussian: From Section 6.5.2 we know that
- 5911 (i) the product of two Gaussian random variables is an (unnormalized)
- 5912 Gaussian distribution, (ii) a linear transformation of a Gaussian random
- 5913 variable is Gaussian distributed. In (9.61b), we require a linear trans-
- 5914 formation to bring $\mathcal{N}(\mathbf{y} | \mathbf{X}\boldsymbol{\theta}, \sigma^2 \mathbf{I})$ into the form $\mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ for some
- 5915 $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. Once this is done, the integral can be solved in closed form. The
- 5916 result is the normalizing constant of the product of the two Gaussians.
- 5917 The normalizing constant itself has Gaussian shape, see (6.80).
- 5918 2 Mean and covariance. We compute the mean and covariance matrix of
- 5919 the marginal likelihood by exploiting the standard results for means
- 5920 and covariances of affine transformations of random variables, see Sec-
- 5921 tion 6.4.4. The mean of the marginal likelihood is computed as

$$\mathbb{E}_{\boldsymbol{\theta}}[\mathcal{Y} | \mathcal{X}] = \mathbb{E}_{\boldsymbol{\theta}}[\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}] = \mathbf{X}\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta}] = \mathbf{X}\mathbf{m}_0. \quad (9.62)$$

Note that $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ is a vector of i.i.d. random variables. The covariance matrix is given as

$$\text{Cov}_{\boldsymbol{\theta}}[\mathcal{Y} | \mathcal{X}] = \text{Cov}[\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}] = \mathbf{X} \text{Cov}_{\boldsymbol{\theta}}[\boldsymbol{\theta}] \mathbf{X}^\top + \sigma^2 \mathbf{I} \quad (9.63a)$$

$$= \mathbf{X}\mathbf{S}_0\mathbf{X}^\top + \sigma^2 \mathbf{I} \quad (9.63b)$$

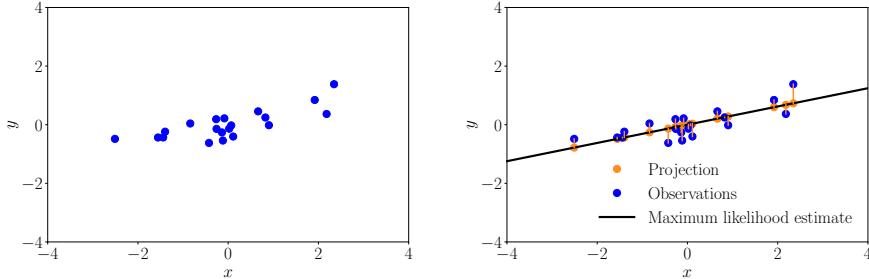
Hence, the marginal likelihood is

$$p(\mathcal{Y} | \mathcal{X}) = (2\pi)^{-\frac{N}{2}} \det(\mathbf{X}\mathbf{S}_0\mathbf{X}^\top + \sigma^2 \mathbf{I})^{-\frac{1}{2}} \quad (9.64a)$$

$$\cdot \exp(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{m}_0)^\top (\mathbf{X}\mathbf{S}_0\mathbf{X}^\top + \sigma^2 \mathbf{I})^{-1}(\mathbf{y} - \mathbf{X}\mathbf{m}_0))$$

$$= \mathcal{N}(\mathbf{y} | \mathbf{X}\mathbf{m}_0, \mathbf{S}_0 + \sigma^2 \mathbf{I}) \quad (9.64b)$$

Given the close connection with the posterior predictive distribution (see Remark on page 308), the functional form of the marginal likelihood should not be too surprising.



(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 $\theta_{ML}x$. 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.

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

$$y = x\theta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \quad (9.65)$$

in which we consider linear functions $f : \mathbb{R} \rightarrow \mathbb{R}$ 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 $\{(x_1, y_1), \dots, (x_N, y_N)\}$ we recall the results from Section 9.2.1 and obtain the maximum likelihood estimator for the slope parameter as

$$\theta_{ML} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \frac{\mathbf{X}^\top \mathbf{y}}{\mathbf{X}^\top \mathbf{X}} \in \mathbb{R}, \quad (9.66)$$

where $\mathbf{X} = [x_1, \dots, x_N]^\top \in \mathbb{R}^N$, $\mathbf{y} = [y_1, \dots, y_N]^\top \in \mathbb{R}^N$.

This means for the training inputs \mathbf{X} we obtain the optimal (maximum likelihood) reconstruction of the training targets as

$$\mathbf{X}\theta_{ML} = \mathbf{X} \frac{\mathbf{X}^\top \mathbf{y}}{\mathbf{X}^\top \mathbf{X}} = \frac{\mathbf{X}\mathbf{X}^\top}{\mathbf{X}^\top \mathbf{X}} \mathbf{y}, \quad (9.67)$$

i.e., we obtain the approximation with the minimum least-squares error between \mathbf{y} and $\mathbf{X}\theta$.

As we are looking for a solution of $\mathbf{y} = \mathbf{X}\theta$, 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

Linear regression can be thought of as a method for solving systems of linear equations.

Maximum likelihood linear regression performs an orthogonal projection. at (9.67) we see that the maximum likelihood estimator θ_{ML} in our example from (9.65) effectively does an orthogonal projection of \mathbf{y} onto the one-dimensional subspace spanned by \mathbf{X} . Recalling the results on orthogonal projections from Section 3.7, we identify $\frac{\mathbf{X}\mathbf{X}^\top}{\mathbf{X}^\top\mathbf{X}}$ as the projection matrix, θ_{ML} as the coordinates of the projection onto the one-dimensional subspace of \mathbb{R}^N spanned by \mathbf{X} and $\mathbf{X}\theta_{\text{ML}}$ as the orthogonal projection of \mathbf{y} onto this subspace.

Therefore, the maximum likelihood solution provides also a geometrically optimal solution by finding the vectors in the subspace spanned by \mathbf{X} that are “closest” to the corresponding observations \mathbf{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 projection of the noisy observations onto the subspace that minimizes the squared distance between the original dataset and its projection (note that the x -coordinate is fixed), which corresponds to the maximum likelihood solution.

In the general linear regression case where

$$y = \phi^\top(\mathbf{x})\theta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \quad (9.68)$$

with vector-valued features $\phi(\mathbf{x}) \in \mathbb{R}^K$, we again can interpret the maximum likelihood result

$$\mathbf{y} \approx \Phi\theta_{\text{ML}}, \quad (9.69)$$

$$\theta_{\text{ML}} = \Phi(\Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y} \quad (9.70)$$

as a projection onto a K -dimensional subspace of \mathbb{R}^N , which is spanned by the columns of the feature matrix Φ , see Section 3.7.2.

If the feature functions ϕ_k that we use to construct the feature matrix Φ are orthonormal (see Section 3.6), we obtain a special case where the columns of Φ form an orthonormal basis (see Section 3.5), such that $\Phi^\top\Phi = \mathbf{I}$. This will then lead to the projection

$$\Phi(\Phi^\top\Phi)^{-1}\Phi\mathbf{y} = \Phi\Phi^\top\mathbf{y} = \left(\sum_{k=1}^K \phi_k\phi_k^\top \right) \mathbf{y} \quad (9.71)$$

so that the coupling between different features has disappeared and the maximum likelihood projection is simply the sum of projections of \mathbf{y} onto the individual basis vectors ϕ_k , i.e., the columns of Φ . 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).

5957

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 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 $f = \boldsymbol{\theta}^\top \phi(\mathbf{x})$ is the linear regression model from (9.13). 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 $y = 1$ of a Bernoulli random variable $y \in \{0, 1\}$. 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 $\mathbf{y} = \sigma(\mathbf{A}\mathbf{x} + \mathbf{b})$, where \mathbf{A} is a weight matrix and \mathbf{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

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{f}_k(\mathbf{x}_k) \\ \mathbf{f}_k(\mathbf{x}_k) &= \sigma_k(\mathbf{A}_k \mathbf{x}_k + \mathbf{b}_k)\end{aligned}\tag{9.72}$$

5958 for $k = 0, \dots, K - 1$ where \mathbf{x}_0 are the input features and $\mathbf{x}_K = \mathbf{y}$
 5959 are the observed outputs, such that $\mathbf{f}_{K-1} \circ \dots \circ \mathbf{f}_0$ is a K -layer deep
 5960 neural network. Therefore, the building blocks of this deep neural net-
 5961 work are the generalized linear models defined in (9.72). A great post
 5962 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 re-
 5964 gression models. However, maximum likelihood parameter estimation is a
 5965

classification

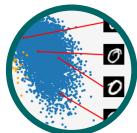
generalized linear
modelslogistic regression
logistic sigmoidtransfer function
activation function
canonical link
function
For ordinary linear
regression the
activation function
would simply be the
identity.Generalized linear
models are the
building blocks of
deep neural
networks.

5966 non-convex optimization problem, and marginalization of the parameters
 5967 in a fully Bayesian setting is analytically intractable.

5968 We briefly hinted at the fact that a distribution over parameters in-
 5969 duces a distribution over regression functions. *Gaussian processes* (Ras-
 5970 mussen and Williams, 2006) are regression models where the concept of
 5971 a distribution over function is central. Instead of placing a distribution
 5972 over parameters a Gaussian process places a distribution directly on the
 5973 space of functions without the “detour” via the parameters. To do so, the
 5974 Gaussian process exploits the *kernel trick* (Schölkopf and Smola, 2002),
 5975 which allows us to compute inner products between two function values
 5976 $f(\mathbf{x}_i), f(\mathbf{x}_j)$ only by looking at the corresponding input $\mathbf{x}_i, \mathbf{x}_j$. A Gaus-
 5977 sian process is closely related to both Bayesian linear regression and sup-
 5978 port vector regression but can also be interpreted as a Bayesian neural
 5979 network with a single hidden layer where the number of units tends to
 5980 infinity (Neal, 1996; Williams, 1997). An excellent introduction to Gaus-
 5981 sian processes can be found in (MacKay, 1998; Rasmussen and Williams,
 5982 2006).

5983 We focused on Gaussian parameter priors in the discussions in this chap-
 5984 ters because they allow for closed-form inference in linear regression mod-
 5985 els. However, even in a regression setting with Gaussian likelihoods we
 5986 may choose a non-Gaussian prior. Consider a setting where the inputs are
 5987 $\mathbf{x} \in \mathbb{R}^D$ and our training set is small and of size $N \ll D$. This means that
 5988 the regression problem is under-determined. In this case, we can choose
 5989 a parameter prior that enforces sparsity, i.e., a prior that tries to set as
 5990 many parameters to 0 as possible (*variable selection*). This prior provides
 5991 a stronger regularizer than the Gaussian prior, which often leads to an in-
 5992 creased prediction accuracy and interpretability of the model. The Laplace
 5993 prior is one example that is frequently used for this purpose. A linear re-
 5994 gression model with the Laplace prior on the parameters is equivalent to
 5995 linear regression with L1 regularization (*LASSO*) (Tibshirani, 1996). The
 5996 Laplace distribution is sharply peaked at zero (its first derivative is discon-
 5997 tinuous) and it concentrates its probability mass closer to zero than the
 5998 Gaussian distribution, which encourages parameters to be 0. Therefore,
 5999 the non-zero parameters are relevant for the regression problem, which is
 6000 the reason why we also speak of “variable selection”.

Dimensionality Reduction with Principal Component Analysis



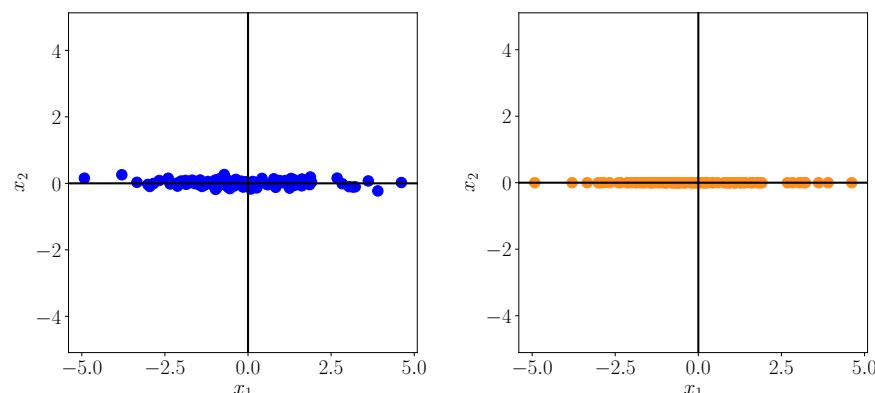
A 640×480 pixels⁵⁸²⁹ color image is a data⁵⁸³⁰ point in a million-dimensional⁵⁸³¹ space, where every⁵⁸³² pixel responds to⁵⁸³³ three dimensions,⁵⁸³⁴ one for each color⁵⁸³⁵ channel (red, green,⁵⁸³⁶ blue).⁵⁸³⁷

principal component⁵⁸⁴²
analysis⁵⁸⁴³
dimensionality⁵⁸⁴⁴
reduction⁵⁸⁴⁵
⁵⁸⁴⁶
⁵⁸⁴⁷

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 sig-

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.



(a) Dataset with x_1 and x_2 coordinates.

(b) Compressed dataset where only the x_1 coordinate is relevant.

nal processing community, PCA is also known as the *Karhunen-Loëve transform*. In this chapter, we derive PCA from first principles, drawing on our understanding of basis and basis change (Sections 2.6.1 and 2.7.2), projections (Section 3.7), eigenvalues (Section 4.2), Gaussian distributions (Section 6.5) and constrained optimization (Section 7.2).

Dimensionality reduction generally exploits a property of high-dimensional data (e.g., images) that it often lies on a low-dimensional subspace. 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 .

Karhunen-Loëve transform

10.1 Problem Setting

In PCA, we are interested in finding projections $\tilde{\mathbf{x}}_n$ of data points \mathbf{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 $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\mathbf{x}_n \in \mathbb{R}^D$, with mean $\mathbf{0}$ that possesses the *data covariance matrix* (6.42)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top. \quad (10.1)$$

data covariance matrix

Furthermore, we assume there exists a low-dimensional compressed representation (code)

$$\mathbf{z}_n = \mathbf{B}^\top \mathbf{x}_n \in \mathbb{R}^M \quad (10.2)$$

of \mathbf{x}_n , where we define the projection matrix

$$\mathbf{B} := [\mathbf{b}_1, \dots, \mathbf{b}_M] \in \mathbb{R}^{D \times M}. \quad (10.3)$$

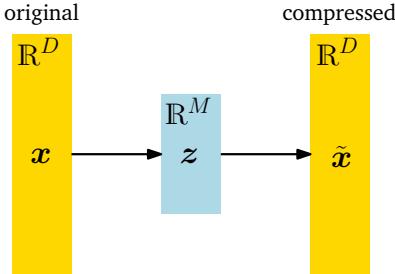
We assume that the columns of \mathbf{B} are orthonormal (Definition 3.7) so that $\mathbf{b}_i^\top \mathbf{b}_j = 0$ if and only if $i \neq j$ and $\mathbf{b}_i^\top \mathbf{b}_i = 1$. We seek an M -dimensional subspace $U \subseteq \mathbb{R}^D$, $\dim(U) = M < D$ onto which we project the data. We denote the projected data by $\tilde{\mathbf{x}}_n \in U$, and their coordinates (with respect to the basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$ of U) by \mathbf{z}_n . Our aim is to find projections $\tilde{\mathbf{x}}_n \in \mathbb{R}^D$ (or equivalently the codes \mathbf{z}_n and the basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$) so that they are as similar to the original data \mathbf{x}_n and minimize the loss due to compression.

The columns $\mathbf{b}_1, \dots, \mathbf{b}_M$ of \mathbf{B} form a basis of the M -dimensional subspace in which the projected data $\tilde{\mathbf{x}} = \mathbf{B}\mathbf{B}^\top \mathbf{x} \in \mathbb{R}^D$ live.

Example 10.1 (Coordinate Representation/Code)

Consider \mathbb{R}^2 with the canonical basis $\mathbf{e}_1 = [1, 0]^\top$, $\mathbf{e}_2 = [0, 1]^\top$. From

Figure 10.2
 Graphical illustration of PCA.
 In PCA, we find a compressed version \tilde{x} of original data x that has an intrinsic lower-dimensional representation z .



Chapter 2 we know that $x \in \mathbb{R}^2$ can be represented as a linear combination of these basis vectors, e.g.,

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} = 5e_1 + 3e_2. \quad (10.4)$$

However, when we consider vectors of the form

$$\tilde{x} = \begin{bmatrix} 0 \\ z \end{bmatrix} \in \mathbb{R}^2, \quad z \in \mathbb{R}, \quad (10.5)$$

they can always be written as $0e_1 + ze_2$. To represent these vectors it is sufficient to remember/store the *coordinate/code* z of \tilde{x} with respect to the e_2 vector.

More precisely, the set of \tilde{x} vectors (with the standard vector addition and scalar multiplication) forms a vector subspace U (see Section 2.4) with $\dim(U) = 1$ because $U = \text{span}[e_2]$.

The dimension of a vector space corresponds to the number of its basis vectors (see Section 2.6.1).

5874 In Section 10.2, we will find low-dimensional representations that
 5875 retain as much information as possible and minimize the compression loss.
 5876 An alternative derivation of PCA is given in Section 10.3, we will be looking
 5877 at minimizing the squared reconstruction error $\|x_n - \tilde{x}_n\|^2$ between
 5878 the original data x_n and its projection \tilde{x}_n .

5879 Figure 10.2 illustrates the setting we consider in PCA, where z represents
 5880 the intrinsic lower dimension of the compressed data \tilde{x} and plays
 5881 the role of a bottleneck, which controls how much information can flow
 5882 between x and \tilde{x} . In PCA, we consider a linear relationship between the
 5883 original data x and its low-dimensional code z so that $z = B^\top x$ and
 5884 $\tilde{x} = Bz$ for a suitable matrix B . Based the motivation of thinking of
 5885 PCA as a data compression technique, we can interpret the arrows in Figure
 5886 10.2 as a pair of operations representing encoders and decoders. The
 5887 linear mapping represented by B can be thought of a decoder, which maps
 5888 the low-dimensional code $z \in \mathbb{R}^M$ back into the original data space \mathbb{R}^D .
 5889 Similarly, B^\top can be thought of an encoder, which encodes the original
 5890 data x as a low-dimensional (compressed) code z .

5891 Throughout this chapter, we will use the MNIST digits dataset as a re-



Figure 10.3
Examples of handwritten digits from the MNIST dataset. <http://yann.lecun.com/exdb/mnist/>

5892 occurring example, which contains 60,000 examples of handwritten digits
 5893 0–9. Each digit is a grayscale image of size 28×28 , i.e., it contains 784
 5894 pixels so that we can interpret every image in this dataset as a vector
 5895 $\mathbf{x} \in \mathbb{R}^{784}$. Examples of these digits are shown in Figure 10.3.

5896 10.2 Maximum Variance Perspective

5897 Figure 10.1 gave an example of how a two-dimensional dataset can be
 5898 represented using a single coordinate. In Figure 10.1(b), we chose to ig-
 5899 nore the x_2 -coordinate of the data because it did not add too much in-
 5900 formation so that the compressed data is similar to the original data in
 5901 Figure 10.1(a). We could have chosen to ignore the x_1 -coordinate, but
 5902 then the compressed data had been very dissimilar from the original data,
 5903 and much information in the data would have been lost.

5904 If we interpret information content in the data as how “space filling”
 5905 the data set is, then we can describe the information contained in the data
 5906 by looking at the spread of the data. From Section 6.4.1 we know that the
 5907 variance is an indicator of the spread of the data, and we can derive PCA as
 5908 a dimensionality reduction algorithm that maximizes the variance in the
 5909 low-dimensional representation of the data to retain as much information
 5910 as possible. Figure 10.4 illustrates this.

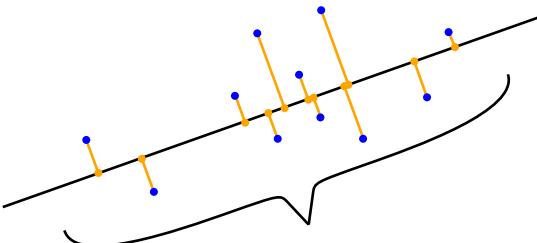
5911 Considering the setting discussed in Section 10.1, our aim is to find
 5912 a matrix \mathbf{B} (see (10.3)) that retains as much information as possible
 5913 when compressing data by projecting it onto the subspace spanned by
 5914 the columns $\mathbf{b}_1, \dots, \mathbf{b}_M$ of \mathbf{B} . Retaining most information after data com-
 5915 pression is equivalent to capturing the largest amount of variance in the
 5916 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 6.4.4 we obtain

$$\mathbb{V}_{\mathbf{z}}[\mathbf{z}] = \mathbb{V}_{\mathbf{x}}[\mathbf{B}^T(\mathbf{x} - \boldsymbol{\mu})] = \mathbb{V}_{\mathbf{x}}[\mathbf{B}^T\mathbf{x} - \mathbf{B}^T\boldsymbol{\mu}] = \mathbb{V}_{\mathbf{x}}[\mathbf{B}^T\mathbf{x}], \quad (10.6)$$

5917 i.e., the variance of the low-dimensional code does not depend on the
 5918 mean of the data. Therefore, we assume without loss of generality that the
 5919 data has mean $\mathbf{0}$ for the remainder of this section. With this assumption
 5920 the mean of the low-dimensional code is also $\mathbf{0}$ since $\mathbb{E}_{\mathbf{z}}[\mathbf{z}] = \mathbb{E}_{\mathbf{x}}[\mathbf{B}^T\mathbf{x}] =$
 5921 $\mathbf{B}^T\mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \mathbf{0}$. \diamond

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).



5922

The vector \mathbf{b}_1 will be the first column of the matrix \mathbf{B} and therefore the first of M orthonormal basis vectors that span the lower-dimensional subspace.

We maximize the variance of the low-dimensional code using a sequential approach. We start by seeking a single vector $\mathbf{b}_1 \in \mathbb{R}^D$ that maximizes the variance of the projected data, i.e., we aim to maximize the variance of the first coordinate z_1 of $\mathbf{z} \in \mathbb{R}^M$ so that

$$V_1 := \mathbb{V}[z_1] = \frac{1}{N} \sum_{n=1}^N z_{1n}^2 \quad (10.7)$$

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 $\mathbf{z}_n \in \mathbb{R}^M$ of $\mathbf{x}_n \in \mathbb{R}^D$. Note that first component of \mathbf{z}_n is given by

$$z_{1n} = \mathbf{b}_1^\top \mathbf{x}_n, \quad (10.8)$$

i.e., it is the coordinate of the orthogonal projection of \mathbf{x}_n onto the one-dimensional subspace spanned by \mathbf{b}_1 (Section 3.7). We substitute (10.8) into (10.7), which yields

$$V_1 = \frac{1}{N} \sum_{n=1}^N (\mathbf{b}_1^\top \mathbf{x}_n)^2 = \frac{1}{N} \sum_{n=1}^N \mathbf{b}_1^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{b}_1 \quad (10.9a)$$

$$= \mathbf{b}_1^\top \left(\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \right) \mathbf{b}_1 = \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1, \quad (10.9b)$$

where \mathbf{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 $\mathbf{b}_1^\top \mathbf{x}_n = \mathbf{x}_n^\top \mathbf{b}_1$.

Notice that arbitrarily increasing the magnitude of the vector \mathbf{b}_1 increases V_1 , that is, a vector \mathbf{b}_1 that is two times longer can result in V_1 that is potentially four times larger. Therefore, we restrict all solutions to $\|\mathbf{b}_1\|^2 = 1$, which results in a constrained optimization problem in which we seek the direction along which the data varies most.

With the restriction of the solution space to unit vectors the vector \mathbf{b}_1 that points in the direction of maximum variance can be found by the

constrained optimization problem

$$\begin{aligned} & \max_{\mathbf{b}_1} \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1 \\ & \text{subject to } \|\mathbf{b}_1\|^2 = 1. \end{aligned} \quad (10.10)$$

Following Section 7.2, we obtain the Lagrangian

$$\mathcal{L}(\mathbf{b}_1, \lambda) = \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1 + \lambda(1 - \mathbf{b}_1^\top \mathbf{b}_1) \quad (10.11)$$

to solve this constrained optimization problem. The partial derivatives of \mathcal{L} with respect to \mathbf{b}_1 and λ are

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}_1} = 2\mathbf{b}_1^\top \mathbf{S} - 2\lambda_1 \mathbf{b}_1^\top, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \mathbf{b}_1^\top \mathbf{b}_1, \quad (10.12)$$

respectively. Setting these partial derivatives to 0 gives us the relations

$$\mathbf{S} \mathbf{b}_1 = \lambda_1 \mathbf{b}_1, \quad (10.13)$$

$$\mathbf{b}_1^\top \mathbf{b}_1 = 1. \quad (10.14)$$

By comparing with the definition of an eigenvalue decomposition (Section 4.4), we see that \mathbf{b}_1 is an eigenvector of the data covariance matrix \mathbf{S} , and the Lagrange multiplier λ_1 plays the role of the corresponding eigenvalue. This eigenvector property (10.13) allows us to rewrite our variance objective (10.10) as

$$V_1 = \mathbf{b}_1^\top \mathbf{S} \mathbf{b}_1 = \lambda_1 \mathbf{b}_1^\top \mathbf{b}_1 = \lambda_1, \quad (10.15)$$

i.e., the variance of the data projected onto a one-dimensional subspace equals the eigenvalue that is associated with the basis vector \mathbf{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 \mathbf{b}_1 in the original data space by mapping the coordinate z_{1n} back into data space, which gives us the projected data point

$$\tilde{\mathbf{x}}_n = \mathbf{b}_1 z_{1n} = \mathbf{b}_1 \mathbf{b}_1^\top \mathbf{x}_n \in \mathbb{R}^D \quad (10.16)$$

⁵⁹³¹ in the original data space.

⁵⁹³² *Remark.* Although $\tilde{\mathbf{x}}_n$ is a D -dimensional vector it only requires a single coordinate z_{1n} to represent it with respect to the basis vector $\mathbf{b}_1 \in \mathbb{R}^D$. ◇

⁵⁹³⁴ 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 \mathbf{S} that are associated with the largest $m - 1$ eigenvalues. Since \mathbf{S} is symmetric, these eigenvectors form an orthonormal basis (ONB) of an $(m - 1)$ -dimensional subspace of \mathbb{R}^D . Generally, the m th

The quantity $\sqrt{\lambda_1}$ is also called the *loading* of the unit vector \mathbf{b}_1 and represents the standard deviation of the data accounted for by the principal subspace $\text{span}[\mathbf{b}_1]$.
principal component

principal component can be found by subtracting the effect of the first $m - 1$ principal components $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$ 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 (10.16), so that we arrive at the new data matrix

$$\hat{\mathbf{X}} := \mathbf{X} - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{X}, \quad (10.17)$$

where $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$ contains the data points as column vectors. The matrix $\hat{\mathbf{X}} := [\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N] \in \mathbb{R}^{D \times N}$ in (10.17) 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 $\mathbf{x}_1, \dots, \mathbf{x}_N$ as the rows of the data matrix, but we define them to be the columns of \mathbf{X} . This means that our data matrix \mathbf{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. \diamond

To find the m th principal component, we maximize the variance

$$V_m = \mathbb{V}[z_m] = \frac{1}{N} \sum_{n=1}^N z_{mn}^2 = \frac{1}{N} \sum_{n=1}^N (\mathbf{b}_m^\top \mathbf{x}_n)^2 = \mathbf{b}_m^\top \hat{\mathbf{S}} \mathbf{b}_m, \quad (10.18)$$

subject to $\|\mathbf{b}_m\|^2 = 1$, where we followed the same steps as in (10.9b) and defined $\hat{\mathbf{S}}$ as the data covariance matrix of the transformed dataset $\hat{\mathcal{X}} := \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N\}$. As previously, when we looked at the first principal component alone, we solve a constrained optimization problem and discover that the optimal solution \mathbf{b}_m is the eigenvector of $\hat{\mathbf{S}}$ that is associated with the largest eigenvalue of $\hat{\mathbf{S}}$.

However, it also turns out that \mathbf{b}_m is an eigenvector of \mathbf{S} . It holds that

$$\hat{\mathbf{S}} = \frac{1}{N} \sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^\top \stackrel{(10.17)}{=} \frac{1}{N} \sum_{n=1}^N \left(\mathbf{x}_n - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \right) \left(\mathbf{x}_n - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \right)^\top \quad (10.19a)$$

$$= \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top - 2 \mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top + \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top, \quad (10.19b)$$

where we exploited the symmetries $\mathbf{x}_n^\top \mathbf{b}_i = \mathbf{b}_i^\top \mathbf{x}_n$ and $\mathbf{b}_i \mathbf{x}_n^\top = \mathbf{x}_n \mathbf{b}_i^\top$ to summarize

$$-\mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top \mathbf{x}_n \mathbf{x}_n^\top = -2 \mathbf{x}_n \mathbf{x}_n^\top \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^\top. \quad (10.20)$$

If we take a vector \mathbf{b}_m with $\|\mathbf{b}_m\| = 1$ that is orthogonal to all $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$ and right-multiply \mathbf{b}_m to $\hat{\mathbf{S}}$ in (10.19b) we obtain

$$\hat{\mathbf{S}}\mathbf{b}_m = \frac{1}{N} \sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^\top \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \mathbf{b}_m = \mathbf{S}\mathbf{b}_m = \lambda_m \mathbf{b}_m. \quad (10.21)$$

5951 Here we applied the orthogonality property $\mathbf{b}_i^\top \mathbf{b}_m = 0$ for $i = 1, \dots, m-1$
 5952 (all terms involving sums up to $m-1$ vanish). Equation (10.21) reveals
 5953 that \mathbf{b}_m is an eigenvector of both $\hat{\mathbf{S}}$ and the original data covariance ma-
 5954 trix \mathbf{S} . In other words, λ_m is the largest eigenvalue of $\hat{\mathbf{S}}$ and λ_m is the
 5955 m th largest eigenvalue of \mathbf{S} , and both have the associated eigenvector
 5956 \mathbf{b}_m . This derivation shows that there is an intimate connection between
 5957 the M -dimensional subspace with maximal variance and the eigenvalue
 5958 decomposition. We will revisit this connection in Section 10.4.

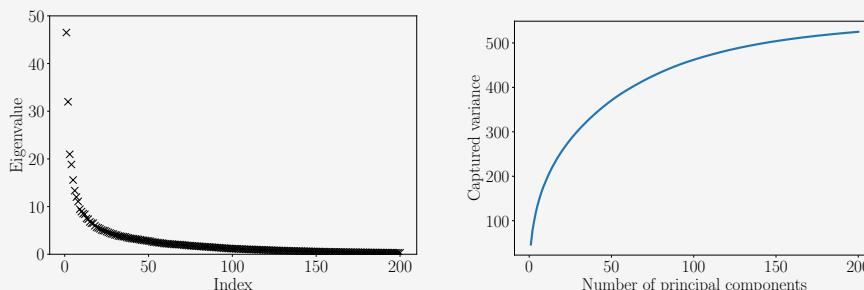
With the relation (10.21) and $\mathbf{b}_m^\top \mathbf{b}_m = 1$ the variance of the data projected onto the m th principal component is

$$V_m = \mathbf{b}_m^\top \mathbf{S} \mathbf{b}_m \stackrel{(10.21)}{=} \lambda_m \mathbf{b}_m^\top \mathbf{b}_m = \lambda_m. \quad (10.22)$$

This means that the variance of the data, when projected onto an M -dimensional subspace, equals the sum of the eigenvalues that are associated with the corresponding eigenvectors of the data covariance matrix.

Example 10.2 (Eigenvalues of MNIST ‘8’)

Taking all digits ‘8’ in the MNIST training data, we compute the eigenvalues of the data covariance matrix.

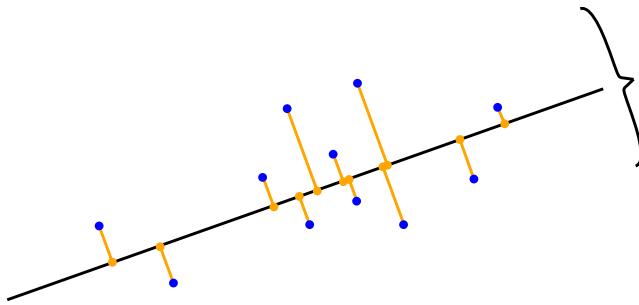


(a) Eigenvalues (sorted in descending order) of the data covariance matrix of all digits ‘8’ in the MNIST training set.
 (b) Variance captured by the principal components.

Figure 10.5
 Properties of the training data of MNIST ‘8’. (a) Eigenvalues sorted in descending order; (b) Variance captured by the principal components associated with the largest eigenvalues.

Figure 10.5(a) shows the 200 largest eigenvalues of the data covariance matrix. We see that only a few of them have a value that differs significantly from 0. Therefore, most of the variance, when projecting data onto the subspace spanned by the corresponding eigenvectors, is captured by only a few principal components as shown in Figure 10.5(b).

Figure 10.6
 Illustration of the projection approach: Find a subspace (line) that minimizes the length of the difference vector between projected (orange) and original (blue) data.



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 B in (10.3) as the M eigenvectors of the data covariance matrix S that are associated with the M largest eigenvalues. The maximum amount of variance PCA can capture with the first M principal components is

$$V_M = \sum_{m=1}^M \lambda_m, \quad (10.23)$$

where the λ_m are the M largest eigenvalues of the data covariance matrix S . Consequently, the variance lost by data compression via PCA is

$$J_M := \sum_{j=M+1}^D \lambda_j = V_D - V_M. \quad (10.24)$$

5959 Instead of these absolute quantities, we can define the relative variance
 5960 captured as $\frac{V_M}{V_D}$, and the relative variance lost by compression as $1 - \frac{V_M}{V_D}$.

5961

10.3 Projection Perspective

5962 In the following, we will derive PCA as an algorithm that directly mini-
 5963 mizes the average reconstruction error. This perspective allows us to in-
 5964 terpret PCA as implementing an optimal linear auto-encoder. We will draw
 5965 heavily from Chapters 2 and 3.

5966 In the previous section, we derived PCA by maximizing the variance
 5967 in the projected space to retain as much information as possible. In the
 5968 following, we will look at the difference vectors between the original data
 5969 x_n and their reconstruction \tilde{x}_n and minimize this distance so that x_n and
 5970 \tilde{x}_n are as close as possible. Figure 10.6 illustrates this setting.

5971

10.3.1 Setting and Objective

5972 Assume an (ordered) orthonormal basis (ONB) $B = (\mathbf{b}_1, \dots, \mathbf{b}_D)$ of \mathbb{R}^D ,
 5973 i.e., $\mathbf{b}_i^\top \mathbf{b}_j = 1$ if and only if $i = j$ and 0 otherwise.

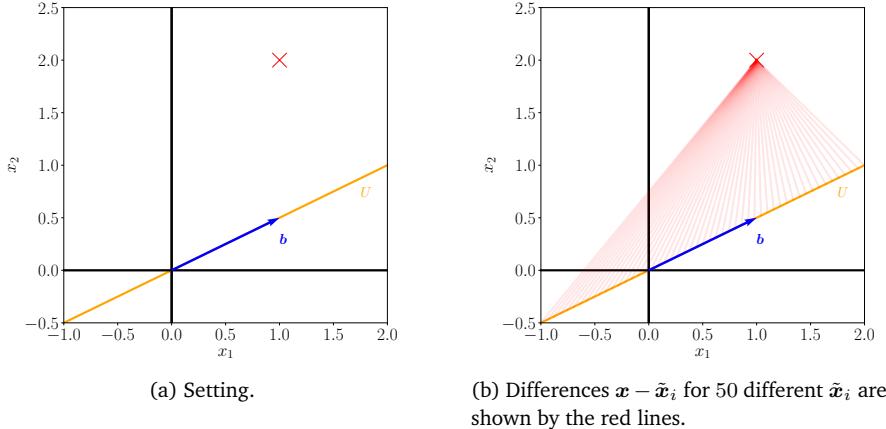


Figure 10.7
Simplified projection setting.
(a) A vector $\mathbf{x} \in \mathbb{R}^2$ (red cross) shall be projected onto a one-dimensional subspace $U \subseteq \mathbb{R}^2$ spanned by \mathbf{b} . (b) shows the difference vectors between \mathbf{x} and some candidates $\tilde{\mathbf{x}}$.

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 $\mathbf{x} \in V$ can be (uniquely) decomposed into

$$\mathbf{x} = \sum_{m=1}^M \lambda_m \mathbf{b}_m + \sum_{j=1}^{D-M} \psi_j \mathbf{b}_j^\perp, \quad \lambda_m, \psi_j \in \mathbb{R}, \quad (10.25)$$

5974 where $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ is a basis of U and $(\mathbf{b}_1^\perp, \dots, \mathbf{b}_{D-M}^\perp)$ is a basis of U^\perp .
5975 \diamond

From Section 2.5 we know that for a basis $(\mathbf{b}_1, \dots, \mathbf{b}_D)$ of \mathbb{R}^D any $\mathbf{x} \in \mathbb{R}^D$ can be written as a linear combination of the basis vectors of \mathbb{R}^D , i.e.,

$$\mathbf{x} = \sum_{d=1}^D \zeta_d \mathbf{b}_d = \sum_{m=1}^M \zeta_m \mathbf{b}_m + \sum_{j=M+1}^D \zeta_j \mathbf{b}_j \quad (10.26)$$

5976 for suitable coordinates $\zeta_d \in \mathbb{R}$.

We are interested in finding vectors $\tilde{\mathbf{x}} \in \mathbb{R}^D$, which live in lower-dimensional subspace $U \subseteq \mathbb{R}^D$, $\dim(U) = M$, so that

$$\tilde{\mathbf{x}} = \sum_{m=1}^M z_m \mathbf{b}_m \in U \subseteq \mathbb{R}^D \quad (10.27)$$

5977 is as similar to \mathbf{x} as possible. Note that at this point we need to assume
5978 that the coordinates z_m of $\tilde{\mathbf{x}}$ and ζ_m of \mathbf{x} are not identical.

Vectors $\tilde{\mathbf{x}} \in U$ 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 .

5979 In the following, we use exactly this kind of representation of $\tilde{\mathbf{x}}$ to find
5980 optimal coordinates z and basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$ such that $\tilde{\mathbf{x}}$ is as sim-
5981 ilar to the original data point \mathbf{x} , i.e., we aim to minimize the (Euclidean)
5982 distance $\|\mathbf{x} - \tilde{\mathbf{x}}\|$. Figure 10.7 illustrates this setting.

5983 Without loss of generality, we assume that the dataset $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$,

5984 $\mathbf{x}_n \in \mathbb{R}^D$, is centered at $\mathbf{0}$, i.e., $\mathbb{E}[\mathcal{X}] = \mathbf{0}$. Without the zero-mean assumption,
5985 we would arrive at exactly the same solution but the notation would
5986 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 $\mathbf{b}_1, \dots, \mathbf{b}_M$. We will call this subspace U the *principal subspace*. The projections of the data points are denoted by

$$\tilde{\mathbf{x}}_n := \sum_{m=1}^M z_{mn} \mathbf{b}_m = \mathbf{B} \mathbf{z}_n \in \mathbb{R}^D, \quad (10.28)$$

5987 where $\mathbf{z}_n := [z_{1n}, \dots, z_{Mn}]^\top \in \mathbb{R}^M$ is the coordinate vector of $\tilde{\mathbf{x}}_n$ with respect to the basis $(\mathbf{b}_1, \dots, \mathbf{b}_M)$. More specifically, we are interested in having the $\tilde{\mathbf{x}}_n$ as similar to \mathbf{x}_n as possible.

The similarity measure we use in the following is the squared Euclidean norm $\|\mathbf{x} - \tilde{\mathbf{x}}\|^2$ between \mathbf{x} and $\tilde{\mathbf{x}}$. We therefore define our objective as the minimizing the average squared Euclidean distance (*reconstruction error*) (Pearson, 1901b)

$$J_M := \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2, \quad (10.29)$$

5990 where we make it explicit that the dimension of the subspace onto which
5991 we project the data is M . In order to find this optimal linear projection,
5992 we need to find the orthonormal basis of the principal subspace and the
5993 coordinates $\mathbf{z}_n \in \mathbb{R}^M$ of the projections with respect to this basis.

5994 To find the coordinates \mathbf{z}_n and the ONB of the principal subspace we
5995 follow a two-step approach. First, we optimize the coordinates \mathbf{z}_n for a
5996 given ONB $(\mathbf{b}_1, \dots, \mathbf{b}_M)$; second, we find the optimal ONB.

5997 10.3.2 Finding Optimal Coordinates

5998 Let us start by finding the optimal coordinates z_{1n}, \dots, z_{Mn} of the projections $\tilde{\mathbf{x}}_n$ for $n = 1, \dots, N$. Consider Figure 10.7(b) where the principal subspace is spanned by a single vector \mathbf{b} . Geometrically speaking, finding the optimal coordinates z corresponds to finding the representation of the linear projection $\tilde{\mathbf{x}}$ with respect to \mathbf{b} that minimizes the distance between $\tilde{\mathbf{x}} - \mathbf{x}$. From Figure 10.7(b) it is clear that this will be the orthogonal projection, and in the following we will show exactly this.

We assume an ONB $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ of $U \subseteq \mathbb{R}^D$. To find the optimal coordinates \mathbf{z}_n with respect to this basis, we require the partial derivatives

$$\frac{\partial J_M}{\partial z_{in}} = \frac{\partial J_M}{\partial \tilde{\mathbf{x}}_n} \frac{\partial \tilde{\mathbf{x}}_n}{\partial z_{in}}, \quad (10.30a)$$

$$\frac{\partial J_M}{\partial \tilde{\mathbf{x}}_n} = -\frac{2}{N} (\mathbf{x}_n - \tilde{\mathbf{x}}_n)^\top \in \mathbb{R}^{1 \times D}, \quad (10.30b)$$

$$\frac{\partial \tilde{\mathbf{x}}_n}{\partial z_{in}} \stackrel{(10.28)}{=} \frac{\partial}{\partial z_{in}} \left(\sum_{m=1}^M z_{mn} \mathbf{b}_m \right) = \mathbf{b}_i \quad (10.30c)$$

for $i = 1, \dots, M$, such that we obtain

$$\frac{\partial J_M}{\partial z_{in}} \stackrel{(10.30b)}{=} -\frac{2}{N} (\mathbf{x}_n - \tilde{\mathbf{x}}_n)^\top \mathbf{b}_i \stackrel{(10.28)}{=} -\frac{2}{N} \left(\mathbf{x}_n - \sum_{m=1}^M z_{mn} \mathbf{b}_m \right)^\top \mathbf{b}_i \quad (10.31a)$$

$$\stackrel{\text{ONB}}{=} -\frac{2}{N} (\mathbf{x}_n^\top \mathbf{b}_i - z_{in} \mathbf{b}_i^\top \mathbf{b}_i) = -\frac{2}{N} (\mathbf{x}_n^\top \mathbf{b}_i - z_{in}). \quad (10.31b)$$

since $\mathbf{b}_i^\top \mathbf{b}_i = 1$. Setting this partial derivative to 0 yields immediately the optimal coordinates

$$z_{in} = \mathbf{x}_n^\top \mathbf{b}_i = \mathbf{b}_i^\top \mathbf{x}_n \quad (10.32)$$

for $i = 1, \dots, M$ and $n = 1, \dots, N$. This means that the optimal coordinates z_{in} of the projection $\tilde{\mathbf{x}}_n$ are the coordinates of the orthogonal projection (see Section 3.7) of the original data point \mathbf{x}_n onto the one-dimensional subspace that is spanned by \mathbf{b}_i . Consequently:

- The optimal linear projection $\tilde{\mathbf{x}}_n$ of \mathbf{x}_n is an orthogonal projection.
- The coordinates of $\tilde{\mathbf{x}}_n$ with respect to the basis $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ are the coordinates of the orthogonal projection of \mathbf{x}_n onto the principal subspace.
- An orthogonal projection is the best linear mapping given the objective (10.29).
- The coordinates ζ_m of \mathbf{x} in (10.26) and the coordinates z_m of $\tilde{\mathbf{x}}$ in (10.27) must be identical for $m = 1, \dots, M$ since $U^\perp = \text{span}[\mathbf{b}_{M+1}, \dots, \mathbf{b}_D]$ is the orthogonal complement of $U = \text{span}[\mathbf{b}_1, \dots, \mathbf{b}_M]$.

The coordinates of the optimal projection of \mathbf{x}_n with respect to the basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$ are the coordinates of the orthogonal projection of \mathbf{x}_n onto the principal subspace.

Remark (Orthogonal Projections with Orthonormal Basis Vectors). Let us briefly recap orthogonal projections from Section 3.7. If $(\mathbf{b}_1, \dots, \mathbf{b}_D)$ is an orthonormal basis of \mathbb{R}^D then

$$\tilde{\mathbf{x}} = \mathbf{b}_j (\mathbf{b}_j^\top \mathbf{b}_j)^{-1} \mathbf{b}_j^\top \mathbf{x} = \mathbf{b}_j \mathbf{b}_j^\top \mathbf{x} \in \mathbb{R}^D \quad (10.33)$$

is the orthogonal projection of \mathbf{x} onto the subspace spanned by the j th basis vector, and $z_j = \mathbf{b}_j^\top \mathbf{x}$ is the coordinate of this projection with respect to the basis vector \mathbf{b}_j that spans that subspace since $z_j \mathbf{b}_j = \tilde{\mathbf{x}}$. Figure 10.8 illustrates this setting.

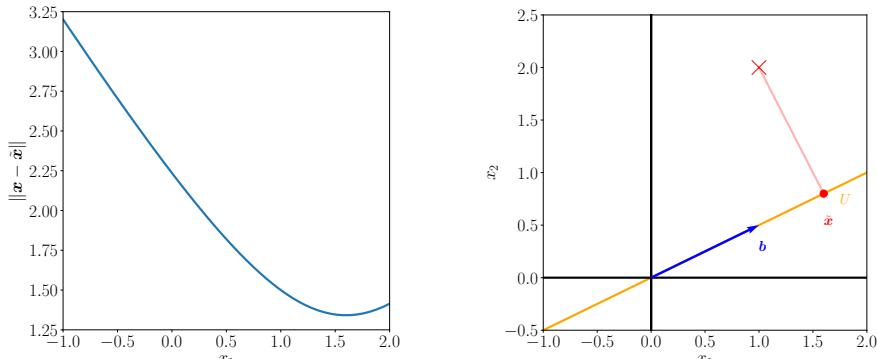
$\mathbf{b}_j^\top \mathbf{x}$ is the coordinate of the orthogonal projection of \mathbf{x} onto the subspace spanned by \mathbf{b}_j .

More generally, if we aim to project onto an M -dimensional subspace of \mathbb{R}^D , we obtain the orthogonal projection of \mathbf{x} onto the M -dimensional subspace with orthonormal basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$ as

$$\tilde{\mathbf{x}} = \mathbf{B} (\underbrace{\mathbf{B}^\top \mathbf{B}}_{=\mathbf{I}})^{-1} \mathbf{B}^\top \mathbf{x} = \mathbf{B} \mathbf{B}^\top \mathbf{x}, \quad (10.34)$$

where we defined $\mathbf{B} := [\mathbf{b}_1, \dots, \mathbf{b}_M] \in \mathbb{R}^{D \times M}$. The coordinates of this

Figure 10.8
 Optimal projection
 of a vector $\mathbf{x} \in \mathbb{R}^2$
 onto a
 one-dimensional
 subspace
 (continuation from
 Figure 10.7).
 (a) Distances
 $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ for some
 $\tilde{\mathbf{x}} \in U$.
 (b) Orthogonal
 projection and
 optimal coordinates.



(a) Distances $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ for some $\tilde{\mathbf{x}} = z_1 \mathbf{b} \in U = \text{span}[\mathbf{b}]$, see panel (b) for the setting.

(b) The vector $\tilde{\mathbf{x}}$ that minimizes the distance in panel (a) is its orthogonal projection onto U . The coordinate of the projection $\tilde{\mathbf{x}}$ with respect to the basis vector \mathbf{b} that spans U is the factor we need to scale \mathbf{b} in order to “reach” $\tilde{\mathbf{x}}$.

6023 projection with respect to the ordered basis $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ are $\mathbf{z} := \mathbf{B}^\top \mathbf{x}$
 6024 as discussed in Section 3.7.

6025 We can think of the coordinates as a representation of the projected
 6026 vector in a new coordinate system defined by $(\mathbf{b}_1, \dots, \mathbf{b}_M)$. Note that al-
 6027 though $\tilde{\mathbf{x}} \in \mathbb{R}^D$ we only need M coordinates z_1, \dots, z_M to represent this
 6028 vector; the other $D - M$ coordinates with respect to the basis vectors
 6029 $(\mathbf{b}_{M+1}, \dots, \mathbf{b}_D)$ are always 0. ◇

6030 So far we have shown that for a given ONB we can find the optimal
 6031 coordinates of $\tilde{\mathbf{x}}$ by an orthogonal projection onto the principal subspace.
 6032 In the following, we will determine what the best basis is.

6033 10.3.3 Finding the Basis of the Principal Subspace

To determine the basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$ of the principal subspace, we rephrase the loss function (10.29) 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

$$\tilde{\mathbf{x}}_n = \sum_{m=1}^M z_{mn} \mathbf{b}_m \stackrel{(10.32)}{=} \sum_{m=1}^M (\mathbf{x}_n^\top \mathbf{b}_m) \mathbf{b}_m. \quad (10.35)$$

We now exploit the symmetry of the dot product, which yields

$$\tilde{\mathbf{x}}_n = \left(\sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^\top \right) \mathbf{x}_n. \quad (10.36)$$

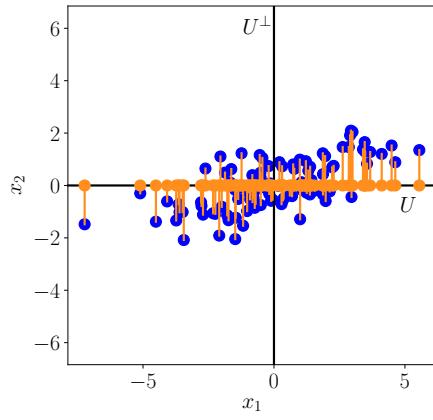


Figure 10.9
Orthogonal projection and displacement vectors. When projecting data points \mathbf{x}_n (blue) onto subspace U_1 we obtain $\tilde{\mathbf{x}}_n$ (orange). The displacement vector $\tilde{\mathbf{x}}_n - \mathbf{x}_n$ lies completely in the orthogonal complement U_2 of U_1 .

Since we can generally write the original data point \mathbf{x}_n as a linear combination of all basis vectors, it holds that

$$\mathbf{x}_n = \sum_{d=1}^D z_{dn} \mathbf{b}_d \stackrel{(10.32)}{=} \sum_{d=1}^D (\mathbf{x}_n^\top \mathbf{b}_d) \mathbf{b}_d = \left(\sum_{d=1}^D \mathbf{b}_d \mathbf{b}_d^\top \right) \mathbf{x}_n \quad (10.37a)$$

$$= \left(\sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^\top \right) \mathbf{x}_n + \left(\sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top \right) \mathbf{x}_n, \quad (10.37b)$$

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 $\mathbf{x}_n - \tilde{\mathbf{x}}_n$, i.e., the difference vector between the original data point and its projection, is

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \left(\sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top \right) \mathbf{x}_n \quad (10.38a)$$

$$= \sum_{j=M+1}^D (\mathbf{x}_n^\top \mathbf{b}_j) \mathbf{b}_j. \quad (10.38b)$$

This means the difference is exactly the projection of the data point onto the orthogonal complement of the principal subspace: We identify the matrix $\sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top$ in (10.38a) as the projection matrix that performs this projection. Hence, the displacement vector $\mathbf{x}_n - \tilde{\mathbf{x}}_n$ lies in the subspace that is orthogonal to the principal subspace as illustrated in Figure 10.9.

Remark (Low-Rank Approximation). In (10.38a), we saw that the projection matrix, which projects \mathbf{x} onto $\tilde{\mathbf{x}}$, is given by

$$\sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^\top = \mathbf{B} \mathbf{B}^\top. \quad (10.39)$$

By construction as a sum of rank-one matrices $\mathbf{b}_m \mathbf{b}_m^\top$ we see that $\mathbf{B} \mathbf{B}^\top$ is

symmetric and has rank M . Therefore, the average squared reconstruction error can also be written as

$$\frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \frac{1}{N} \sum_{n=1}^N \left\| \mathbf{x}_n - \mathbf{B}\mathbf{B}^\top \mathbf{x}_n \right\|^2 \quad (10.40a)$$

$$= \frac{1}{N} \sum_{n=1}^N \left\| (\mathbf{I} - \mathbf{B}\mathbf{B}^\top) \mathbf{x}_n \right\|^2. \quad (10.40b)$$

PCA finds the best⁶⁰³⁹
rank- M
approximation of⁶⁰⁴⁰
the identity matrix.⁶⁰⁴¹
⁶⁰⁴²

Finding orthonormal basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_M$, which minimize the difference between the original data \mathbf{x}_n and their projections $\tilde{\mathbf{x}}_n$, is equivalent to finding the best rank- M approximation $\mathbf{B}\mathbf{B}^\top$ of the identity matrix \mathbf{I} (see Section 4.6). \diamond

Now, we have all the tools to reformulate the loss function (10.29).

$$J_M = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 \stackrel{(10.38b)}{=} \frac{1}{N} \sum_{n=1}^N \left\| \sum_{j=M+1}^D (\mathbf{b}_j^\top \mathbf{x}_n) \mathbf{b}_j \right\|^2. \quad (10.41)$$

We now explicitly compute the squared norm and exploit the fact that the \mathbf{b}_j form an ONB, which yields

$$J_M = \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D (\mathbf{b}_j^\top \mathbf{x}_n)^2 = \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D \mathbf{b}_j^\top \mathbf{x}_n \mathbf{b}_j^\top \mathbf{x}_n \quad (10.42a)$$

$$= \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D \mathbf{b}_j^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{b}_j, \quad (10.42b)$$

where we exploited the symmetry of the dot product in the last step to write $\mathbf{b}_j^\top \mathbf{x}_n = \mathbf{x}_n^\top \mathbf{b}_j$. We now swap the sums and obtain

$$J_M = \sum_{j=M+1}^D \mathbf{b}_j^\top \underbrace{\left(\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \right)}_{=: \mathbf{S}} \mathbf{b}_j = \sum_{j=M+1}^D \mathbf{b}_j^\top \mathbf{S} \mathbf{b}_j \quad (10.43a)$$

$$= \sum_{j=M+1}^D \text{tr}(\mathbf{b}_j^\top \mathbf{S} \mathbf{b}_j) \sum_{j=M+1}^D \text{tr}(\mathbf{S} \mathbf{b}_j \mathbf{b}_j^\top) = \text{tr} \left(\underbrace{\left(\sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^\top \right)}_{\text{projection matrix}} \mathbf{S} \right), \quad (10.43b)$$

where we exploited the property that the trace operator $\text{tr}(\cdot)$, 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 \mathbf{S} as the data covariance matrix. Since the projection matrix in (10.43b) is constructed as a sum of rank-one matrices $\mathbf{b}_j \mathbf{b}_j^\top$ it itself is of rank $D - M$.

Equation (10.43a) 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 in Section 10.2 and summarize the results from earlier in the light of the projection perspective.

The average squared reconstruction error, when projecting onto the M -dimensional principal subspace, is

$$J_M = \sum_{j=M+1}^D \lambda_j, \quad (10.44)$$

where λ_j are the eigenvalues of the data covariance matrix. Therefore, to minimize (10.44) 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 b_1, \dots, b_M that are associated with the largest M eigenvalues of the data covariance matrix.

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.
Minimizing the average squared reconstruction error is equivalent to maximizing the variance of the projected data.

Example 10.3 (MNIST Digits Embedding)

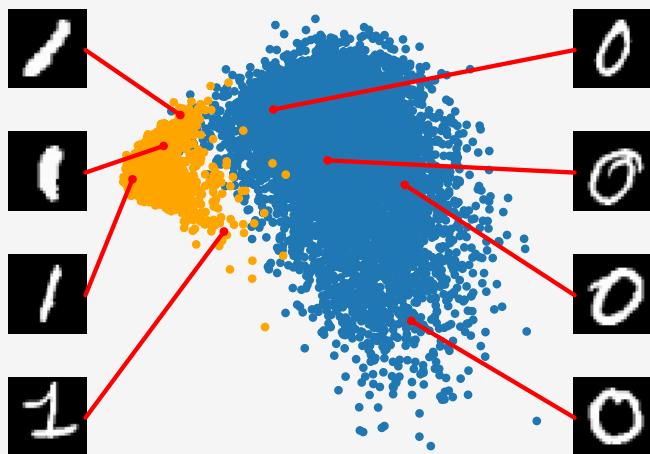


Figure 10.10
Embedding of MNIST digits 0 (blue) and 1 (orange) in a two-dimensional principal subspace using PCA. Four embeddings of the digits '0' and '1' in the principal subspace are highlighted in red with their corresponding original digit.

Figure 10.10 visualizes the training data of the MNIST digits ‘0’ and ‘1’ embedded in the vector subspace spanned by the first two principal components. We observe a relatively clear separation between ‘0’s (blue dots) and ‘1’s (orange dots), and we see the variation within each individual cluster. Four embeddings of the digits ‘0’ and ‘1’ in the principal subspace are highlighted in red with their corresponding original digit. The figure reveals that the variation within the set of ‘0’ is significantly greater than the variation within the set of ‘1’.

6068 10.4 Eigenvector Computation and Low-Rank Approximations

In the previous sections, we obtained the basis of the principal subspace as the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix

$$S = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top = \frac{1}{N} \mathbf{X} \mathbf{X}^\top, \quad (10.45)$$

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}. \quad (10.46)$$

6069 Note that \mathbf{X} is a $D \times N$ matrix, i.e., it is the transpose of the “typical”
 6070 data matrix (Bishop, 2006; Murphy, 2012). To get the eigenvalues (and
 Eigendecomposition 6071 the corresponding eigenvectors) of S , we can follow two approaches:
 or SVD to compute
 6072 eigenvectors.

- We perform an eigendecomposition (see Section 4.2) and compute the eigenvalues and eigenvectors of S directly.
- We use a singular value decomposition (see Section 4.5). Since S is symmetric and factorizes into $\mathbf{X} \mathbf{X}^\top$ (ignoring the factor $\frac{1}{N}$), the eigenvalues of S are the squared singular values of \mathbf{X} .

More specifically, the SVD of \mathbf{X} is given by

$$\underbrace{\mathbf{X}}_{D \times N} = \underbrace{\mathbf{U}}_{D \times D} \underbrace{\Sigma}_{D \times N} \underbrace{\mathbf{V}^\top}_{N \times N}, \quad (10.47)$$

where $\mathbf{U} \in \mathbb{R}^{D \times D}$ and $\mathbf{V}^\top \in \mathbb{R}^{N \times N}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{D \times N}$ is a matrix whose only non-zero entries are the singular values $\sigma_{ii} \geq 0$. It then follows that

$$S = \frac{1}{N} \mathbf{X} \mathbf{X}^\top = \frac{1}{N} \mathbf{U} \Sigma \underbrace{\mathbf{V}^\top \mathbf{V}}_{=I_N} \Sigma^\top \mathbf{U}^\top = \frac{1}{N} \mathbf{U} \Sigma \Sigma^\top \mathbf{U}^\top. \quad (10.48)$$

The columns of \mathbf{U}
 are the eigenvectors
 of S .

With the results from Section 4.5 we get that the columns of \mathbf{U} are the eigenvectors of $\mathbf{X} \mathbf{X}^\top$ (and therefore S). Furthermore, the eigenvalues λ_d of S are related to the singular values of \mathbf{X} via

$$\lambda_d = \frac{\sigma_d^2}{N}. \quad (10.49)$$

6077 This relationship between the eigenvalues of \mathbf{S} and the singular values
 6078 of \mathbf{X} provides the connection between the maximum variance view (Sec-
 6079 tion 10.2) and the singular value decomposition.

6080 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 \mathbf{U} in (10.48) to be the eigenvectors that are associated with the M largest eigenvalues of the data covariance matrix \mathbf{S} so that we identify \mathbf{U} as the projection matrix \mathbf{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

$$\tilde{\mathbf{X}}_M := \operatorname{argmin}_{\operatorname{rk}(\mathbf{A}) \leq M} \|\mathbf{X} - \mathbf{A}\|_2 \in \mathbb{R}^{D \times N} \quad (10.50)$$

of \mathbf{X} , where $\|\cdot\|_2$ is the spectral norm defined in (4.110). The Eckart-Young Theorem states that $\tilde{\mathbf{X}}_M$ is given by truncating the SVD at the top- M singular value. In other words, we obtain

$$\tilde{\mathbf{X}}_M = \underbrace{\mathbf{U}_M}_{D \times M} \underbrace{\Sigma_M}_{M \times M} \underbrace{\mathbf{V}_M^\top}_{M \times N} \in \mathbb{R}^{D \times N} \quad (10.51)$$

6081 with orthogonal matrices $\mathbf{U}_M := [\mathbf{u}_1, \dots, \mathbf{u}_M] \in \mathbb{R}^{D \times M}$ and $\mathbf{V}_M :=$
 6082 $[\mathbf{v}_1, \dots, \mathbf{v}_M] \in \mathbb{R}^{N \times M}$ and a diagonal matrix $\Sigma_M \in \mathbb{R}^{M \times M}$ whose diag-
 6083 onal entries are the M largest singular values of \mathbf{X} .

Eckart-Young
Theorem

6084 10.4.2 Practical Aspects

6085 Finding eigenvalues and eigenvectors is also important in other funda-
 6086 mental machine learning methods that require matrix decompositions. In
 6087 theory, as we discussed in Section 4.2, we can solve for the eigenvalues as
 6088 roots of the characteristic polynomial. However, for matrices larger than
 6089 4×4 this is not possible because we would need to find the roots of a poly-
 6090 nomial of degree 5 or higher. However, the Abel-Ruffini theorem (Ruffini,
 6091 1799; Abel, 1826) states that there exists no algebraic solution to this
 6092 problem for polynomials of degree 5 or more. Therefore, in practice, we
 6093 solve for eigenvalues or singular values using iterative methods, which are
 6094 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), then iterative processes,

power iteration

If S is invertible, it is sufficient to ensure that $x_0 \neq 0$.

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 x_0 that is not in the null space of S and follows the iteration

$$x_{k+1} = \frac{Sx_k}{\|Sx_k\|}, \quad k = 0, 1, \dots. \quad (10.52)$$

This means the vector x_k is multiplied by S in every iteration and then normalized, i.e., we always have $\|x_k\| = 1$. This sequence of vectors converges to the eigenvector associated with the largest eigenvalue of 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 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 $x_1, \dots, x_N, x_n \in \mathbb{R}^D$. Assuming the data is centered, the data covariance matrix is given as

$$S = \frac{1}{N} \mathbf{X} \mathbf{X}^\top \in \mathbb{R}^{D \times D}, \quad (10.53)$$

where $\mathbf{X} = [x_1, \dots, x_N]$ 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 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

$$Sb_m = \lambda_m b_m, \quad m = 1, \dots, M, \quad (10.54)$$

where b_m is a basis vector of the principal subspace. Let us re-write this

equation a bit: With \mathbf{S} defined in (10.53), we obtain

$$\mathbf{S}\mathbf{b}_m = \frac{1}{N} \mathbf{X} \mathbf{X}^\top \mathbf{b}_m = \lambda_m \mathbf{b}_m. \quad (10.55)$$

We now multiply $\mathbf{X}^\top \in \mathbb{R}^{N \times D}$ from the left-hand side, which yields

$$\frac{1}{N} \underbrace{\mathbf{X}^\top \mathbf{X}}_{\substack{N \times N \\ =: \mathbf{c}_m}} \mathbf{X}^\top \mathbf{b}_m = \lambda_m \mathbf{X}^\top \mathbf{b}_m \iff \frac{1}{N} \mathbf{X}^\top \mathbf{X} \mathbf{c}_m = \lambda_m \mathbf{c}_m, \quad (10.56)$$

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 $\mathbf{X} \mathbf{X}^\top$ equal the non-zero eigenvalues of $\mathbf{X}^\top \mathbf{X}$. We obtain the eigenvector of the matrix $\frac{1}{N} \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{N \times N}$ associated with λ_m as $\mathbf{c}_m := \mathbf{X}^\top \mathbf{b}_m$. Assuming we have no duplicate data points, this matrix has rank N and is invertible. This also implies that $\frac{1}{N} \mathbf{X}^\top \mathbf{X}$ has the same (non-zero) eigenvalues as the data covariance matrix \mathbf{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 $\frac{1}{N} \mathbf{X}^\top \mathbf{X}$, we are going to recover the original eigenvectors, which we still need for PCA. Currently, we know the eigenvectors of $\frac{1}{N} \mathbf{X}^\top \mathbf{X}$. If we left-multiply our eigenvalue/eigenvector equation with \mathbf{X} , we get

$$\underbrace{\frac{1}{N} \mathbf{X} \mathbf{X}^\top}_{\mathbf{S}} \mathbf{X} \mathbf{c}_m = \lambda_m \mathbf{X} \mathbf{c}_m \quad (10.57)$$

and we recover the data covariance matrix again. This now also means that we recover $\mathbf{X} \mathbf{c}_m$ as an eigenvector of \mathbf{S} .

Remark. If we want to apply the PCA algorithm that we discussed in Section 10.6 we need to normalize the eigenvectors $\mathbf{X} \mathbf{c}_m$ of \mathbf{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 10.11. We are given a two-dimensional dataset (Figure 10.11(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 μ of the dataset and subtracting it from every single data point. This ensures that the dataset has mean $\mathbf{0}$ (Figure 10.11(b)). Mean subtraction is not strictly necessary but reduces the risk of numerical problems.

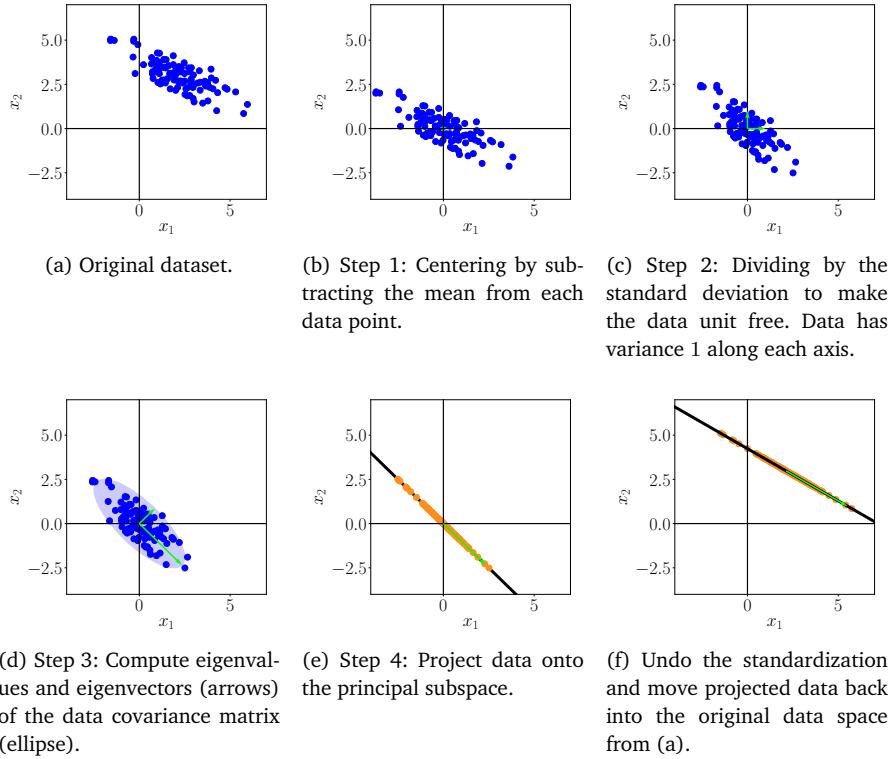


Figure 10.11 Steps of PCA.

- 6144 2 **Standardization** Divide the data points by the standard deviation σ_d
 6145 of the dataset for every dimension $d = 1, \dots, D$. Now the data is unit
 6146 free, and it has variance 1 along each axis, which is indicated by the
 6147 two arrows in Figure 10.11(c). This step completes the *standardization*
 6148 of the data.
- 6149 3 **Eigendecomposition of the covariance matrix** Compute the data
 6150 covariance matrix and its eigenvalues and corresponding eigenvectors.
 6151 Since the covariance matrix is symmetric, the eigenvectors form an or-
 6152 thogonal basis. In Figure 10.11(d), the eigenvectors are scaled by the
 6153 magnitude of the corresponding eigenvalue. The longer vector spans
 6154 the principal subspace, which we denote by U . The data covariance
 6155 matrix is represented by the ellipse.
- 6156 4 **Projection** We can project any data point $x_* \in \mathbb{R}^D$ onto the principal
 6157 subspace: To get this right, we need to standardize x_* using the mean
 6158 μ_d and standard deviation σ_d of the training data in the d th dimension,
 6159 respectively, so that

$$x_*^{(d)} \leftarrow \frac{x_*^{(d)} - \mu_d}{\sigma_d}, \quad d = 1, \dots, D, \quad (10.58)$$

where $x_*^{(d)}$ is the d th component of \mathbf{x}_* . We obtain the projection as

$$\tilde{\mathbf{x}}_* = \mathbf{B}\mathbf{B}^\top \mathbf{x}_* \quad (10.59)$$

with coordinates

$$\mathbf{z}_* = \mathbf{B}^\top \mathbf{x}_* \quad (10.60)$$

with respect to the basis of the principal subspace. Here, \mathbf{B} is the matrix that contains the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix as columns. PCA returns the coordinates (10.60), not the projections \mathbf{x}_* .

Having standardized our dataset, (10.59) only yields the projections in the context of the standardized dataset. To obtain our projection in the original data space (i.e., before standardization), we need to undo the standardization (10.58) and multiply by the standard deviation before adding the mean so that we obtain

$$\tilde{x}_*^{(d)} \leftarrow \tilde{x}_*^{(d)}\sigma_d + \mu_d, \quad d = 1, \dots, D. \quad (10.61)$$

Figure 10.11(f) illustrates the projection in the original data space.

Example 10.4 (MNIST Digits: Reconstruction)

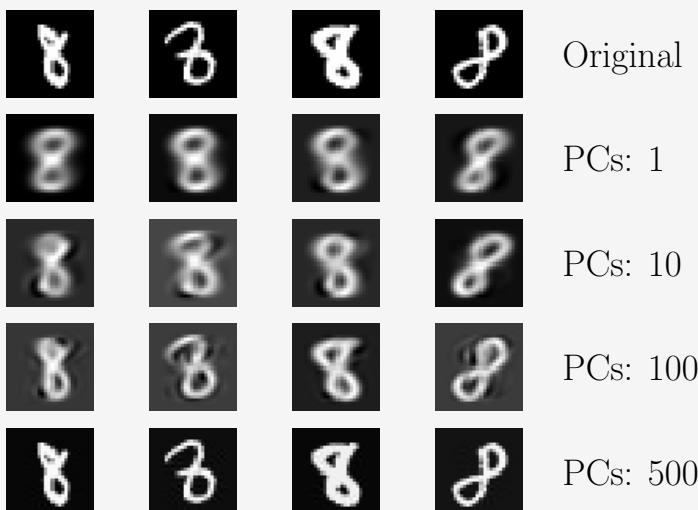


Figure 10.12 Effect of increasing number of principal components on reconstruction.

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

<http://yann.lecun.com/exdb/mnist/>

every image in this dataset as a vector $\mathbf{x} \in \mathbb{R}^{784}$. 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 10.12. The first row of Figure 10.12 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 see that even with a single-dimensional principal subspace we get a half-way 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 are 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.

Figure 10.13
Average squared reconstruction error as a function of the number of principal components. The average squared reconstruction error is the sum of the eigenvalues in the orthogonal complement of the principal subspace.

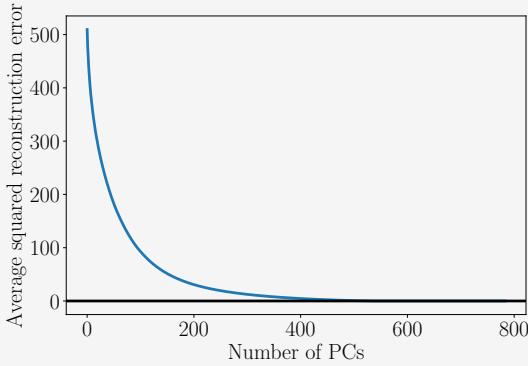


Figure 10.13 shows the average squared reconstruction error, which is

$$\frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \sum_{i=M+1}^D \lambda_i, \quad (10.62)$$

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. This matches exactly our observation in Figure 10.5 where we discovered that the most of the variance of the projected data is captured by only a few principal components. 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).

6161 10.7 Latent Variable Perspective

6162 In the previous sections, we derived PCA without any notion of a prob-
 6163 abilistic model using the maximum-variance and the projection perspec-
 6164 tives. On the one hand this approach may be appealing as it allows us to
 6165 sidestep all the mathematical difficulties that come with probability the-
 6166 ory, but on the other hand a probabilistic model would offer us more flex-
 6167 ibility and useful insights. More specifically, a probabilistic model would

- 6168 • come with a likelihood function, and we can explicitly deal with noisy
 6169 observations (which we did not even discuss earlier),
- 6170 • allow us to do Bayesian model comparison via the marginal likelihood
 6171 as discussed in Section 8.5,
- 6172 • view PCA as a generative model, which allows us to simulate new data,
 6173 • allow us to make straightforward connections to related algorithms
 6174 • deal with data dimensions that are missing at random by applying
 6175 Bayes' theorem,
- 6176 • give us a notion of the novelty of a new data point,
- 6177 • give us a principled way to extend the model, e.g., to a mixture of PCA
 6178 models,
- 6179 • have the PCA we derived in earlier sections as a special case,
- 6180 • allow for a fully Bayesian treatment by marginalizing out the model
 6181 parameters.

6182 By introducing a continuous-valued latent variable $\mathbf{z} \in \mathbb{R}^M$ it is possible
 6183 to phrase PCA as a probabilistic latent-variable model. Tipping and Bishop
 6184 (1999) proposed this latent-variable model as *Probabilistic PCA* (PPCA).
 6185 PPCA addresses most of the issues above, and the PCA solution that we
 6186 obtained by maximizing the variance in the projected space or by minimiz-
 6187 ing the reconstruction error is obtained as the special case of maximum
 6188 likelihood estimation in a noise-free setting.

Probabilistic PCA

6189 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 $\mathbf{z} \in \mathbb{R}^M$ with a standard-Normal prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and a linear relationship between the latent variables and the observed \mathbf{x} data where

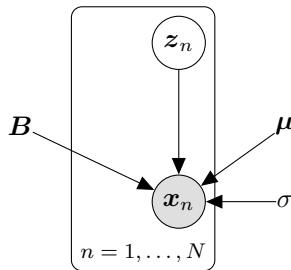
$$\mathbf{x} = \mathbf{B}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \in \mathbb{R}^D, \quad (10.63)$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ is Gaussian observation noise, $\mathbf{B} \in \mathbb{R}^{D \times M}$ 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

$$p(\mathbf{x} | \mathbf{z}, \mathbf{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}(\mathbf{x} | \mathbf{B}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}). \quad (10.64)$$

Figure 10.14

Graphical model for probabilistic PCA. The observations x_n explicitly depend on corresponding latent variables $z_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The model parameters B, μ and the likelihood parameter σ are shared across the dataset.



Overall, PPCA induces the following generative process:

$$z_n \sim \mathcal{N}(z | \mathbf{0}, \mathbf{I}) \quad (10.65)$$

$$x_n | z_n \sim \mathcal{N}(x | Bz_n + \mu, \sigma^2 \mathbf{I}) \quad (10.66)$$

6190 To generate a data point that is typical given the model parameters, we
6191 ancestral sampling follow an *ancestral sampling* scheme: We first sample a latent variable z_n
6192 from $p(z)$. Then, we use z_n in (10.64) to sample a data point conditioned
6193 on the sampled z_n , i.e., $x_n \sim p(x | z_n, B, \mu, \sigma^2)$.

This generative process allows us to write down the probabilistic model (i.e., the joint distribution of all random variables, see Section 8.3) as

$$p(x, z | B, \mu, \sigma^2) = p(x | z, B, \mu, \sigma^2)p(z), \quad (10.67)$$

6194 which immediately gives rise to the graphical model in Figure 10.14 using
6195 the results from Section 8.4.

6196 *Remark.* Note the direction of the arrow that connects the latent variables
6197 z and the observed data x : The arrow points from z to x , which means
6198 that the PPCA model assumes a lower-dimensional latent cause z for high-
6199 dimensional observations x . In the end, we are obviously interested in
6200 finding something out about z given some observations. To get there we
6201 will apply Bayesian inference to “invert” the arrow implicitly and go from
6202 observations to latent variables. ◇

Example 10.5 (Generating New Data using Latent Variables)

Figure 10.15 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 z_* in this latent space and generate an image $\tilde{x}_* = Bz_*$ 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 and more artefacts, e.g., the top-left and top-right digits. Note that the intrinsic dimensionality of these generated images is only two.

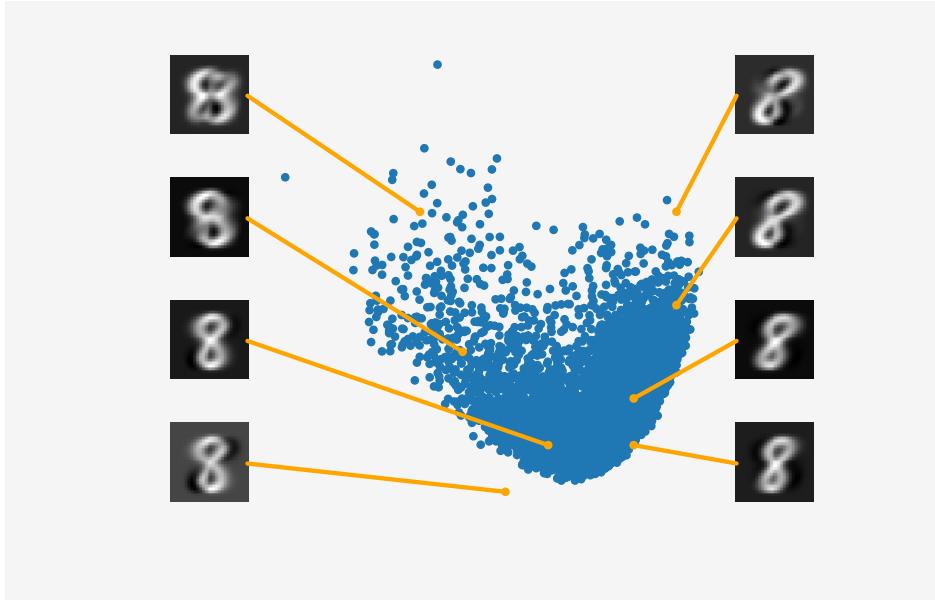


Figure 10.15
Generating new MNIST digits. The latent variables z can be used to generate new data $\tilde{x} = Bz$. The closer we stay to the training data the more realistic the generated data.

6203

10.7.2 Likelihood and Joint Distribution

Using the results from Chapter 6, we obtain the likelihood of this probabilistic model by integrating out the latent variable z (see Section 8.3.4) so that

$$\begin{aligned} p(\mathbf{x} | \mathbf{B}, \boldsymbol{\mu}, \sigma^2) &= \int p(\mathbf{x} | \mathbf{z}, \boldsymbol{\mu}, \sigma^2) p(\mathbf{z}) d\mathbf{z} \\ &= \int \mathcal{N}(\mathbf{x} | \mathbf{Bz} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}) \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}) d\mathbf{z}. \end{aligned} \quad (10.68)$$

The likelihood does not depend on the latent variables z .

From Section 6.5, we know that the solution to this integral is a Gaussian distribution with mean

$$\mathbb{E}_{\mathbf{x}}[\mathbf{x}] = \mathbb{E}_{\mathbf{z}}[\mathbf{Bz} + \boldsymbol{\mu}] + \mathbb{E}_{\epsilon}[\boldsymbol{\epsilon}] = \boldsymbol{\mu} \quad (10.69)$$

and with covariance matrix

$$\begin{aligned} \mathbb{V}[\mathbf{x}] &= \mathbb{V}_{\mathbf{z}}[\mathbf{Bz} + \boldsymbol{\mu}] + \mathbb{V}_{\epsilon}[\boldsymbol{\epsilon}] = \mathbb{V}_{\mathbf{z}}[\mathbf{Bz}] + \sigma^2 \mathbf{I} \\ &= \mathbf{B} \mathbb{V}_{\mathbf{z}}[\mathbf{z}] \mathbf{B}^T + \sigma^2 \mathbf{I} = \mathbf{B} \mathbf{B}^T + \sigma^2 \mathbf{I}. \end{aligned} \quad (10.70)$$

6204 The likelihood in (10.68) can be used for maximum likelihood or MAP
6205 estimation of the model parameters.

6206 *Remark.* We cannot use the conditional distribution in (10.64) for maxi-
6207 mum likelihood estimation as it still depends on the latent variables. The
6208 likelihood function we require for maximum likelihood (or MAP) estima-
6209 tion should only be a function of the data \mathbf{x} and the model parameters,
6210 but must not depend on the latent variables. ◇

From Section 6.5 we know that a Gaussian random variable \mathbf{z} and a linear/affine transformation $\mathbf{x} = \mathbf{B}\mathbf{z}$ of it are jointly Gaussian distributed. We already know the marginals $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$ and $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I})$. The missing cross-covariance is given as

$$\text{Cov}[\mathbf{x}, \mathbf{z}] = \text{Cov}_{\mathbf{z}}[\mathbf{B}\mathbf{z} + \boldsymbol{\mu}] = \mathbf{B} \text{Cov}_{\mathbf{z}}[\mathbf{z}, \mathbf{z}] = \mathbf{B}. \quad (10.71)$$

Therefore, the probabilistic model of PPCA, i.e., the joint distribution of latent and observed random variables is explicitly given by

$$p(\mathbf{x}, \mathbf{z} | \mathbf{B}, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{I} \end{bmatrix}\right), \quad (10.72)$$

- 6211 with a mean vector of length $D + M$ and a covariance matrix of size
- 6212 $(D + M) \times (D + M)$.

6213 10.7.3 Posterior Distribution

The joint Gaussian distribution $p(\mathbf{x}, \mathbf{z} | \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$ in (10.72) allows us to determine the posterior distribution $p(\mathbf{z} | \mathbf{x})$ immediately by applying the rules of Gaussian conditioning from Section 6.5.1. The posterior distribution of the latent variable given an observation \mathbf{x} is then

$$p(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\mathbf{z} | \mathbf{m}, \mathbf{C}), \quad (10.73a)$$

$$\mathbf{m} = \mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I})^{-1}(\mathbf{x} - \boldsymbol{\mu}), \quad (10.73b)$$

$$\mathbf{C} = \mathbf{I} - \mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top + \sigma^2\mathbf{I})^{-1}\mathbf{B}. \quad (10.73c)$$

- 6214 Note that the posterior covariance does not depend on the observed data \mathbf{x} . For a new observation \mathbf{x}_* in data space, we use (10.73) to determine
- 6215 the posterior distribution of the corresponding latent variable \mathbf{z}_* . The co-
- 6216 variance matrix \mathbf{C} allows us to assess how confident the embedding is. A
- 6217 covariance matrix \mathbf{C} with a small determinant (which measures volumes)
- 6218 tells us that the latent embedding \mathbf{z}_* is fairly certain. If we obtain a pos-
- 6219 terior distribution $p(\mathbf{z}_* | \mathbf{x}_*)$ with much variance, we may be faced with
- 6220 an outlier. However, we can explore this posterior distribution to under-
- 6221 stand what other data points \mathbf{x} are plausible under this posterior. To do
- 6222 this, we exploit the generative process underlying PPCA, which allows us
- 6223 to explore the posterior distribution on the latent variables by generating
- 6224 new data that are plausible under this posterior:

- 6226 1 Sample a latent variable $\mathbf{z}_* \sim p(\mathbf{z} | \mathbf{x}_*)$ from the posterior distribution
- 6227 over the latent variables (10.73)
- 6228 2 Sample a reconstructed vector $\tilde{\mathbf{x}}_* \sim p(\mathbf{x} | \mathbf{z}_*, \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$ from (10.64)

- 6229 If we repeat this process many times, we can explore the posterior dis-
- 6230 tribution (10.73) on the latent variables \mathbf{z}_* and its implications on the
- 6231 observed data. The sampling process effectively hypothesizes data, which
- 6232 is plausible under the posterior distribution.

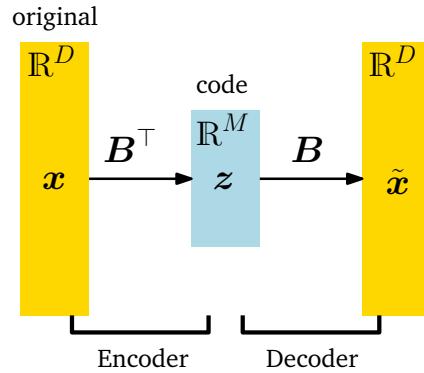


Figure 10.16 PCA can be viewed as a linear auto-encoder. It encodes the high-dimensional data x into a lower-dimensional representation (code) $z \in \mathbb{R}^M$ and decodes z using a decoder. The decoded vector \tilde{x} is the orthogonal projection of the original data x onto the M -dimensional principal subspace.

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 $x \in \mathbb{R}^D$ and used a matrix B^\top to find a lower-dimensional representation $z \in \mathbb{R}^M$. The columns of B are the eigenvectors of the data covariance matrix S that are associated with the largest eigenvalues. Once we have a low-dimensional representation z , we can get a high-dimensional version of it (in the original data space) as $x \approx \tilde{x} = Bz = BB^\top x \in \mathbb{R}^D$, where BB^\top is a projection matrix.

We can also think of PCA as a linear *auto-encoder* as illustrated in Figure 10.16. An auto-encoder encodes the data $x_n \in \mathbb{R}^D$ to a *code* $z_n \in \mathbb{R}^M$ and decodes it to a \tilde{x}_n similar to x_n . The mapping from the data to the code is called the *encoder*, and 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 $z_n = B^\top x_n \in \mathbb{R}^M$ and we are interested in minimizing the average squared error between the data x_n and its reconstruction $\tilde{x}_n = Bz_n$, $n = 1, \dots, N$, we obtain

$$\frac{1}{N} \sum_{n=1}^N \|x_n - \tilde{x}_n\|^2 = \frac{1}{N} \sum_{n=1}^N \|x_n - B^\top B x_n\|^2. \quad (10.74)$$

This means we end up with the same objective function as in (10.29) 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 is also known as a or , whereas the decoder is also called a .

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

auto-encoder
code

encoder
decoder

recognition network
inference network
generator

The code is a compressed version²⁵⁷ of the original data.⁶²⁵⁸

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., $\mathbf{B}, \boldsymbol{\mu}$ and the likelihood parameter σ^2 are known. Tipping and Bishop (1999) describe how to derive maximum likelihood estimates for these parameters 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 $\mathbf{\Lambda} - \sigma^2 \mathbf{I}$ 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 .

by

$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n, \quad (10.75)$$

$$\mathbf{B}_{\text{ML}} = \mathbf{T}(\mathbf{\Lambda} - \sigma^2 \mathbf{I})^{\frac{1}{2}} \mathbf{R}, \quad (10.76)$$

$$\sigma_{\text{ML}}^2 = \frac{1}{D-M} \sum_{j=M+1}^D \lambda_j, \quad (10.77)$$

where $\mathbf{T} \in \mathbb{R}^{D \times M}$ contains M eigenvectors of the data covariance matrix, $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_M) \in \mathbb{R}^{M \times M}$ is a diagonal matrix with the eigenvalues associated with the principal axes on its diagonal, and $\mathbf{R} \in \mathbb{R}^{M \times M}$ is an arbitrary orthogonal matrix. The maximum likelihood solution \mathbf{B}_{ML} is unique up to an arbitrary orthogonal transformation, e.g., we can right-multiply \mathbf{B}_{ML} with any rotation matrix \mathbf{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 $\sigma \rightarrow 0$, PPCA and PCA provide identical solutions: Since the data covariance matrix \mathbf{S} is symmetric, it can be diagonalized (see Section 4.4), i.e., there exists a matrix \mathbf{T} of eigenvectors of \mathbf{S} so that

$$\mathbf{S} = \mathbf{T} \mathbf{\Lambda} \mathbf{T}^{-1}. \quad (10.78)$$

In the PPCA model, the data covariance matrix is the covariance matrix of the Gaussian likelihood $p(\mathbf{x} | \mathbf{B}, \boldsymbol{\mu}, \sigma^2)$, which is $\mathbf{B}\mathbf{B}^\top + \sigma^2 \mathbf{I}$, see (10.70).

For $\sigma \rightarrow 0$, we obtain $\mathbf{B}\mathbf{B}^\top$ so that this data covariance must equal the PCA data covariance (and its factorization given in (10.78)) so that

$$\text{Cov}[\mathcal{X}] = \mathbf{T}\Lambda\mathbf{T}^{-1} = \mathbf{B}\mathbf{B}^\top \iff \mathbf{B} = \mathbf{T}\Lambda^{\frac{1}{2}}\mathbf{R}, \quad (10.79)$$

i.e., we obtain 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 (nested) cross validation (Section 8.5.1) or Bayesian model selection criteria (discussed in Section 8.5.2) to determine a good estimate of the intrinsic dimensionality of the data (Minka, 2001b).

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. By doing so, we a) avoid point estimates of the parameters and the issues that come with these point estimates (see Section 8.5) and b) allow for an automatic selection of the appropriate dimensionality M of the latent space. In this *Bayesian PCA*, which was proposed by Bishop (1999), a prior $p(\boldsymbol{\mu}, \mathbf{B}, \sigma^2)$ is placed 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 $p(\mathbf{x}_n | \mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n | \mathbf{B}\mathbf{z}_n + \boldsymbol{\mu}, \sigma^2\mathbf{I})$ with prior $p(\mathbf{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, where 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 \mathbf{B} , $\boldsymbol{\mu}$. However, FA no longer allows for a closed-form maximum likelihood solution so that we need to use an iterative scheme, such as the Expectation Maximization 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.

factor analysis

An overly flexible likelihood would be able to explain more than just the noise.

Independent Component Analysis 6316
 An algorithm that is also closely related to PCA is *Independent Component Analysis* (ICA, (Hyvarinen et al., 2001)) Starting again with the latent-variable perspective $p(\mathbf{x}_n | \mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n | \mathbf{B}\mathbf{z}_n + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$ we now change the prior on \mathbf{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 $p(\mathbf{z})$ on the latent sources to require non-Gaussian priors $p(\mathbf{z})$. We refer to the books by Hyvarinen et al. (2001) and Murphy (2012) for more details on ICA.

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.

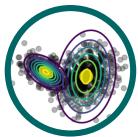
The (P)PCA model we discussed here allows for several important extensions. In Section 10.5, we explained how to do PCA when the input dimensionality D is significantly greater than the number N of data points. By exploiting the insight that PCA can be performed by computing (many) inner products, this 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).

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 \mathbf{z} and the observations \mathbf{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 \mathbf{z} . Similar to Bayesian PCA, the *Bayesian GP-LVM* proposed by Titsias and Lawrence (2010) maintains a distribution on the latent variables \mathbf{z} and uses approximate inference to integrate them out as well.

11

6265

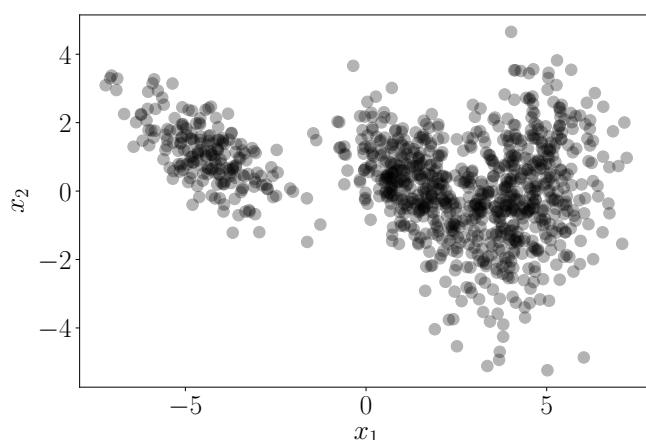
Density Estimation with Gaussian Mixture Models



6266 In earlier chapters, we covered already two fundamental problems in
6267 machine learning: regression (Chapter 9) and dimensionality reduction
6268 (Chapter 10). In this chapter, we will have a look at a third pillar of ma-
6269 chine learning: density estimation. On our journey, we introduce impor-
6270 tant concepts, such as the Expectation Maximization (EM) algorithm and
6271 a latent variable perspective of density estimation with mixture models.

6272 When we apply machine learning to data we often aim to represent
6273 data in some way. A straightforward way is to take the data points them-
6274 selves as the representation of the data, see Figure 11.1 for an example.
6275 However, this approach may be unhelpful if the dataset is huge or if we
6276 are interested in representing characteristics of the data. In density esti-
6277 mation, we represent the data compactly using a density from a paramet-
6278 ric family, e.g., a Gaussian or Beta distribution. For example, we may be
6279 looking for the mean and variance of a dataset in order to represent the
6280 data compactly using a Gaussian distribution. The mean and variance can
6281 be found using tools we discussed in Section 8.2: maximum likelihood or
6282 maximum-a-posteriori estimation. We can then use the mean and variance
6283 of this Gaussian to represent the distribution underlying the data, i.e., we
6284 think of the dataset to be a typical realization from this distribution if we
6285 were to sample from it.

Figure 11.1
Two-dimensional
dataset that cannot
be meaningfully
represented by a
Gaussian.



In practice, the Gaussian (or similarly all other distributions we encountered 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 can be used to describe a distribution $p(\mathbf{x})$ by a convex combination of K simple (base) distributions

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k p_k(\mathbf{x}) \quad (11.1)$$

$$0 \leq \pi_k \leq 1, \quad \sum_{k=1}^K \pi_k = 1, \quad (11.2)$$

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.

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

A *Gaussian mixture model* is a density model where we combine a finite number of K Gaussian distributions $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ so that

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (11.3)$$

$$0 \leq \pi_k \leq 1, \quad \sum_{k=1}^K \pi_k = 1, \quad (11.4)$$

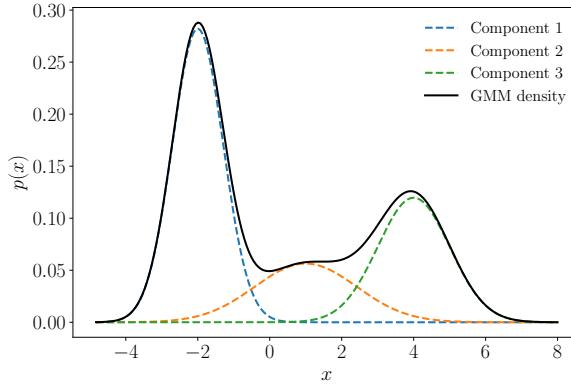
where we defined $\boldsymbol{\theta} := \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k : k = 1, \dots, K\}$ as the collection of all parameters of the model. This convex combination of Gaussian distribution gives us significantly more flexibility for modeling complex densities than a simple Gaussian distribution (which we recover from (11.3) for $K = 1$). An illustration is given in Figure 11.2, displaying the weighted

mixture models

mixture weights

Gaussian mixture model

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.



components and the mixture density, which is given as

$$p(x | \theta) = 0.5\mathcal{N}(x | -2, \frac{1}{2}) + 0.2\mathcal{N}(x | 1, 2) + 0.3\mathcal{N}(x | 4, 1). \quad (11.5)$$

6307

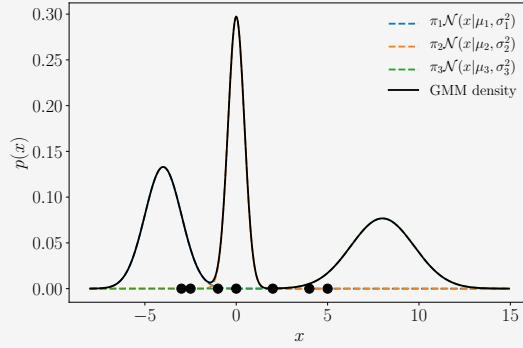
6308

11.2 Parameter Learning via Maximum Likelihood

6309 Assume we are given a dataset $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ where $\mathbf{x}_n, n = 1, \dots, N$
 6310 are drawn i.i.d. from an unknown distribution $p(\mathbf{x})$. Our objective is to
 6311 find a good approximation/representation of this unknown distribution
 6312 $p(\mathbf{x})$ by means of a Gaussian mixture model (GMM) with K mixture com-
 6313 ponents. The parameters of the GMM are the K means μ_k , the covariances
 6314 Σ_k and mixture weights π_k . We summarize all these free parameters in
 6315 $\theta := \{\pi_k, \mu_k, \Sigma_k : k = 1, \dots, K\}$.

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 consider a one-dimensional dataset $\mathcal{X} = \{-3, -2.5, -1, 0, 2, 4, 5\}$ consisting of seven data points and wish to find a GMM with $K = 3$ components that models the density of the data. We initialize the mixture components as

$$p_1(x) = \mathcal{N}(x | -4, 1) \quad (11.6)$$

$$p_2(x) = \mathcal{N}(x | 0, 0.2) \quad (11.7)$$

$$p_3(x) = \mathcal{N}(x | 8, 3) \quad (11.8)$$

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 predictive distribution of the training data given the parameters. We exploit our i.i.d. assumption, which leads to the factorized likelihood

$$p(\mathcal{X} | \boldsymbol{\theta}) = \prod_{n=1}^N p(\mathbf{x}_n | \boldsymbol{\theta}), \quad p(\mathbf{x}_n | \boldsymbol{\theta}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (11.9)$$

where every individual likelihood term $p(\mathbf{x}_n | \boldsymbol{\theta})$ is a Gaussian mixture density. Then, we obtain the log-likelihood as

$$\log p(\mathcal{X} | \boldsymbol{\theta}) = \sum_{n=1}^N \log p(\mathbf{x}_n | \boldsymbol{\theta}) = \underbrace{\sum_{n=1}^N \log \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{=: \mathcal{L}}. \quad (11.10)$$

6316 We aim to find parameters $\boldsymbol{\theta}_{\text{ML}}^*$ that maximize the log-likelihood \mathcal{L} defined
6317 in (11.10). Our “normal” procedure would be to compute the gradient
6318 $d\mathcal{L}/d\boldsymbol{\theta}$ of the log-likelihood with respect to the model parameters $\boldsymbol{\theta}$, set
6319 it to $\mathbf{0}$ and solve for $\boldsymbol{\theta}$. However, unlike our previous examples for maxi-
6320 mum likelihood estimation (e.g., when we discussed linear regression in
6321 Section 9.2), we cannot obtain a closed-form solution. However, we can
6322 exploit an iterative scheme to find good model parameters $\boldsymbol{\theta}_{\text{ML}}$, which will
6323 out to be the EM algorithm for Gaussian mixture models. The key idea is
6324 to update one model parameter at a time while keeping the others fixed.

Remark. 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

$$\log \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{D}{2} \log(2\pi) - \frac{1}{2} \log \det(\boldsymbol{\Sigma}) - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}). \quad (11.11)$$

6325 This simple form allows us find closed-form maximum likelihood esti-
6326 mates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, as discussed in Chapter 8. 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. \diamond

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 $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \mathbf{0}^\top \iff \sum_{n=1}^N \frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \mathbf{0}^\top, \quad (11.12)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \mathbf{0} \iff \sum_{n=1}^N \frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k} = \mathbf{0}, \quad (11.13)$$

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = 0 \iff \sum_{n=1}^N \frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \pi_k} = 0. \quad (11.14)$$

For all three necessary conditions, by applying the chain rule (see Section 5.2.2), we require partial derivatives of the form

$$\frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{p(\mathbf{x}_n | \boldsymbol{\theta})} \frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \quad (11.15)$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k, k = 1, \dots, K\}$ are the model parameters and

$$\frac{1}{p(\mathbf{x}_n | \boldsymbol{\theta})} = \frac{1}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \quad (11.16)$$

In the following, we will compute the partial derivatives (11.12)–(11.14). But before we do this, we introduce a quantity that will play a central role in the remainder of this chapter: responsibilities.

6332

11.2.1 Responsibilities

We define the quantity

$$r_{nk} := \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \quad (11.17)$$

responsibility

as the *responsibility* of the k th mixture component for the n th data point. The responsibility r_{nk} of the k th mixture component for data point \mathbf{x}_n is proportional to the likelihood

$$p(\mathbf{x}_n | \pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (11.18)$$

6333

of the mixture component given the data point. Therefore, mixture components have a high responsibility for a data point when the data point could be a plausible sample from that mixture component. Note that $\mathbf{r}_n := [r_{n1}, \dots, r_{nK}]^\top \in \mathbb{R}^K$ is a (normalized) probability vector, i.e.,

\mathbf{r}_n follows a Boltzmann/Gibbs distribution.

6334

6335

6336

6337 $\sum_k r_{nk} = 1$ with $r_{nk} \geq 0$. This probability vector distributes probability mass among the K mixture components, and we can think of r_n as a “soft assignment” of x_n to the K mixture components. Therefore, the responsibility r_{nk} from (11.17) represents the probability that x_n has been generated by the k th mixture component.

The responsibility r_{nk} is the probability that the k th mixture component generated the n th data point.

Example 11.2 (Responsibilities)

For our example from Figure 11.3 we compute the responsibilities r_{nk}

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.057 & 0.943 & 0.0 \\ 0.001 & 0.999 & 0.0 \\ 0.0 & 0.066 & 0.934 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \in \mathbb{R}^{N \times K}. \quad (11.19)$$

Here, the n th 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 k th column gives us an overview of the responsibility of the k th 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 k th mixture component. In our example, we get $N_1 = 2.058$, $N_2 = 2.008$, $N_3 = 2.934$.

6342 In the following, we determine the updates of the model parameters 6343 μ_k, Σ_k, π_k for given responsibilities. We will see that the update equations 6344 all depend on the responsibilities, which makes a closed-form solution to the maximum likelihood estimation problem impossible. However, 6345 for given responsibilities we will be updating one model parameter at a time, while keeping the others fixed. After this, we will re-compute the 6346 responsibilities. Iterating these two steps will eventually converge to a local optimum and is a specific instantiation of the EM algorithm. We will 6347 discuss this in some more detail in Section 11.3.

6351 11.2.2 Updating the Means

Theorem 11.1 (Update of the GMM Means). *The update of the mean parameters μ_k , $k = 1, \dots, K$, of the GMM is given by*

$$\mu_k^{new} = \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}}, \quad (11.20)$$

6352 where the responsibilities r_{nk} are defined in (11.17).

6353 *Remark.* The update of the means $\boldsymbol{\mu}_k$ of the individual mixture components
 6354 in (11.20) depends on all means, covariance matrices $\boldsymbol{\Sigma}_k$ and mixture weights π_k via r_{nk} given in (11.17). Therefore, we cannot obtain a
 6355 closed-form solution for all $\boldsymbol{\mu}_k$ at once. \diamond
 6356

Proof From (11.15), we see that the gradient of the log-likelihood with respect to the mean parameters $\boldsymbol{\mu}_k$, $k = 1, \dots, K$, requires us to compute the partial derivative

$$\frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \sum_{j=1}^K \pi_j \frac{\partial \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\partial \boldsymbol{\mu}_k} = \pi_k \frac{\partial \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} \quad (11.21)$$

$$= \pi_k (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (11.22)$$

6357 where we exploited that only the k th mixture component depends on $\boldsymbol{\mu}_k$.

We use our result from (11.22) 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

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^N \frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^N \frac{1}{p(\mathbf{x}_n | \boldsymbol{\theta})} \frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} \quad (11.23a)$$

$$= \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{=r_{nk}} \quad (11.23b)$$

$$= \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}. \quad (11.23c)$$

6358 Here, we used the identity from (11.16) and the result of the partial
 6359 derivative in (11.22) to get to (11.23b). The values r_{nk} are the responsibilities we defined in (11.17).

6360 We now solve (11.23c) for $\boldsymbol{\mu}_k^{\text{new}}$ so that $\frac{\partial \mathcal{L}(\boldsymbol{\mu}_k^{\text{new}})}{\partial \boldsymbol{\mu}_k} = \mathbf{0}^\top$ and obtain

$$\sum_{n=1}^N r_{nk} \mathbf{x}_n = \sum_{n=1}^N r_{nk} \boldsymbol{\mu}_k^{\text{new}} \iff \boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}} = \frac{1}{N_k} \sum_{n=1}^N r_{nk} \mathbf{x}_n, \quad (11.24)$$

where we defined

$$N_k := \sum_{n=1}^N r_{nk} \quad (11.25)$$

6361 as the total responsibility of the k th mixture component for the entire
 6362 dataset. This concludes the proof of Theorem 11.1. \square

Intuitively, (11.20) can be interpreted as an importance-weighted Monte-Carlo estimate of the mean, where the importance weights of data point \mathbf{x}_n are the responsibilities r_{nk} of the k th cluster for \mathbf{x}_n , $k = 1, \dots, K$.

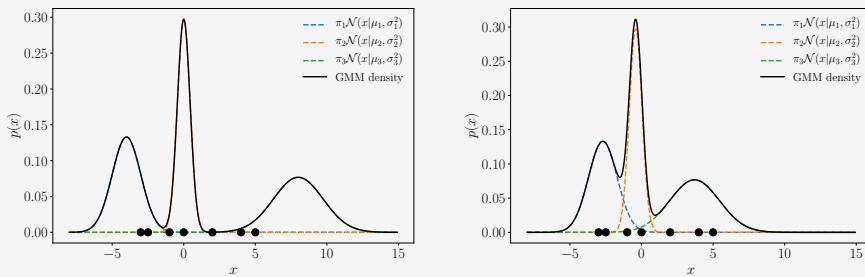
Therefore, the mean μ_k is pulled toward a data point x_n with strength given by r_{nk} . The means are pulled stronger toward data points for which the corresponding mixture component has a high responsibility, i.e., a high likelihood. Figure 11.4 illustrates this. We can also interpret the mean update in (11.20) as the expected value of all data points under the distribution given by

$$\mathbf{r}_k := [r_{1k}, \dots, r_{Nk}]^\top / N_k, \quad (11.26)$$

which is a normalized probability vector, i.e.,

$$\boldsymbol{\mu}_k \leftarrow \mathbb{E}_{\mathbf{r}_k}[\mathcal{X}]. \quad (11.27)$$

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.

In our example from Figure 11.3, the mean values are updated as follows:

$$\mu_1 : -4 \rightarrow -2.7 \quad (11.28)$$

$$\mu_2 : 0 \rightarrow -0.4 \quad (11.29)$$

$$\mu_3 : 8 \rightarrow 3.7 \quad (11.30)$$

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 .

6363 The update of the mean parameters in (11.20) look fairly straight-
 6364 forward. However, note that the responsibilities r_{nk} are a function of
 6365 $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ for all $j = 1, \dots, K$, such that the updates in (11.20) depend
 6366 on all parameters of the GMM, and a closed-form solution, which we ob-
 6367 tained for linear regression in Section 9.2 or PCA in Chapter 10, cannot
 6368 be obtained.

Figure 11.4 Update of the mean parameter of mixture component in a GMM. The mean μ is being pulled toward individual data points with the weights given by the corresponding responsibilities.

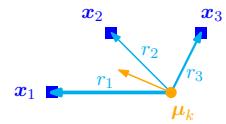


Figure 11.5 Effect of updating the mean values in a GMM. (a) GMM before updating the mean values; (b) GMM after updating the mean values μ_k while retaining the variances and mixture weights.

6369

11.2.3 Updating the Covariances

Theorem 11.2 (Updates of the GMM Covariances). *The update of the covariance parameters Σ_k , $k = 1, \dots, K$ of the GMM is given by*

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top, \quad (11.31)$$

6370 where r_{nk} and N_k are defined in (11.17) and (11.25), 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 Σ_k , set them to $\mathbf{0}$ and solve for Σ_k . We start with our general approach

$$\frac{\partial \mathcal{L}}{\partial \Sigma_k} = \sum_{n=1}^N \frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \Sigma_k} = \sum_{n=1}^N \frac{1}{p(\mathbf{x}_n | \boldsymbol{\theta})} \frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \Sigma_k}. \quad (11.32)$$

We already know $1/p(\mathbf{x}_n | \boldsymbol{\theta})$ from (11.16). To obtain the remaining partial derivative $\partial p(\mathbf{x}_n | \boldsymbol{\theta})/\partial \Sigma_k$, we write down the definition of the Gaussian distribution $p(\mathbf{x}_n | \boldsymbol{\theta})$, see (11.9), and drop all terms but the k th. We then obtain

$$\frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \Sigma_k} \quad (11.33a)$$

$$= \frac{\partial}{\partial \Sigma_k} \left((2\pi)^{-\frac{D}{2}} \det(\Sigma_k)^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k)) \right) \quad (11.33b)$$

$$= \pi_k (2\pi)^{-\frac{D}{2}} \left[\frac{\partial}{\partial \Sigma_k} \det(\Sigma_k)^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k)) + \det(\Sigma_k)^{-\frac{1}{2}} \frac{\partial}{\partial \Sigma_k} \exp(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k)) \right]. \quad (11.33c)$$

We now use the identities

$$\frac{\partial}{\partial \Sigma_k} \det(\Sigma_k)^{-\frac{1}{2}} \stackrel{(5.100)}{=} -\frac{1}{2} \det(\Sigma_k)^{-\frac{1}{2}} \Sigma_k^{-1}, \quad (11.34)$$

$$\frac{\partial}{\partial \Sigma_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) \stackrel{(5.105)}{=} -\Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} \quad (11.35)$$

and obtain (after some re-arranging) the desired partial derivative required in (11.32) as

$$\begin{aligned} \frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \Sigma_k} &= \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k) \\ &\cdot \left[-\frac{1}{2} (\Sigma_k^{-1} - \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}) \right]. \end{aligned} \quad (11.36)$$

Putting everything together, the partial derivative of the log-likelihood

with respect to Σ_k is given by

$$\frac{\partial \mathcal{L}}{\partial \Sigma_k} = \sum_{n=1}^N \frac{\partial \log p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \Sigma_k} = \sum_{n=1}^N \frac{1}{p(\mathbf{x}_n | \boldsymbol{\theta})} \frac{\partial p(\mathbf{x}_n | \boldsymbol{\theta})}{\partial \Sigma_k} \quad (11.37a)$$

$$= \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)}}_{=r_{nk}} \cdot \left[-\frac{1}{2} (\Sigma_k^{-1} - \Sigma_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}) \right] \quad (11.37b)$$

$$= -\frac{1}{2} \sum_{n=1}^N r_{nk} (\Sigma_k^{-1} - \Sigma_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}) \quad (11.37c)$$

$$= -\frac{1}{2} \underbrace{\Sigma_k^{-1} \sum_{n=1}^N r_{nk}}_{=N_k} + \frac{1}{2} \Sigma_k^{-1} \left(\sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \right) \Sigma_k^{-1}. \quad (11.37d)$$

We see that the responsibilities r_{nk} also appear in this partial derivative. Setting this partial derivative to 0, we obtain the necessary optimality condition

$$N_k \Sigma_k^{-1} = \Sigma_k^{-1} \left(\sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \right) \Sigma_k^{-1} \quad (11.38a)$$

$$\iff N_k \mathbf{I} = \left(\sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \right) \Sigma_k^{-1} \quad (11.38b)$$

By solving for Σ_k , we obtain

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top, \quad (11.39)$$

6371 where \mathbf{r}_k is the probability vector defined in (11.26). This gives us a sim-
6372 ple update rule for Σ_k for $k = 1, \dots, K$ and proves Theorem 11.2. \square

6373 Similar to the update of $\boldsymbol{\mu}_k$ in (11.20), we can interpret the update of
6374 the covariance in (11.31) as an importance-weighted expected value of
6375 the square of the centered data $\tilde{\mathcal{X}}_k := \{\mathbf{x}_1 - \boldsymbol{\mu}_k, \dots, \mathbf{x}_N - \boldsymbol{\mu}_k\}$.

Example 11.4 (Variance Updates)

In our example from Figure 11.3, the variances are updated as follows:

$$\sigma_1^2 : 1 \rightarrow 0.14 \quad (11.40)$$

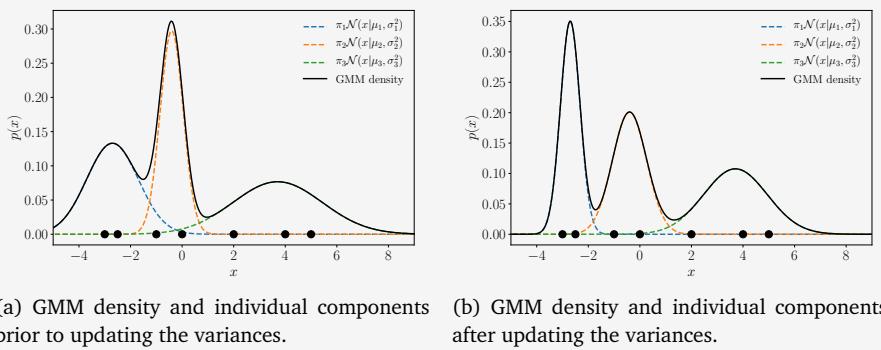
$$\sigma_2^2 : 0.2 \rightarrow 0.44 \quad (11.41)$$

$$\sigma_3^2 : 3 \rightarrow 1.53 \quad (11.42)$$

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.

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.



(a) GMM density and individual components prior to updating the variances. (b) GMM density and individual components after updating the variances.

Similar to the update of the mean parameters, we can interpret (11.31) as a Monte-Carlo estimate of the weighted covariance of data points x_n associated with the k th mixture component, where the weights are the responsibilities r_{nk} . As with the updates of the mean parameters, this update depends on all π_j, μ_j, Σ_j , $j = 1, \dots, K$, through the responsibilities r_{nk} , which prohibits a closed-form solution.

11.2.4 Updating the Mixture Weights

Theorem 11.3 (Update of the GMM Mixture Weights). *The mixture weights of the GMM are updated as*

$$\pi_k^{new} = \frac{N_k}{N} \quad (11.43)$$

for $k = 1, \dots, K$, where N is the number of data points and N_k is defined in (11.25).

Proof To find the partial derivative of the log-likelihood with respect to the weight parameters π_k , $k = 1, \dots, K$, we take the constraint $\sum_k \pi_k = 1$ into account by using Lagrange multipliers (see Section 7.2). The Lagrangian is

$$\mathcal{L} = \mathcal{L} + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right) \quad (11.44a)$$

$$= \sum_{n=1}^N \log \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right), \quad (11.44b)$$

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 derivative with respect to π_k as

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{n=1}^N \frac{\mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \quad (11.45a)$$

$$= \frac{1}{\pi_k} \underbrace{\sum_{n=1}^N \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{=N_k} + \lambda = \frac{N_k}{\pi_k} + \lambda, \quad (11.45b)$$

and the partial derivative with respect to the Lagrange multiplier λ as

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \sum_{k=1}^K \pi_k - 1. \quad (11.46)$$

Setting both partial derivatives to $\mathbf{0}$ (necessary condition for optimum) yields the system of equations

$$\pi_k = -\frac{N_k}{\lambda}, \quad (11.47)$$

$$1 = \sum_{k=1}^K \pi_k. \quad (11.48)$$

Using (11.47) in (11.48) and solving for π_k , we obtain

$$\sum_{k=1}^K \pi_k = 1 \iff -\sum_{k=1}^K \frac{N_k}{\lambda} = 1 \iff -\frac{N}{\lambda} = 1 \iff \lambda = -N. \quad (11.49)$$

This allows us to substitute $-N$ for λ in (11.47) to obtain

$$\pi_k^{\text{new}} = \frac{N_k}{N}, \quad (11.50)$$

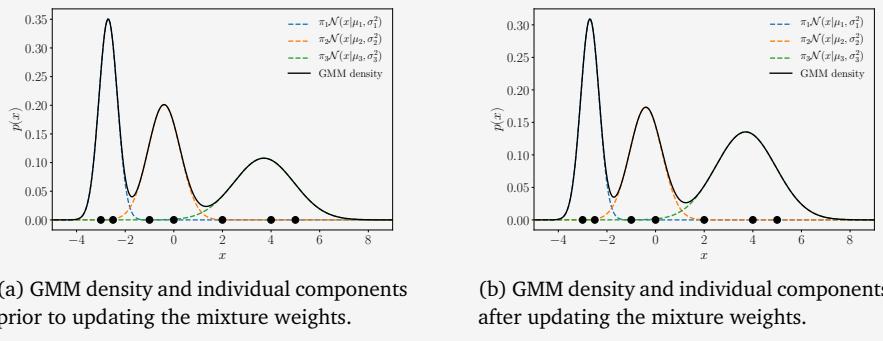
6385 which gives us the update for the weight parameters π_k and proves Theorem 11.3. \square

6387 We can identify the mixture weight in (11.43) as the ratio of the total responsibility of the k th cluster and the number of data points. Since 6388 $N = \sum_k N_k$ 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 k th mixture component for the dataset.

6392 *Remark.* Since $N_k = \sum_{i=1}^N r_{nk}$, the update equation (11.43) for the mixture weights π_k also depends on all $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, j = 1, \dots, K$ via the responsibilities r_{nk} . \diamond

Example 11.5 (Weight Parameter Updates)

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:

$$\pi_1 : \frac{1}{3} \rightarrow 0.29 \quad (11.51)$$

$$\pi_2 : \frac{1}{3} \rightarrow 0.29 \quad (11.52)$$

$$\pi_3 : \frac{1}{3} \rightarrow 0.42 \quad (11.53)$$

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 (11.20), (11.31), and (11.43) do not constitute a closed-form solution for the updates of the parameters μ_k, Σ_k, π_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 algo-*

EM algorithm

6402 *rithm*) was proposed by Dempster et al. (1977) and is a general iterative
 6403 scheme for learning parameters (maximum likelihood or MAP) in mixture
 6404 models and, more generally, latent-variable models.

6405 In our example of the Gaussian mixture model, we choose initial values
 6406 for μ_k, Σ_k, π_k and alternate until convergence between

- 6407 • *E-step*: Evaluate the responsibilities r_{nk} (posterior probability of data
 6408 point n belonging to mixture component k).
- 6409 • *M-step*: Use the updated responsibilities to re-estimate the parameters
 6410 μ_k, Σ_k, π_k .

6411 Every step in the EM algorithm increases the log-likelihood function (Neal
 6412 and Hinton, 1999). For convergence, we can check the log-likelihood or
 6413 the parameters directly. A concrete instantiation of the EM algorithm for
 6414 estimating the parameters of a GMM is as follows:

- 6415 1 Initialize μ_k, Σ_k, π_k
- 2 *E-step*: Evaluate responsibilities r_{nk} for every data point x_n using current parameters π_k, μ_k, Σ_k :

$$r_{nk} = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}. \quad (11.54)$$

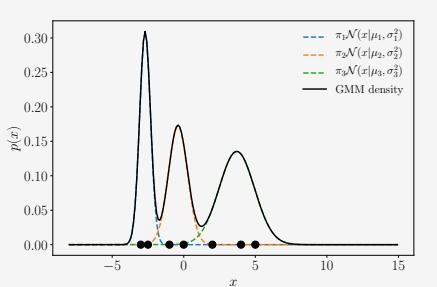
- 3 *M-step*: Re-estimate parameters π_k, μ_k, Σ_k using the current responsibilities r_{nk} (from E-step):

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N r_{nk} x_n, \quad (11.55)$$

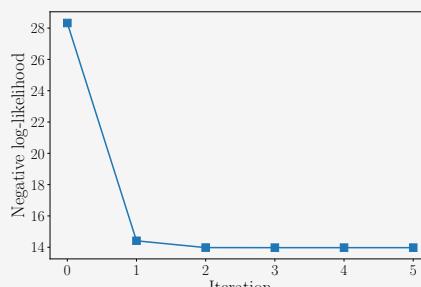
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N r_{nk} (x_n - \mu_k)(x_n - \mu_k)^\top, \quad (11.56)$$

$$\pi_k = \frac{N_k}{N}. \quad (11.57)$$

Example 11.6 (GMM Fit)



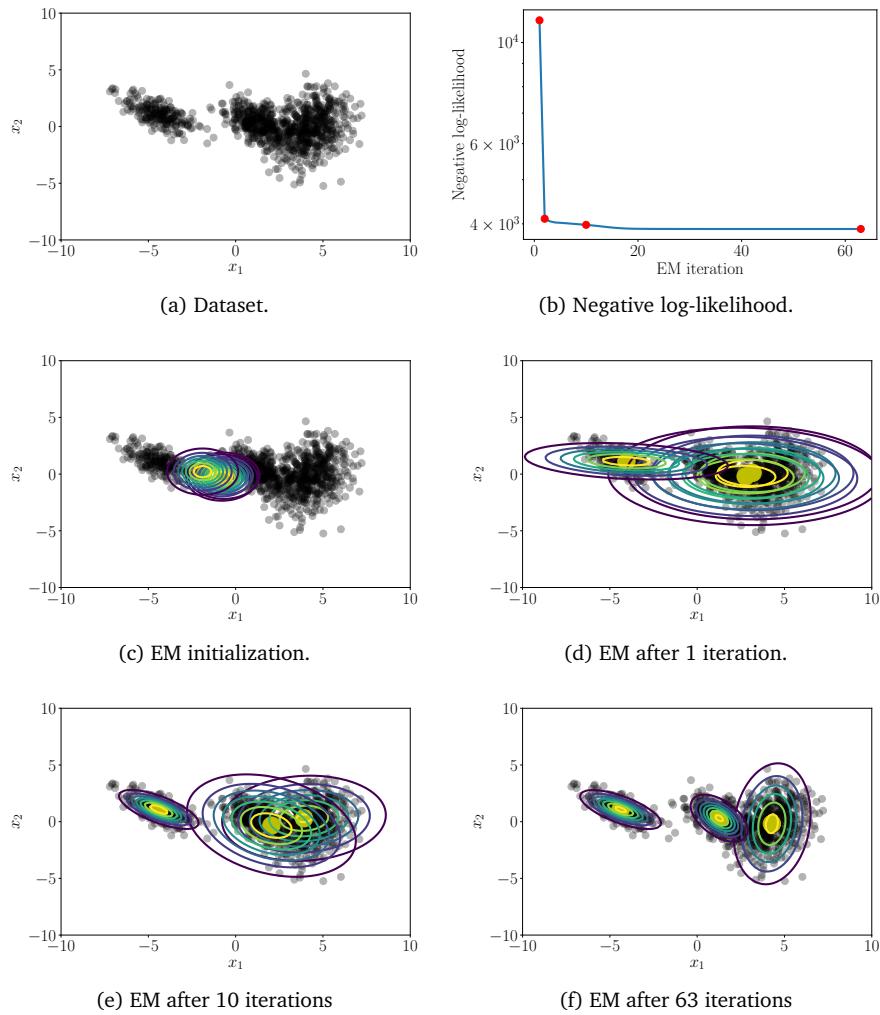
(a) Final GMM fit. After five iterations, the EM algorithm converges and returns this mixture density.



(b) Negative log-likelihood as a function of the EM iterations.

Figure 11.8 EM algorithm applied to the GMM from Figure 11.2.

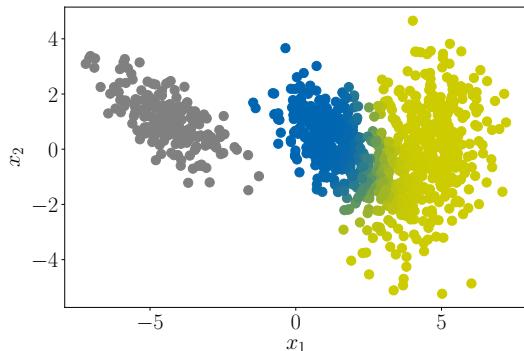
Figure 11.9
 Illustration of the EM algorithm for fitting a Gaussian mixture model with three components to a two-dimensional dataset. (a) Dataset; (b) Negative log likelihood (lower is better) as a function of the EM iterations. The red dots indicate the iterations for which the corresponding GMM fits are shown in (c)–(f). The yellow discs indicate the mean of the Gaussian distribution.



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

$$p(x) = 0.29\mathcal{N}(x | -2.75, 0.06) + 0.28\mathcal{N}(x | 0.50, 0.25) + 0.43\mathcal{N}(x | 3.64, 1.63). \quad (11.58)$$

We applied the EM algorithm to the two-dimensional dataset shown in Figure 11.1 with $K = 3$ mixture components. Figure 11.9 illustrates some steps of the EM algorithm and the negative log-likelihood as a function of the EM iteration (Figure 11.9(b)).



6420 Figure 11.10 visualizes the final responsibilities of the mixture compo-
 6421 nents for the data points. It becomes clear that there are data points that
 6422 cannot be uniquely assigned to a single (either blue or yellow) compo-
 6423 nent, such that the responsibilities of these two clusters for those points
 6424 are around 0.5.

Figure 11.10
 Dataset 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.

11.4 Latent Variable Perspective

6425 We can look at the GMM from the perspective of a discrete latent variable
 6426 model, i.e., where the latent variable z can attain only a finite set of val-
 6427 ues. This is in contrast to PCA where the latent variables were continuous-
 6428 valued numbers in \mathbb{R}^M .

6429 The advantages of the probabilistic perspective are that (i) it will justify
 6430 some ad-hoc decisions we made in the previous sections, (ii) it allows for
 6431 a concrete interpretation of the responsibilities as posterior probabilities,
 6432 (iii) the iterative algorithm for updating the model parameters can be de-
 6433 rived in a principled manner as the EM algorithm for maximum likelihood
 6434 parameter estimation in latent variable models.

11.4.1 Generative Process and Probabilistic Model

6436 To derive the probabilistic model for GMMs, it is useful to think about the
 6437 generative process, i.e., the process that allows us to generate data, using
 6438 a probabilistic model.

6439 We assume a mixture model with K components and that a data point x can be generated by exactly one mixture component. We introduce a binary indicator variable $z_k \in \{0, 1\}$ with two states (see Section 6.2) that indicates whether the k th mixture component generated that data point so that

$$p(x | z_k = 1) = \mathcal{N}(x | \mu_k, \Sigma_k). \quad (11.59)$$

6440 We define $z := [z_1, \dots, z_K]^\top \in \mathbb{R}^K$ as a probability vector consisting of

Sometimes this kind⁶⁴⁴¹ of probability distribution is called “multinoulli”, a generalization of the Bernoulli distribution to more than two values (Murphy, 2012). one-hot encoding
1-of- K representation

$K - 1$ many 0s and exactly one 1. For example, for $K = 3$, a valid \mathbf{z} would be $\mathbf{z} = [z_1, z_2, z_3]^\top = [0, 1, 0]^\top$, which would select the second mixture component since $z_2 = 1$. The properties of \mathbf{z} imply that $\sum_{k=1}^K z_k = 1$. Therefore, \mathbf{z} is a *one-hot encoding* (also: *1-of- K representation*).

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 \mathbf{z} . Given that we do not know which mixture component generated the data point, we treat the indicator vector \mathbf{z} as a latent variable and place a prior

$$p(\mathbf{z}) = \boldsymbol{\pi} = [\pi_1, \dots, \pi_K]^\top, \quad \sum_{k=1}^K \pi_k = 1, \quad (11.60)$$

on it. Then the k th entry

$$\pi_k = p(z_k = 1) \quad (11.61)$$

of this probability vector describes the probability that the k th mixture component generated data point \mathbf{x} .

Remark (Sampling from a GMM). The construction of this latent variable model (see the corresponding graphical model in Figure 11.11) lends itself to a very simple sampling procedure (generative process) to generate data:

- 1 Sample $z^{(i)} \sim p(\mathbf{z})$
- 2 Sample $\mathbf{x}^{(i)} \sim p(\mathbf{x} | z^{(i)} = 1)$

In the first step, we select a mixture component i (via the one-hot encoding \mathbf{z}) at random according to $p(\mathbf{z}) = \boldsymbol{\pi}$; in the second step we draw a sample from the corresponding mixture component. When we discard the samples of the latent variable so that we are left with the $\mathbf{x}^{(i)}$, we have valid samples from the GMM.

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*. ◇

Generally, a probabilistic model is defined by the joint distribution of the data and the latent variables (see Section 8.3). With the prior $p(\mathbf{z})$ defined in (11.60)–(11.61) and the conditional $p(\mathbf{x} | \mathbf{z})$ from (11.59) we obtain all K components of this joint distribution via

$$p(\mathbf{x}, z_k = 1) = p(\mathbf{x} | z_k = 1)p(z_k = 1) = \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (11.62)$$

for $k = 1, \dots, K$, so that

$$p(\mathbf{x}, \mathbf{z}) = \begin{bmatrix} p(\mathbf{x}, z_1 = 1) \\ \vdots \\ p(\mathbf{x}, z_K = 1) \end{bmatrix} = \begin{bmatrix} \pi_1 \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ \vdots \\ \pi_K \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K) \end{bmatrix}, \quad (11.63)$$

which fully specifies the probabilistic model.

6462

11.4.2 Likelihood

To obtain the likelihood $p(\mathbf{x} | \boldsymbol{\theta})$ in a latent variable model, we need to marginalize out the latent variables. In our case, this can be done by summing out all latent variables from the joint $p(\mathbf{x}, \mathbf{z})$ in (11.63) so that

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{\mathbf{z}} p(\mathbf{x} | \boldsymbol{\theta}, \mathbf{z}) p(\mathbf{z} | \boldsymbol{\theta}), \quad \boldsymbol{\theta} := \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k : k = 1, \dots, K\}. \quad (11.64)$$

We now explicitly condition on the parameters $\boldsymbol{\theta}$ of the probabilistic model, which we previously omitted. In (11.64), we sum over all K possible one-hot encodings of \mathbf{z} , which is denoted by $\sum_{\mathbf{z}}$. Since there is only a single non-zero single entry in each \mathbf{z} there are only K possible configurations/settings of \mathbf{z} . For example, if $K = 3$ then \mathbf{z} can have the configurations

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \quad (11.65)$$

Summing over all possible configurations of \mathbf{z} in (11.64) is equivalent to looking at the non-zero entry of the \mathbf{z} -vector and write

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{\mathbf{z}} p(\mathbf{x} | \boldsymbol{\theta}, \mathbf{z}) p(\mathbf{z} | \boldsymbol{\theta}) \quad (11.66a)$$

$$= \sum_{k=1}^K p(\mathbf{x} | \boldsymbol{\theta}, z_k = 1) p(z_k = 1 | \boldsymbol{\theta}) \quad (11.66b)$$

so that the desired marginal distribution is given as

$$p(\mathbf{x} | \boldsymbol{\theta}) \stackrel{(11.66b)}{=} \sum_{k=1}^K p(\mathbf{x} | \boldsymbol{\theta}, z_k = 1) p(z_k = 1 | \boldsymbol{\theta}) \quad (11.67a)$$

$$= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (11.67b)$$

which we identify as the GMM model from (11.3). Given a dataset \mathcal{X} we immediately obtain the likelihood

$$p(\mathcal{X} | \boldsymbol{\theta}) = \prod_{n=1}^N p(\mathbf{x}_n | \boldsymbol{\theta}) \stackrel{(11.67b)}{=} \prod_{n=1}^N \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (11.68)$$

6463 which is exactly the GMM likelihood from (11.9). Therefore, the latent
6464 variable model with latent indicators z_k is an equivalent way of thinking
6465 about a Gaussian mixture model.

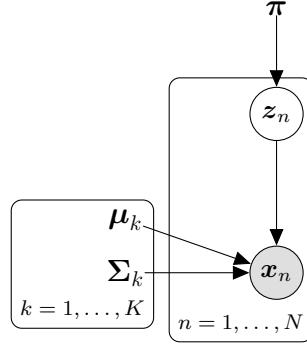
6466

11.4.3 Posterior Distribution

Let us have a brief look at the posterior distribution on the latent variable \mathbf{z} . According to Bayes' theorem, the posterior of the k th component having

Figure 11.12

Graphical model for a GMM with N data points.



generated data point \mathbf{x}

$$p(z_k = 1 | \mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x} | z_k = 1)}{p(\mathbf{x})}, \quad (11.69)$$

where the marginal $p(\mathbf{x})$ is given in (11.67b). This yields the posterior distribution for the k th indicator variable z_k

$$p(z_k = 1 | \mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x} | z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x} | z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \mu_j, \Sigma_j)}, \quad (11.70)$$

which we identify as the responsibility of the k th mixture component for data point \mathbf{x} . Note that we omitted the explicit conditioning on the GMM parameters π_k, μ_k, Σ_k where $k = 1, \dots, K$.

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 \mathbf{x} . However, the concepts of the prior and the posterior can be directly extended to the case of N data points $\mathcal{X} := \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$.

In the probabilistic interpretation of the GMM, every data point \mathbf{x}_n possesses its own latent variable

$$\mathbf{z}_n = [z_{n1}, \dots, z_{nK}]^\top \in \mathbb{R}^K. \quad (11.71)$$

Previously (when we only considered a single data point \mathbf{x}) we omitted the index n , but now this becomes important.

We share the same prior distribution π across all latent variables \mathbf{z}_n . The corresponding graphical model is shown in Figure 11.12, where we use the plate notation.

The conditional distribution $p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{z}_1, \dots, \mathbf{z}_N)$ factorizes over the data points and is given as

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{z}_1, \dots, \mathbf{z}_N) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n). \quad (11.72)$$

To obtain the posterior distribution $p(z_{nk} = 1 | \mathbf{x}_n)$ we follow the same reasoning as in Section 11.4.3 and apply Bayes' theorem to obtain

$$p(z_{nk} = 1 | \mathbf{x}_n) = \frac{p(\mathbf{x}_n | z_{nk} = 1)p(z_{nk} = 1)}{\sum_{j=1}^K p(\mathbf{x}_n | z_{nj} = 1)p(z_{nj} = 1)} \quad (11.73a)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} = r_{nk}. \quad (11.73b)$$

This means that $p(z_k = 1 | \mathbf{x}_n)$ is the (posterior) probability that the k th mixture component generated data point \mathbf{x}_n and corresponds to the responsibility r_{nk} we introduced in (11.17). 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

$$Q(\boldsymbol{\theta} | \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\mathbf{z} | \mathbf{x}, \boldsymbol{\theta}^{(t)}} [\log p(\mathbf{x}, \mathbf{z} | \boldsymbol{\theta})] \quad (11.74a)$$

$$= \int \log p(\mathbf{x}, \mathbf{z} | \boldsymbol{\theta}) p(\mathbf{z} | \mathbf{x}, \boldsymbol{\theta}^{(t)}) d\mathbf{z}, \quad (11.74b)$$

where the expectation of $\log p(\mathbf{x}, \mathbf{z} | \boldsymbol{\theta})$ is taken with respect to the posterior $p(\mathbf{z} | \mathbf{x}, \boldsymbol{\theta}^{(t)})$ of the latent variables. The M-step selects an updated set of model parameters $\boldsymbol{\theta}^{(t+1)}$ by maximizing (11.74b).

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 $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1, \dots, K$, we sample an index k from the probability vector $[\pi_1, \dots, \pi_K]^\top$ and then sample a data point $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. 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 nested cross-validation, as discussed in Section 8.5.1, 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 (2003a), K -means makes a “hard” assignments of data points to cluster centers μ_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 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 π_k, μ_k, Σ_k for $k = 1, \dots, K$, 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).

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

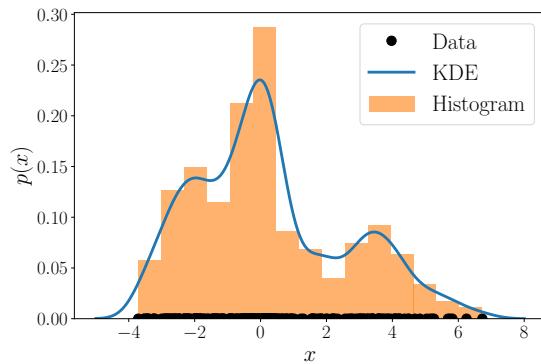


Figure 11.13
Histogram (orange bars) and kernel density estimation (blue line). The kernel density estimator produces a smooth estimate of the underlying density, whereas the histogram is an unsmoothed count measure of how many data points (black) fall into a single bin.

6547 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
 6548 be used to determine a good bin size.
 6549

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

$$p(\mathbf{x}) = \frac{1}{Nh} \sum_{n=1}^N k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right), \quad (11.75)$$

6550 where k is a kernel function, i.e., a non-negative function that integrates
 6551 to 1 and $h > 0$ is a smoothing/bandwidth parameter, which plays a simi-
 6552 lar role as the bin size in histograms. Note that we place a kernel on every
 6553 single data point \mathbf{x}_n in the dataset. Commonly used kernel functions are
 6554 the uniform distribution and the Gaussian distribution. Kernel density esti-
 6555 mates are closely related to histograms, but by choosing a suitable kernel,
 6556 we can guarantee smoothness of the density estimate. Figure 11.13 illus-
 6557 trates the difference between a histogram and a kernel density estimator
 6558 (with a Gaussian-shaped kernel) for a given dataset of 250 data points.

kernel density estimation

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

$$f : \mathbb{R}^D \rightarrow \{+1, -1\}. \quad (12.1)$$

Recall from Chapter 8 that we represent each example 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.

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 $x_n \in \mathbb{R}^D$ along with their corresponding labels $y_n \in \{+1, -1\}$. Given the training data consisting of example-label pairs $(x_1, y_1), \dots, (x_N, y_N)$, 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.

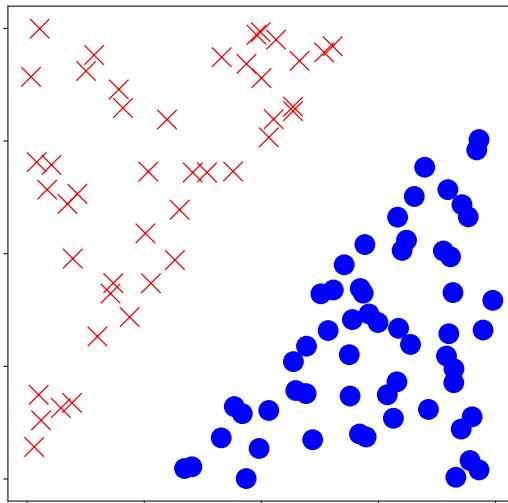
The SVM provides state of the art results in many applications, with sound theoretical guarantees (Steinwart and Christmann, 2008). In this book, the first reason we choose to discuss the SVM is to illustrate a

An example of structure is if the outcomes were ordered, like in the case of small, medium and large t-shirts.
binary classification

Input example x_n may also be referred to as inputs, data points, features or instances.
classes

For probabilistic models, it is mathematically convenient to use $\{0, 1\}$ as a binary representation. See remark after Example 6.15.

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.



6225 geometric way to think about supervised machine learning. Whereas in
 6226 Chapter 9 we considered the machine learning problem in terms of prob-
 6227 abilistic models and attacked it using maximum likelihood estimation and
 6228 Bayesian inference, here we will consider an alternative approach where
 6229 we reason geometrically about the machine learning task. It relies heavily
 6230 on concepts, such as inner products and projections, which we discussed
 6231 in Chapter 3. In contrast to Chapter 9, the optimization problem for SVM
 6232 does not admit an analytic solution. Hence, we resort to the optimization
 6233 tools introduced in Chapter 7. This is the second reason for introducing
 6234 the SVM – as an illustration of what to do when we cannot analytically
 6235 derive a solution.

6236 The SVM view of machine learning is also subtly different from the
 6237 maximum likelihood view of Chapter 9. The maximum likelihood view
 6238 proposes a model based on a probabilistic view of the data distribution,
 6239 from which an optimization problem is derived. In contrast, the SVM view
 6240 starts by designing a particular function that is to be optimized during
 6241 training, based on geometric intuitions. In other words, we start by de-
 6242 signing an objective function that is to be minimized on training data,
 6243 following the principles of empirical risk minimization 8.1. This can also
 6244 be understood as designing a particular loss function.

6245 Let us derive the optimization problem corresponding to training an
 6246 SVM on example-label pairs. Intuitively, we imagine binary classification
 6247 data which can be separated by a hyperplane as illustrated in Figure 12.1,
 6248 where the example (a vector of dimension 2) is used to indicate the lo-
 6249 cation and the label is represented as different symbols (and colours).
 6250 Hyperplane is a word that is commonly used in machine learning, and we
 6251 saw them in Section 2.8 introduced as an affine subspace, which is the
 6252 phrase used in linear algebra. The examples consists of two classes that

6253 have features (the components of the vector representing the example) ar-
 6254 ranged in such a way as to allow us to separate/classify them by drawing
 6255 a straight line.

6256 In the following, we start by formalizing this idea of finding a linear
 6257 separator. We introduce the idea of the margin and then extend linear
 6258 separators to allow for examples to fall on the wrong side. We present
 6259 two equivalent ways of formalizing the SVM: the geometric view (Sec-
 6260 tion 12.2.4) and the loss function view (Section 12.2.5). We derive the
 6261 dual version of the SVM using Lagrange multipliers (Section 7.2). The
 6262 dual SVM allows us to observe a third way of formalizing the SVM: in
 6263 terms of the convex hulls of the examples of each class (Section 12.3.2).
 6264 We conclude by briefly describing kernels and how to numerically solve
 6265 the nonlinear kernel-SVM optimization problem.

6266 12.1 Separating Hyperplanes

6267 Given two examples represented as vectors \mathbf{x}_i and \mathbf{x}_j , one way to com-
 6268 pute the similarity between them is using a inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Recall
 6269 from Section 3.2 that inner products measure the angle between two vec-
 6270 tors. The value of the inner product also depends on the length (norm)
 6271 of each vector. Furthermore, inner products allow us to rigorously define
 6272 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 $\mathbf{x} \in \mathbb{R}^D$ be an element of the data space. Consider a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ parametrized by $\mathbf{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}$ as follows

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b. \quad (12.2)$$

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

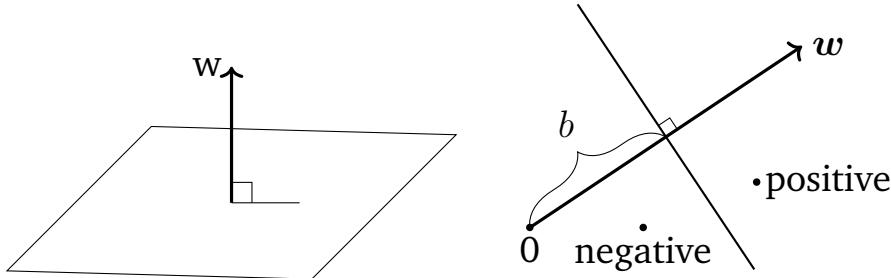
$$\{\mathbf{x} \in \mathbb{R}^D : f(\mathbf{x}) = 0\}. \quad (12.3)$$

An illustration of the hyperplane is shown in Figure 12.2 where the vector \mathbf{w} is a vector normal to the hyperplane and b the intercept. We can derive that \mathbf{w} is a normal vector to the hyperplane in (12.3) by choosing any two examples \mathbf{x}_a and \mathbf{x}_b on the hyperplane and showing that the vector between them is orthogonal to \mathbf{w} . In the form of an equation,

$$f(\mathbf{x}_a) - f(\mathbf{x}_b) = \langle \mathbf{w}, \mathbf{x}_a \rangle + b - (\langle \mathbf{w}, \mathbf{x}_b \rangle + b) \quad (12.4)$$

$$= \langle \mathbf{w}, \mathbf{x}_a - \mathbf{x}_b \rangle, \quad (12.5)$$

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 \mathbf{x}_a and \mathbf{x}_b to be on the hyperplane, this implies that $f(\mathbf{x}_a) = 0$ and $f(\mathbf{x}_b) = 0$ and hence $\langle \mathbf{w}, \mathbf{x}_a - \mathbf{x}_b \rangle = 0$. Recall that two vectors are orthogonal when their inner product is zero, therefore we obtain that \mathbf{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 \mathbf{w} as an arrow indicating a direction. That is we consider \mathbf{w} to be a geometric vector. In contrast, we think of the example vector \mathbf{x} as a point (as indicated by its coordinates). That is we consider \mathbf{x} to be the coordinates of a vector with respect to the standard basis. \diamond

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 \mathbf{x}_{test} , we calculate the value of the function $f(\mathbf{x}_{\text{test}})$ and classify the example as $+1$ if $f(\mathbf{x}_{\text{test}}) \geq 0$ 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.,

$$\langle \mathbf{w}, \mathbf{x}_n \rangle + b \geq 0 \quad \text{when } y_n = +1 \quad (12.6)$$

and the examples with negative labels are on the negative side,

$$\langle \mathbf{w}, \mathbf{x}_n \rangle + b < 0 \quad \text{when } y_n = -1. \quad (12.7)$$

Refer to Figure 12.2 for a geometric intuition of positive and negative examples. These two conditions are often presented in a single equation,

$$y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq 0. \quad (12.8)$$

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.

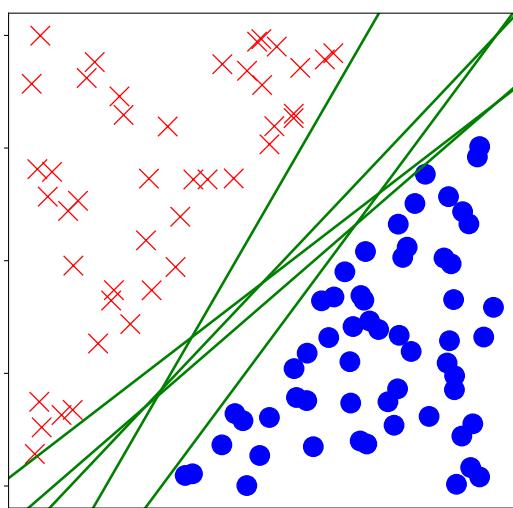


Figure 12.3
Possible separating hyperplanes. There are many linear classifiers (green lines) that separate red crosses from blue dots.

6295

12.2 Primal Support Vector Machine

6296 Based on the concept of distances from points to a hyperplane, we now
 6297 are in a position to discuss the support vector machine. For a dataset
 6298 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ that is linearly separable, we have many candidate
 6299 hyperplanes (refer to Figure 12.3) that solve our classification problem
 6300 without any (training) errors. In other words, for a given training set we
 6301 have many candidate classifiers. One idea is to choose the separating hy-
 6302 perplane that maximizes the margin between the positive and negative
 6303 examples. In the following, we use the concept of a hyperplane, see also
 6304 Section 2.8, and derive the distance between an example and a hyper-
 6305 plane. Recall that the closest point on the hyperplane to a given point
 6306 (example \mathbf{x}_n) is obtained by the orthogonal projection (Section 3.7). We
 6307 will see in the next section how to use the orthogonal projection to derive
 6308 the margin.

A classifier with large margin turns out to generalize well (Steinwart and Christmann, 2008).

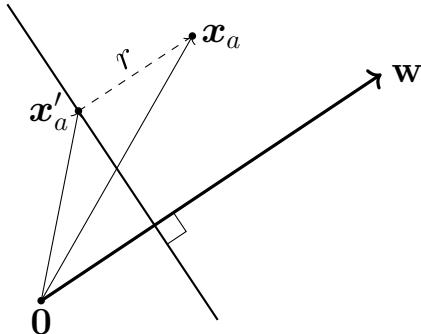
6309

12.2.1 Concept Of The Margin

6310 The concept of the *margin* is intuitively simple: It is the distance of the
 6311 separating hyperplane to the closest examples in the dataset, assuming
 6312 that the dataset is linearly separable. However, when trying to formalize
 6313 this distance, there is a technical wrinkle that is confusing. The technical
 6314 wrinkle is that we need to define a scale at which to measure the dis-
 6315 tance. A potential scale is to consider the scale of the data, i.e., the raw
 6316 values of \mathbf{x}_n . There are problems with this, as we could change the units
 6317 of measurement of \mathbf{x}_n and change the values in \mathbf{x}_n , and, hence, change
 6318 the distance to the hyperplane. As we will see shortly, we define the scale
 6319 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:
 $\mathbf{x}_a = \mathbf{x}'_a + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$.



Consider a hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b$, and an example \mathbf{x}_a as illustrated in Figure 12.4. Without loss of generality, we can consider the example \mathbf{x}_a to be on the positive side of the hyperplane, i.e., $\langle \mathbf{w}, \mathbf{x}_a \rangle + b > 0$. We would like to derive the distance $r > 0$ of \mathbf{x}_a from the hyperplane. We do so by considering the orthogonal projection (Section 3.7) of \mathbf{x}_a onto the hyperplane, which we denote by \mathbf{x}'_a . Since \mathbf{w} is orthogonal to the hyperplane, we know that the distance r is just a scaling of this vector \mathbf{w} . However, we need to use a vector of unit length (its norm must be 1), and obtain this by dividing \mathbf{w} by its norm, $\frac{\mathbf{w}}{\|\mathbf{w}\|}$. Using vector addition (Section 2.4) we obtain

$$\mathbf{x}_a = \mathbf{x}'_a + r \frac{\mathbf{w}}{\|\mathbf{w}\|}. \quad (12.9)$$

Another way of thinking about r is that it is the coordinate of \mathbf{x}_a in the subspace spanned by \mathbf{w} . We have now expressed the distance of \mathbf{x}_a from the hyperplane as r , and if we choose \mathbf{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

$$y_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq r. \quad (12.10)$$

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 \mathbf{w} is of unit length, that is, $\|\mathbf{w}\| = 1$ where we use the Euclidean norm $\|\mathbf{w}\| = \sqrt{\mathbf{w}^\top \mathbf{w}}$ (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 $\|\mathbf{w}\| = 1$ 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.

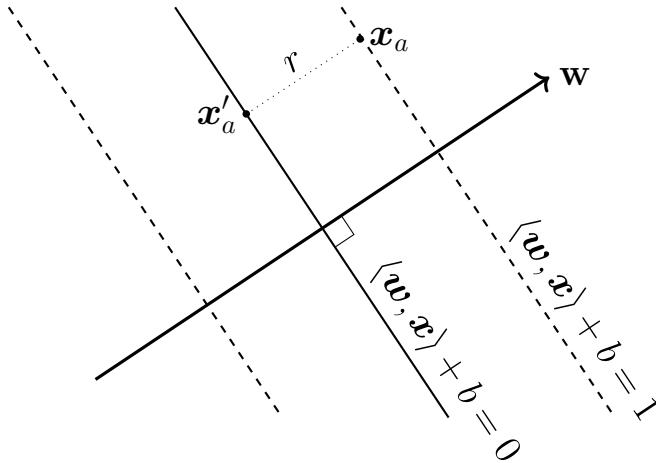


Figure 12.5
Derivation of the
margin: $r = \frac{1}{\|w\|}$.

obtain the following

$$\max_{w,b,r} \underbrace{r}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle w, x_n \rangle + b) \geq r}_{\text{data fitting}}, \quad \underbrace{\|w\| = 1}_{\text{normalization}}, \quad r > 0, \quad (12.11)$$

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). \diamond

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 w and not its length, leading to the assumption that $\|w\| = 1$. 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 $\langle w, x \rangle + b$ is 1 at the closest example. Let us also consider x_a to be the example in the dataset that is closest to the hyperplane.

Figure 12.5 is the same as Figure 12.4, except that now we have rescaled the axes, such that we have the example x_a exactly on the margin, i.e., $\langle w, x_a \rangle + b = 1$. Since x'_a is the orthogonal projection of x_a onto the

Recall that we currently consider linearly separable data.

hyperplane, it must by definition lie on the hyperplane, i.e.,

$$\langle \mathbf{w}, \mathbf{x}'_a \rangle + b = 0. \quad (12.12)$$

By substituting (12.9) into (12.12) we obtain

$$\left\langle \mathbf{w}, \mathbf{x}_a - r \frac{\mathbf{w}}{\|\mathbf{w}\|} \right\rangle + b = 0. \quad (12.13)$$

Multiplying out the inner product, we get

$$\langle \mathbf{w}, \mathbf{x}_a \rangle + b - r \frac{\langle \mathbf{w}, \mathbf{w} \rangle}{\|\mathbf{w}\|} = 0, \quad (12.14)$$

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, $\langle \mathbf{w}, \mathbf{x}_a \rangle + b = 1$. From (3.18) in Section 3.1 we recall that $\langle \mathbf{w}, \mathbf{w} \rangle = \|\mathbf{w}\|^2$, and hence the second term reduces to $r\|\mathbf{w}\|$. Using these simplifications, we obtain

$$r = \frac{1}{\|\mathbf{w}\|}, \quad (12.15)$$

where we have derived the distance r in terms of the normal vector \mathbf{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 \mathbf{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 $\frac{1}{\|\mathbf{w}\|}$. In Section 12.2.3 we will see that the choice that the margin equals 1 is equivalent to our previous assumption of $\|\mathbf{w}\| = 1$ in Section 12.2.1.

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

$$y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq 1. \quad (12.16)$$

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

$$\max_{w,b} \frac{1}{\|\mathbf{w}\|} \quad (12.17)$$

$$\text{subject to } y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq 1 \quad \text{for all } n = 1, \dots, N. \quad (12.18)$$

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 \mathbf{w}, b but yields a tidier form when we take the derivative. Then, our objective becomes

$$\min_{w,b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (12.19)$$

The squared norm results in a convex quadratic programming problem for the SVM (Section 12.3.4).

subject to $y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq 1$ for all $n = 1, \dots, N$. (12.20)

6354 Equation (12.19) is known as the *hard margin SVM*. The reason for the
 6355 expression “hard” is because the above formulation does not allow for any
 6356 violations of the margin condition. We will see in Section 12.2.4 that this
 6357 “hard” condition can be relaxed to accommodate violations.

hard margin SVM

12.2.3 Why We Can Set The Margin To 1

6359 In Section 12.2.1 we argue that we would like to maximize some value
 6360 r , which represents the distance of the closest example to the hyperplane.
 6361 In Section 12.2.2 we scaled the data such that the closest example is of
 6362 distance 1 to the hyperplane. Here we relate the two derivations, and
 6363 show that they are actually equivalent.

Theorem 12.1. *Maximizing the margin r where we consider normalized weights as in (12.11),*

$$\max_{\mathbf{w}, b, r} \underbrace{r}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq r}_{\text{data fitting}}, \quad \underbrace{\|\mathbf{w}\| = 1}_{\text{normalization}}, \quad r > 0 \quad (12.21)$$

is equivalent to scaling the data such that the margin is unity

$$\min_{\mathbf{w}, b} \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq 1}_{\text{data fitting}}. \quad (12.22)$$

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 $\|\mathbf{w}\| = 1$ we can reparameterize the equation with a new weight vector \mathbf{w}' that is not normalized by explicitly using $\frac{\mathbf{w}'}{\|\mathbf{w}'\|}$,

$$\max_{\mathbf{w}', b, r} r^2 \quad \text{subject to} \quad y_n \left(\left\langle \frac{\mathbf{w}'}{\|\mathbf{w}'\|}, \mathbf{x}_n \right\rangle + b \right) \geq r, \quad r > 0. \quad (12.23)$$

In (12.23) we have explicitly written that distances are non-negative. We can divide the first constraint by r ,

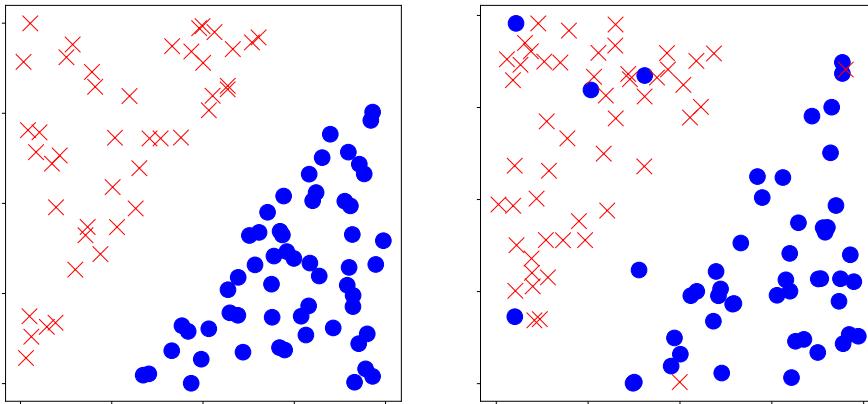
$$\max_{\mathbf{w}', b, r} r^2 \quad \text{subject to} \quad y_n \left(\underbrace{\left\langle \frac{\mathbf{w}'}{\|\mathbf{w}'\| r}, \mathbf{x}_n \right\rangle}_{\mathbf{w}''} + \underbrace{\frac{b}{r}}_{b''} \right) \geq 1, \quad r > 0 \quad (12.24)$$

Note that $r > 0$ because we assumed linear separability, and hence there is no issue to divide by r .

renaming the parameters to \mathbf{w}'' and b'' . Since $\mathbf{w}'' = \frac{\mathbf{w}'}{\|\mathbf{w}'\| r}$, rearranging for r gives

$$\|\mathbf{w}''\| = \left\| \frac{\mathbf{w}'}{\|\mathbf{w}'\| r} \right\| = \left| \frac{1}{r} \right| \cdot \left\| \frac{\mathbf{w}'}{\|\mathbf{w}'\|} \right\| = \frac{1}{r}. \quad (12.25)$$

Figure 12.6 (left) linearly separable data, with a large margin. (right) non-separable data.



Substituting into (12.24), we obtain

$$\max_{\mathbf{w}'', b''} \frac{1}{\|\mathbf{w}''\|^2} \quad \text{subject to} \quad y_n (\langle \mathbf{w}'', \mathbf{x}_n \rangle + b'') \geq 1. \quad (12.26)$$

6364 The final step is to observe that maximizing $\frac{1}{\|\mathbf{w}''\|^2}$ yields the same solution
 6365 as minimizing $\frac{1}{2} \|\mathbf{w}''\|^2$. \square

6366 12.2.4 Soft Margin SVM: Geometric View

6367 We may wish to allow some examples to fall within the margin region,
 6368 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
 6369 when we do not have linearly separable data.

soft margin SVM 6371 The resulting model is called the *soft margin SVM*. In this section, we
 6372 derive the resulting optimization problem using geometric arguments. In
 6373 Section 12.2.5, we will derive the same optimization problem using the
 6374 idea of a loss function. Using Lagrange multipliers (Section 7.2), we will
 6375 derive the dual optimization problem of the SVM in Section 12.3. This
 6376 dual optimization problem allows us to observe a third interpretation of
 6377 the SVM, as a hyperplane that bisects the line between convex hulls cor-
 6378 responding to the positive and negative data examples (Section 12.3.2).

slack variable 6379 The key geometric idea is to introduce a *slack variable* ξ_n corresponding
 6380 to each example (\mathbf{x}_n, y_n) that allows a particular example to be within the
 6381 margin or even on the wrong side of the hyperplane (refer to Figure 12.7).
 6382 We subtract the value of ξ_n from the margin, constraining ξ_n to be non-
 6383 negative. To encourage correct classification of the samples, we add ξ_n to
 6384 the objective

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \quad (12.27)$$

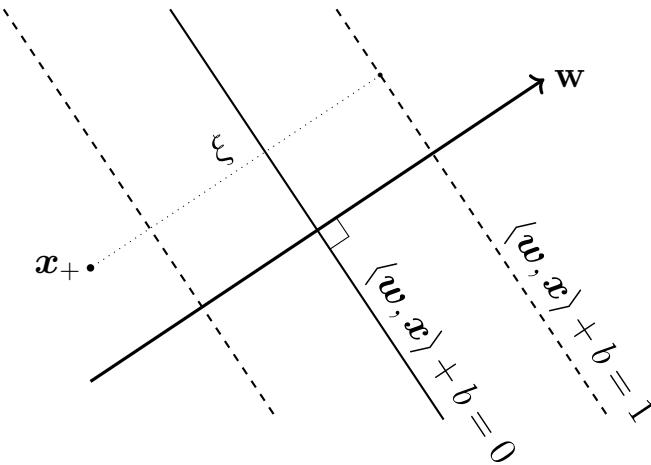


Figure 12.7 Soft Margin SVM allows examples to be within the margin or on the wrong side of the hyperplane. The slack variable ξ measures the distance of a positive example x_+ to the positive margin hyperplane $\langle w, x \rangle + b = 1$ when x_+ is on the wrong side.

$$\text{subject to } y_n(\langle w, x_n \rangle + b) \geq 1 - \xi_n \quad \text{for all } n = 1, \dots, N \quad (12.28)$$

$$\xi_n \geq 0 \quad \text{for all } n = 1, \dots, N. \quad (12.29)$$

6379 In contrast to the optimization problem (12.19) from the previous section
 6380 (the hard margin SVM), this one is called the *soft margin SVM*. The pa-
 6381 rameter $C > 0$ trades off the size of the margin and the total amount of
 6382 slack that we have. This parameter is called the *regularization parameter*
 6383 since, as we will see in the following section, the margin term in the ob-
 6384 jective function (12.27) is a regularization term. The margin term $\|w\|^2$ is
 6385 called the *regularizer*, and in many books on numerical optimization, the
 6386 regularization parameter multiplied with this term (Section 8.1.3). This
 6387 is in contrast to our formulation in this section. Some care needs to be
 6388 taken when interpreting the regularizer, as a large value of C implies low
 6389 regularization, as we give the slack variables larger weight.
 6390 *Remark.* One detail to note is that in the formulation of the SVM (12.27)
 6391 w is regularized but b is not regularized. We can see this by observing that
 6392 the regularization term does not contain b . The unregularized term b com-
 6393 plifies theoretical analysis (Steinwart and Christmann, 2008, Chapter 1)
 6394 and decreases computational efficiency (Fan et al., 2008). ◇

soft margin SVM

regularization parameter

regularizer

There are alternative parametrizations of this regularization, which is why (12.27) is also often referred to as the C -SVM.

6395 12.2.5 Soft Margin SVM: Loss Function View

6396 Recall from Section 9.2.1 that when performing maximum likelihood es-
 6397 timation we usually consider the negative log likelihood. Furthermore since
 6398 the likelihood term for linear regression with Gaussian noise is Gaussian,
 6399 the negative log likelihood for each example is a squared error function
 6400 (9.8). The squared error function is the term that is minimized when look-
 6401 ing for the maximum likelihood solution. Let us consider the error func-
 6402 tion point of view, which is also known as the *loss function* point of view.

loss function

6403 In contrast to Chapter 9 where we consider regression problems (the out-
 6404 put of the predictor is a real number), in this chapter we consider binary
 6405 classification problems (the output of the predictor is one of two labels
 6406 $\{+1, -1\}$). Therefore the error function or the loss function for each sin-
 6407 gle (example, label) pair needs to be appropriate for binary classification.
 6408 For example, the squared loss that is used for regression (9.9b) is not
 6409 suitable for binary classification.

6410 *Remark.* The ideal loss function between binary labels is to count the num-
 6411 ber of mismatches between the prediction and the label. That is for a pre-
 6412 dictor f applied to an example \mathbf{x}_n , we compare the output $f(\mathbf{x}_n)$ with the
 6413 label y_n . We define the loss to be zero if they match, and one if they do not
 6414 match. This is denoted by $\mathbf{1}(f(\mathbf{x}_n) \neq y_n)$ and is called the zero-one loss.
 6415 Unfortunately the zero-one loss results in a difficult optimization problem
 6416 for finding the best parameters \mathbf{w}, b . \diamond

What is the loss function corresponding to the SVM? Consider the error
 between the output of a predictor $f(\mathbf{x}_n)$ 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*

$$\ell(t) = \max\{0, 1 - t\} \quad \text{where } t = y f(\mathbf{x}) = y(\langle \mathbf{w}, \mathbf{x} \rangle + b). \quad (12.30)$$

If $f(\mathbf{x})$ is on the correct side (based on y) of the hyperplane, and further
 than distance 1, this means that $t \geq 1$ and the hinge loss returns a value
 of zero. If $f(\mathbf{x})$ is on the correct side but close to the hyperplane, that
 is, $0 < t < 1$, then the example \mathbf{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

$$\ell(t) = \begin{cases} 0 & \text{if } t \geq 1 \\ 1 - t & \text{if } t < 1 \end{cases}, \quad (12.31)$$

as illustrated in Figure 12.8. The loss corresponding to the hard margin
 SVM 12.19 is defined as follows

$$\ell(t) = \begin{cases} 0 & \text{if } t \geq 1 \\ \infty & \text{if } t < 1 \end{cases}. \quad (12.32)$$

6417 This loss can be interpreted as never allowing any examples inside the
 6418 margin.

For a given training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ we would like to min-
 imize 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-

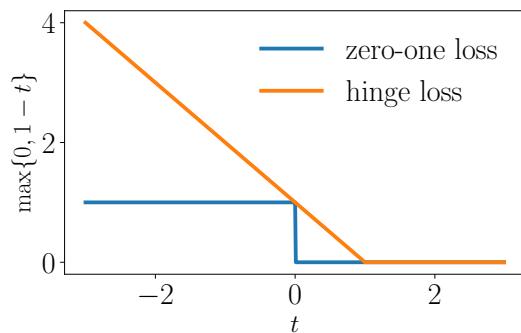


Figure 12.8 Hinge Loss is a convex envelope of zero-one loss.

strained optimization problem

$$\min_{\mathbf{w}, b} \underbrace{\frac{1}{2} \|\mathbf{w}\|^2}_{\text{regularizer}} + C \underbrace{\sum_{n=1}^N \max\{0, 1 - y_n(\langle \mathbf{w}, \mathbf{x}_n \rangle + b)\}}_{\text{error term}}. \quad (12.33)$$

6419 The first term in (12.33) is called the regularization term or the *regularizer*
 6420 (see Section 9.2.3), and the second term is called the *loss term* or the
 6421 *error term*. Recall from Section 12.2.4 that the term $\frac{1}{2} \|\mathbf{w}\|^2$ is actually the
 6422 term arising from the margin. In other words, margin maximization can
 6423 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.,

$$\min_t \max\{0, 1 - t\} \quad (12.34)$$

is equivalent to

$$\begin{aligned} \min_{\xi, t} \quad & \xi \\ \text{subject to} \quad & \xi \geq 0 \\ & \xi \geq 1 - t. \end{aligned} \quad (12.35)$$

6424 By substituting this into (12.33) and rearranging one of the constraints,
 6425 we obtain exactly the soft margin SVM (12.27).

6426 12.3 Dual Support Vector Machine

6427 The description of the SVM in the previous sections, in terms of the vari-
 6428 ables \mathbf{w} and b , is known as the primal SVM. Recall that we are considering

6429 input vectors \mathbf{x} , which have dimension D , i.e., we are looking at input ex-
 6430 amples with D features. Since \mathbf{w} is of the same dimension as \mathbf{x} , this means
 6431 that the number of parameters (the dimension of \mathbf{w}) of the optimization
 6432 problem grows linearly with the number of features.

6433 In the following, we consider an equivalent optimization problem (the
 6434 so-called dual view) which is independent of the number of features. We
 6435 see a similar idea appear in Chapter 10 where we express the learning
 6436 problem in a way that does not scale with the number of features. This
 6437 is useful for problems where we have more features than the number of
 6438 examples. Instead the number of parameters increases with the number
 6439 of examples in the training set. The dual SVM also has the additional ad-
 6440 vantage that it easily allows kernels to be applied, as we shall see at the
 6441 end of this chapter. The word “dual” appears often in mathematical liter-
 6442 ature, and in this particular case it refers to convex duality. The following
 6443 subsections are essentially an application of convex duality as discussed
 6444 in Section 7.2.

6445

12.3.1 Convex Duality Via Lagrange Multipliers

Recall the primal soft margin SVM (12.27). We call the variables \mathbf{w} , b and ξ corresponding to the primal SVM the primal variables. We use $\alpha_n \geq 0$ as the Lagrange multiplier corresponding to the constraint (12.28) that the examples are classified correctly and $\gamma_n \geq 0$ as the Lagrange multiplier corresponding to the non-negativity constraint of the slack variable, see (12.29). The Lagrangian is then given by

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \gamma) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \underbrace{\sum_{n=1}^N \alpha_n (\langle \mathbf{w}, \mathbf{x}_n \rangle + b) - 1 + \xi_n}_{\text{constraint (12.28)}} - \underbrace{\sum_{n=1}^N \gamma_n \xi_n}_{\text{constraint (12.29)}}. \quad (12.36)$$

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 .

Differentiating the Lagrangian (12.36) with respect to the three primal variables \mathbf{w} , b and ξ respectively, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n, \quad (12.37)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{n=1}^N \alpha_n y_n, \quad (12.38)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = C - \alpha_n - \gamma_i. \quad (12.39)$$

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

$$\mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n, \quad (12.40)$$

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 existence can be found in Yu et al. (2013).

Remark. The representer theorem (12.40) also provides an explanation of the name Support Vector Machine. The examples \mathbf{x}_n whose corresponding parameters $\alpha_n = 0$ do not contribute to the solution \mathbf{w} at all. The other examples, where $\alpha_n > 0$, are called *support vectors* since they “support” the hyperplane. ◇

By substituting the expression for \mathbf{w} into the Lagrangian (12.36), we obtain the dual

$$\begin{aligned} \mathfrak{D}(\xi, \alpha, \gamma) = & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^N y_i \alpha_i \left\langle \sum_{j=1}^N y_j \alpha_j \mathbf{x}_j, \mathbf{x}_i \right\rangle \\ & + C \sum_{i=1}^N \xi_i - b \sum_{i=1}^N y_i \alpha_i + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \alpha_i \xi_i - \sum_{i=1}^N \gamma_i \xi_i. \end{aligned} \quad (12.41)$$

Note that there are no longer any terms involving the primal variable \mathbf{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

$$\mathfrak{D}(\xi, \alpha, \gamma) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^N \alpha_i + \sum_{i=1}^N (C - \alpha_i - \gamma_i) \xi_i. \quad (12.42)$$

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 $\alpha_i \leq C$.

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. support vectors

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*

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^N \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^N y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \text{for all } i = 1, \dots, N. \end{aligned} \tag{12.43}$$

The equality constraint in (12.43) is obtained from setting (12.38) to zero. The inequality constraint $\alpha_i \geq 0$ is the condition imposed on Lagrange multipliers of inequality constraints (Section 7.2). The inequality constraint $\alpha_i \leq C$ 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 \mathbf{w} by using the representer theorem (12.40). Let us call the optimal primal parameter \mathbf{w}^* . However there remains the question on how to obtain the parameter b^* . Consider an example (\mathbf{x}_n) that lies exactly on the margin’s boundary, that is, $\langle \mathbf{w}^*, \mathbf{x}_n \rangle + b = y_n$. Recall that y_n is either +1 or -1, and therefore the only unknown is b which can be computed by

$$b^* = y_n - \langle \mathbf{w}^*, \mathbf{x}_n \rangle. \tag{12.44}$$

Remark. In principle there may be no examples that lie exactly on the margin. In this case we should compute $|y_n - \langle \mathbf{w}^*, \mathbf{x}_n \rangle|$ 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/>. ◇

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 Tuck conditions, for example in Schölkopf and Smola (2002).

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 \mathbf{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.

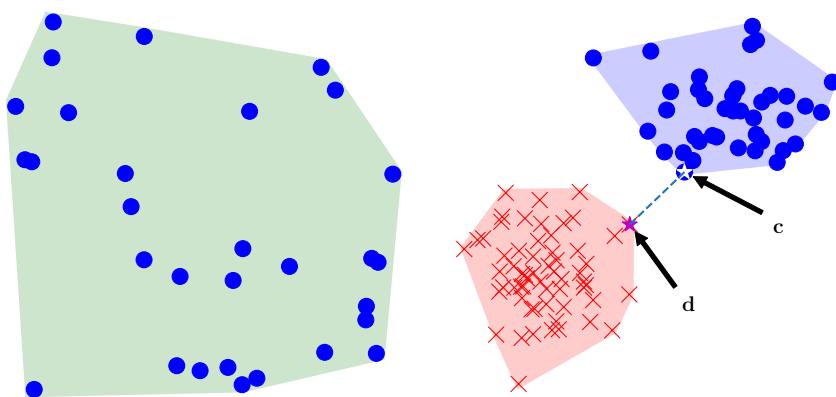


Figure 12.9 (left) Convex hull of points (right) A convex hull around positive and negative examples.

6483 Consider two points x_1 and x_2 and corresponding non-negative weights
 6484 $\alpha_1, \alpha_2 \geq 0$ such that $\alpha_1 + \alpha_2 = 1$. The equation $\alpha_1 x_1 + \alpha_2 x_2$ describes
 6485 each point on a line between x_1 and x_2 . Consider what happens when we
 6486 add a third point x_3 along with a weight $\alpha_3 \geq 0$ such that $\sum_{n=1}^3 \alpha_n =$
 6487 1. The convex combination of these three points x_1, x_2, x_3 span a two
 6488 dimensional area. The convex hull of this area is the triangle formed by
 6489 the edges corresponding to each pair of points. As we add more points,
 6490 and the number of points become greater than the number of dimensions,
 6491 some of the points will be inside the convex hull, as we can see in the left
 6492 of Figure 12.9.

In general, building a convex boundary of points (called the *convex hull*)
 convex hull
 can be done by introducing non-negative weights $\alpha_n \geq 0$ corresponding
 to each example x_n . Then the convex hull can be described as the set

$$\text{conv}(\mathbf{X}) = \left\{ \sum_{n=1}^N \alpha_n \mathbf{x}_n \right\} \quad \text{with} \quad \sum_{n=1}^N \alpha_n = 1 \quad \text{and} \quad \alpha_n \geq 0, \quad (12.45)$$

for all $n = 1, \dots, N$. 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 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ we form two convex hulls, corresponding to the positive and negative classes respectively. We pick a point 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 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 d to c

$$\mathbf{w} = \mathbf{c} - \mathbf{d}. \quad (12.46)$$

Picking the points c and 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 \mathbf{w} , such that we end up with the corresponding optimization

problem

$$\operatorname{argmin}_{\mathbf{w}} \|\mathbf{w}\| = \operatorname{argmin}_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2. \quad (12.47)$$

Since \mathbf{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^+

$$\mathbf{c} = \sum_{y_n=+1} \alpha_n^+ \mathbf{x}_n. \quad (12.48)$$

Similarly, for the examples with negative labels we obtain

$$\mathbf{d} = \sum_{y_n=-1} \alpha_n^- \mathbf{x}_n. \quad (12.49)$$

By substituting (12.46), (12.48) and (12.49) into (12.47), we obtain the following objective function

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \left\| \sum_{y_n=+1} \alpha_n^+ \mathbf{x}_n - \sum_{y_n=-1} \alpha_n^- \mathbf{x}_n \right\|^2. \quad (12.50)$$

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,

$$\sum_{y_n=+1} \alpha_n^+ = 1 \quad \text{and} \quad \sum_{y_n=-1} \alpha_n^- = 1. \quad (12.51)$$

This implies the constraint

$$\sum_{n=1}^N y_n \alpha_n = 0. \quad (12.52)$$

This result can be seen by multiplying out the individual classes

$$\begin{aligned} \sum_{n=1}^N y_n \alpha_n &= \sum_{y_n=+1} (+1) \alpha_n^+ + \sum_{y_n=-1} (-1) \alpha_n^- \\ &= \sum_{y_n=+1} \alpha_n^+ - \sum_{y_n=-1} \alpha_n^- = 1 - 1 = 0. \end{aligned} \quad (12.53)$$

6493 The objective function (12.50) and the constraint (12.52), along with the
6494 assumption that $\boldsymbol{\alpha} \geq 0$, give us a constrained (convex) optimization prob-
6495 lem. This optimization problem can be shown to be the same as that of
6496 the dual hard margin SVM (Bennett and Bredensteiner, 2000a).

6497 *Remark.* To obtain the soft margin dual, we consider the reduced hull. The
6498 *reduced hull* is similar to the convex hull but has an upper bound to the
6499 size of the coefficients $\boldsymbol{\alpha}$. The maximum possible value of the elements
6500 of $\boldsymbol{\alpha}$ restricts the size that the convex hull can take. In other words, the
6501 bound on $\boldsymbol{\alpha}$ shrinks the convex hull to a smaller volume (Bennett and
6502 Bredensteiner, 2000b). \diamond

reduced hull

6503 **12.3.3 Kernels**

6504 Consider the formulation of the dual SVM (12.43). Notice that the inner
 6505 product in the objective occurs only between examples \mathbf{x}_i and \mathbf{x}_j . There
 6506 are no inner products between the examples and the parameters. There-
 6507 fore if we consider a set of features $\phi(\mathbf{x}_i)$ to represent \mathbf{x}_i , the only change
 6508 in the dual SVM will be to replace the inner product. This modularity,
 6509 where the choice of the classification method (the SVM) and the choice
 6510 of the feature representation $\phi(\mathbf{x})$ can be considered separately, provides
 6511 flexibility for us to explore the two problems independently.

Since $\phi(\mathbf{x})$ 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 $\phi(\cdot)$ and computing the resulting inner product between examples \mathbf{x}_i and \mathbf{x}_j , we define a similarity function $k(\mathbf{x}_i, \mathbf{x}_j)$ between \mathbf{x}_i and \mathbf{x}_j . For a certain class of similarity functions called *kernels*, the definition of the similarity function *implicitly* defines a non-linear feature map $\phi(\cdot)$. Kernels are by definition functions $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ for which there exists a Hilbert space \mathcal{H} and $\phi : \mathcal{X} \rightarrow \mathcal{H}$ a feature map such that

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}. \quad (12.54)$$

There is a unique reproducing kernel Hilbert space associated with every kernel k (Aronszajn, 1950; Berlinet and Thomas-Agnan, 2004). In this unique association $\phi(\mathbf{x}) = k(\cdot, \mathbf{x})$ 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 $\mathbf{K} \in \mathbb{R}^{N \times N}$, 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 \mathbf{K} must be symmetric and positive semi-definite (Section 3.2.3):

$$\forall \mathbf{z} \in \mathbb{R}^N \quad \mathbf{z}^\top \mathbf{K} \mathbf{z} \geq 0. \quad (12.55)$$

6512 Some popular examples of kernels for multivariate real-valued data $\mathbf{x}_i \in$
 6513 \mathbb{R}^D are the polynomial kernel, the Gaussian radial basis function kernel,
 6514 and the rational quadratic kernel. Figure 12.10 illustrates the effect of
 6515 different kernels on separating hyperplanes on an example dataset.

6516 *Remark.* Unfortunately for the fledgling machine learner, there are multi-
 6517 ple meanings of the word kernel. In this chapter, the word kernel comes
 6518 from the idea of the Reproducing Kernel Hilbert Space (RKHS) (Aron-
 6519 szajn, 1950; Saitoh, 1988). We have discussed the idea of the kernel in

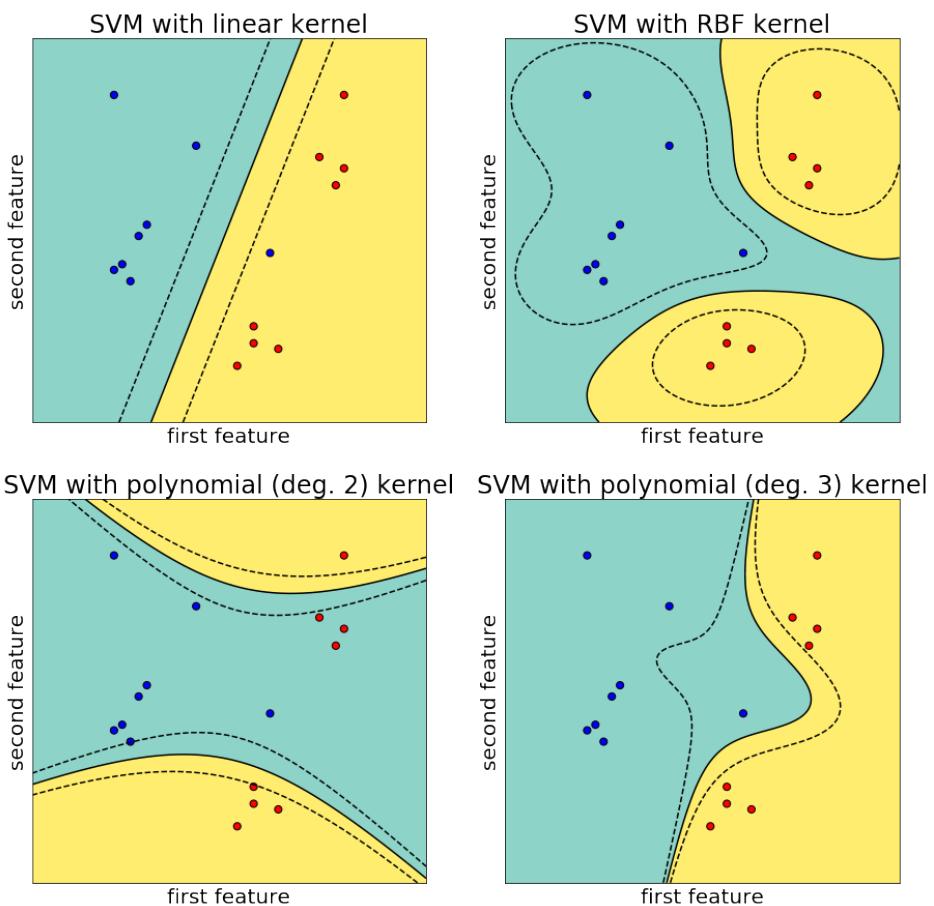
kernels

kernel trick

Gram matrix

kernel matrix

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 $\phi(x)$ is mathematically equivalent to the kernel representation $k(\mathbf{x}_i, \mathbf{x}_j)$, 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,

6536 sequences, strings and graphs (Ben-Hur et al., 2008; Gärtner, 2008; Shi
 6537 et al., 2009; Vishwanathan et al., 2010).

6538 12.3.4 Numerical Solution

6539 We conclude our discussion of SVMs by looking at how to express the
 6540 problems derived in this chapter in terms of the concepts presented in
 6541 Chapter 7. We consider two different approaches for finding the optimal
 6542 solution for the SVM. First we consider the loss view of SVM 8.1.2 and ex-
 6543 press this as an unconstrained optimization problem. Then we express the
 6544 constrained versions of the primal and dual SVMs as quadratic programs
 6545 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

$$g(t) = \begin{cases} -1 & t < 1 \\ [-1, 0] & t = 1 \\ 0 & t > 1 \end{cases}. \quad (12.56)$$

6546 Using this subgradient above, we can apply the optimization methods pre-
 6547 sented in Section 7.1.

6548 Both the primal and the dual SVM result in a convex quadratic pro-
 6549 gramming problem (constrained optimization). Note that the primal SVM
 6550 in (12.27) has optimization variables that have the size of the dimension
 6551 D of the input examples. The dual SVM in (12.43) has optimization vari-
 6552 ables 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

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \quad (12.57)$$

$$\text{subject to } \begin{aligned} -y_n \mathbf{x}_n^\top \mathbf{w} - y_n b - \xi_n &\leq -1 \\ -\xi_n &\leq 0 \end{aligned} \quad (12.58)$$

Recall from
 Section 3.2 that in
 this book, we use
 the phrase dot
 product to mean the
 inner product on
 Euclidean vector
 space.

for all $n = 1, \dots, N$. By concatenating the variables $\mathbf{w}, b, \mathbf{x}_n$ 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 $[\mathbf{w}^\top, b, \xi^\top]^\top \in \mathbb{R}^{D+1+N}$, and we have used

the notation: \mathbf{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:

$$\min_{w,b,\xi} \frac{1}{2} \begin{bmatrix} w \\ b \\ \xi \end{bmatrix}^\top \begin{bmatrix} \mathbf{I}_D & \mathbf{0}_{D,N+1} \\ \mathbf{0}_{N+1,D} & \mathbf{0}_{N+1,N+1} \end{bmatrix} \begin{bmatrix} w \\ b \\ \xi \end{bmatrix} + [\mathbf{0}_{D+1,1} \ C \mathbf{1}_{N,1}]^\top \begin{bmatrix} w \\ b \\ \xi \end{bmatrix} \quad (12.59)$$

$$\text{subject to } \begin{bmatrix} -\mathbf{Y}\mathbf{X} & -\mathbf{y} & -\mathbf{I}_N \\ \mathbf{0}_{N,D+1} & & -\mathbf{I}_N \end{bmatrix} \begin{bmatrix} w \\ b \\ \xi \end{bmatrix} \leq \begin{bmatrix} -\mathbf{1}_{N,1} \\ \mathbf{0}_{N,1} \end{bmatrix}, \quad (12.60)$$

where \mathbf{y} is the vector of labels $[y_1, \dots, y_N]^\top$, $\mathbf{Y} = \text{diag}(\mathbf{y})$ is an N by N matrix where the elements of the diagonal are from \mathbf{y} , and $\mathbf{X} \in \mathbb{R}^{N \times D}$ 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 \mathbf{K} such that each entry is $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Or if we are using an explicit feature representation $K_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. For convenience of notation we introduce a matrix with zeros everywhere except on the diagonal, where we store the labels, that is, $\mathbf{Y} = \text{diag}(\mathbf{y})$. The dual SVM can be written as

$$\min_{\alpha} \frac{1}{2} \alpha^\top \mathbf{Y} \mathbf{K} \mathbf{Y} \alpha - \mathbf{1}_{N,1}^\top \alpha \quad (12.61)$$

$$\text{subject to } \begin{bmatrix} \mathbf{y}^\top \\ -\mathbf{y}^\top \\ -\mathbf{I}_N \\ \mathbf{I}_N \end{bmatrix} \alpha \leq \begin{bmatrix} \mathbf{0}_{N+2,1} \\ C \mathbf{1}_{N,1} \end{bmatrix}. \quad (12.62)$$

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.,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{is replaced by} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \quad \text{and} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} \quad (12.63)$$

Particular software implementations of convex optimization methods may provide the ability to express equality constraints. \diamond

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.

6608 A common non-linear function is the sigmoid function (Bishop, 2006).
6609 When the non-linearity results in well calibrated probabilities (Gneiting
6610 and Raftery, 2007; Reid and Williamson, 2011), this is called class proba-
6611 bility estimation. The third output of a binary classifier is the final binary
6612 decision, which is the one most commonly assumed to be the output of
6613 the classifier.