Imperial College
London

LECTURE NOTES

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Mathematics for Inference and Machine Learning

Instructors:
Marc Deisenroth, Stefanos Zafeiriou

Version: October 11, 2016

Contents

Introduction

These lecture notes support the course "Mathematics for Inference and Machine Learning" in the Department of Computing at Imperial College London. The aim of the course is to provide students the basic mathematical background and skills necessary to understand, design and implement modern statistical machine learning methodologies as well as inference mechanisms. The course will focus on examples regarding the use of mathematical tools for the design of basic machine learning and inference methodologies, such as Principal Component Analysis (PCA), Bayesian Regression and Support Vector Machines (SVMs).
This course is a hard pre-requisite for the following courses:

- CO-424H: Learning in Autonomous Systems
- CO-433: Advanced Robotics
- CO-493: Data Analysis and Probabilistic Inference[1]
- CO-495: Advanced Statistical Machine Learning and Pattern Recognition

and a soft pre-requisite for CO-477.
Relevant for the exam are

- The lecture notes
- The following chapters from the book by Bishop (2006):
  - Chapter 1
  - Chapter 2.2–2.3
  - Chapter 3
  - Chapter 8.1
  - Chapter 12.1–12.2
- The following chapters from the book by Golub and Van Loan (2012):
  - Chapter 1: Matrix Multiplication (without 1.5 and 1.6)
  - Chapter 2: Matrix Analysis (without 2.6 and 2.7)

[1] MSc Computing Science students are exempt from this hard constraint, but they are required to know the material of this course.

  - Chapter 5: 5.2 The QR Factorisation using only the Gram-Schmidt process

- The papers by Turk and Pentland (1991); Belhumeur et al. (1997)
- The following tutorials:
  - Tutorial by Burges (1998) until Section 4.4
  - From the tutorial by Lin (2006) only the basic topics.

Chapter 1

Part I: Linear Regression

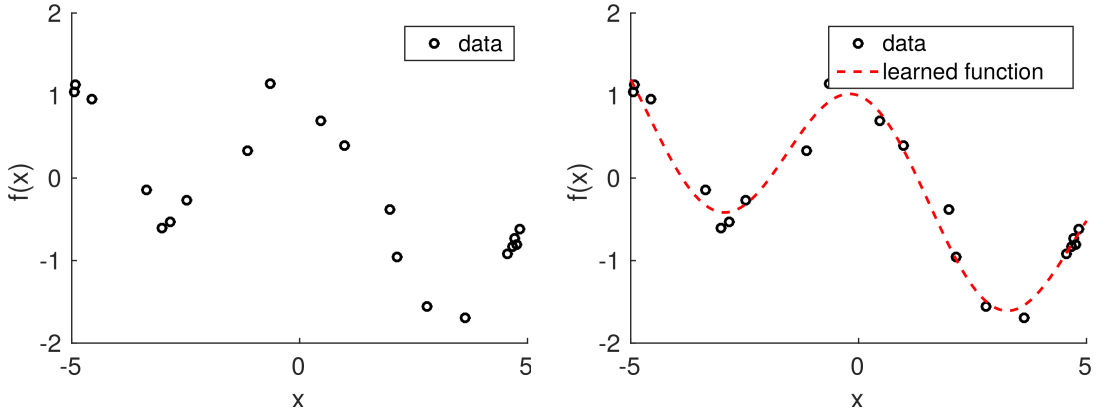
In this part of the course, we will be looking at regression problems, where we want to find a function f that maps inputs $\boldsymbol{x} \in \mathbb{R}^D$ to corresponding function values $f(\boldsymbol{x}) \in \mathbb{R}$ based on noisy observations $y = f(\boldsymbol{x}) + \epsilon$, where ϵ is a random variable.
An example is given in Fig. 1.1. A typical regression problem is given in Fig. 1.1(a): For some values x we observe (noisy) function values $y = f(x) + \epsilon$. The task is to infer the function f that generated the data. A possible solution is given in Fig. 1.1(b).
Regression is a fundamental problem, and regression problems appear in a diverse range of research areas and applications, including time-series analysis (e.g., system identification), control and robotics (e.g., reinforcement learning, forward/inverse model learning), optimization (e.g., line searches, global optimization), deep-learning applications (e.g., computer games, speech-to-text translation, image recognition, automatic video annotation).
Finding a regression function requires solving a variety of problems, including

- **Choice of the parametrization** of the regression function. Given a data set, what function classes (e.g., polynomials) are good candidates for modeling the data, and what particular parametrization (e.g., degree of the polynomial) should we choose?

- **Finding good parameters.** Having chosen a parametrization of the regression function, how do we find the parameter values? Here, we will need to look at different loss functions (they determine what a "good" fit is) and optimization algorithms that allow us to minimize this loss function.

- **Probabilistic models.** Loss functions are often motivated by probabilistic models. We will derive a set of useful models and discuss the corresponding underlying assumptions.

- **Overfitting and model selection.** Overfitting is a problem when the regression function fits the training data "too well" but does not generalize to the test data. Overfitting typically occurs if the underlying model (or its parametrization) is overly flexible and expressive. Model selection allows us to compare various models to find the simplest model that explains the training data reasonably well.



(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.

Figure 1.1: (a) Regression problem and (b) possible solution.

- **Uncertainty modeling.** In any practical setting, we have access to only a finite, potentially large, amount of (training) data for selecting the model class and the corresponding parameters. Given that this finite amount of training data does not cover all possible scenarios, we need to model the remaining uncertainty to obtain a measure of confidence of the model's prediction at test time. The smaller the training set the more important uncertainty modeling. Consistent modeling of uncertainty equips model predictions with confidence bounds.

In this chapter, we will be reviewing the necessary mathematical background for solving regression problems. This includes a brief introduction to probabilities and graphical models, which are useful to visualize relationships between random variables. Furthermore, we will go through some vector calculus, which is required for gradient-based optimization in the context of parameter learning. Once we know how to learn regression functions, we will discuss the problem of overfitting and model selection: Assuming we have a set of "realistic" models, are there some models that are better than others? Toward the end of this chapter, we will then discuss Bayesian linear regression, which allows us to reason about parameters at a higher level, thereby removing some of the problems encountered in maximum likelihood and MAP estimation.

1.1 Problem Formulation

We consider the regression problem

$$y = f(\boldsymbol{x}) + \epsilon, \tag{1.1}$$

where $\boldsymbol{x} \in \mathbb{R}^D$ are inputs and $y \in \mathbb{R}$ are observed targets. Furthermore, $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ is independent, identically distributed (i.i.d.) measurement noise. In this particular

case, ϵ is Gaussian distributed with mean 0 and variance σ^2. The objective is to find a function f that is close to the unknown function that generated the data.
In this course, we focus on parametric models, i.e., we choose a parametrization of f and find parameters that "work well". In the most simple case, the parametrization of linear regression is

$$y = f(\boldsymbol{x}) + \epsilon = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon \tag{1.2}$$

where $\boldsymbol{\theta} \in \mathbb{R}^D$ are the parameters we seek and $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$.[1]
In this course, we will discuss in more detail how to

- Find good parameters $\boldsymbol{\theta}$
- Evaluate whether a parameter set "works well"

We will start by introducing some background material that is necessary and useful: concepts in probability theory, standard probability distributions, and probabilistic graphical models.

1.2 Probabilities

Probability theory is a mathematical foundation of statistics and a consistent way of expressing uncertainty. Jaynes (2003) provides a great introduction to probability theory.

Definition 1 (Probability Density Function)
A function $p : \mathbb{R}^D \to \mathbb{R}$ is called a **probability density function** *if (1) its integral exists, (2) $\forall \boldsymbol{x} \in \mathbb{R}^D : p(\boldsymbol{x}) \geq 0$ and (3)*

$$\int_{\mathbb{R}^D} p(\boldsymbol{x}) d\boldsymbol{x} = 1\,. \tag{1.3}$$

Here, $\boldsymbol{x} \in \mathbb{R}^D$ is a (continuous) **random variable**.[2] *For discrete random variables, the integral in* (1.3) *is replaced with a sum.*

Remark 1
We will be using the expression "probability distribution" not only for discrete distributions but also for continuous probability density functions, although this is technically incorrect.

There are two fundamental rules in probability theory that appear everywhere in machine learning and Bayesian statistics:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \qquad \textbf{Sum rule/Marginalization property} \tag{1.4}$$

[1] It would be more precise to call this model "linear in the parameters". We will see later that $\Phi^\top(\boldsymbol{x})\boldsymbol{\theta}$ for nonlinear transformations Φ is also a linear regression model.
[2] We omit the definition of a random variable as this will become too technical for the purpose of this course.

$$p(\boldsymbol{x},\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \qquad \textbf{Product rule} \tag{1.5}$$

Here, $p(\boldsymbol{x},\boldsymbol{y})$ is the **joint distribution** of the two random variables $\boldsymbol{x},\boldsymbol{y}$, $p(\boldsymbol{x}), p(\boldsymbol{y})$ are the corresponding **marginal distributions**, and $p(\boldsymbol{y}|\boldsymbol{x})$ is the **conditional distribution** of $\boldsymbol{y}$ given $\boldsymbol{x}$. If we consider discrete random variables $\boldsymbol{x},\boldsymbol{y}$, the integral in (1.4) is replaced by a sum. This is where the name comes from.
In machine learning and Bayesian statistics, we are often interested in making inferences of random variables given that we have observed other random variables. Let us assume, we have some prior knowledge $p(\boldsymbol{x})$ about a random variable $\boldsymbol{x}$ and some relationship $p(\boldsymbol{y}|\boldsymbol{x})$ between $\boldsymbol{x}$ and a second random variable $\boldsymbol{y}$. If we now observe $\boldsymbol{y}$, we can use Bayes' theorem[3] to draw some conclusions about $\boldsymbol{x}$ given the observed values of $\boldsymbol{y}$. **Bayes' theorem** follows immediately from the sum and product rules in (1.4)–(1.5) as

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})}. \tag{1.6}$$

Here, $p(\boldsymbol{x})$ is the **prior**, which encapsulates our prior knowledge of $\boldsymbol{x}$, $p(\boldsymbol{y}|\boldsymbol{x})$ is the **likelihood**[4] that describes how $\boldsymbol{y}$ and $\boldsymbol{x}$ are related. The quantity $p(\boldsymbol{y})$ is the **marginal likelihood** or **evidence** and is a normalizing constant (independent of $\boldsymbol{x}$), which is obtained as the integral $\int p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$ of the numerator with respect to $\boldsymbol{x}$ and ensures that the fraction is normalized. The **posterior** $p(\boldsymbol{x}|\boldsymbol{y})$ expresses exactly what we are interested in, i.e., what we know about $\boldsymbol{x}$ if we observe $\boldsymbol{y}$.

Remark 2 (Marginal Likelihood)
Thus far, we looked at the marginal likelihood simply as a normalizing constant that ensures that the posterior probability distribution integrates to 1. In Section 1.7 we will see that the marginal likelihood also plays an important role in model selection.

1.2.1 Means and Covariances

Mean and (co)variance are often useful to describe properties of probability distributions (expected values and spread).

Definition 2 (Mean and Covariance)
The **mean** *of a random variable $\boldsymbol{x} \in \mathbb{R}^D$ is defined as*

$$\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] = \int \boldsymbol{x}p(\boldsymbol{x})d\boldsymbol{x} = \begin{bmatrix}\mathbb{E}[x_1]\\ \vdots \\ \mathbb{E}[x_D]\end{bmatrix} \in \mathbb{R}^D, \tag{1.7}$$

where the subscript indicates the corresponding dimension of $\boldsymbol{x}$.
If we consider two random variables $\boldsymbol{x} \in \mathbb{R}^D, \boldsymbol{y} \in \mathbb{R}^E$, the **covariance** *between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as*

$$\mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}\boldsymbol{y}^\top] - \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]\mathbb{E}_{\boldsymbol{y}}[\boldsymbol{y}]^\top = \mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}]^\top \in \mathbb{R}^{D\times E}. \tag{1.8}$$

[3] Also called the "probabilistic inverse"
[4] Also called the "measurement model"

Here, the subscript makes it explicit with respect to which variable we need to average. The **variance** *of a random variable* $\boldsymbol{x} \in \mathbb{R}^D$ *with mean vector* $\boldsymbol{\mu}$ *is defined as*

$$\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}] = \mathbb{E}_{\boldsymbol{x}}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top] = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}\boldsymbol{x}^\top] - \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]^\top \tag{1.9}$$

$$= \begin{bmatrix} \mathrm{Cov}[x_1,x_1] & \mathrm{Cov}[x_1,x_2] & \dots & \mathrm{Cov}[x_1,x_D] \\ \mathrm{Cov}[x_2,x_1] & \mathrm{Cov}[x_2,x_2] & \dots & \mathrm{Cov}[x_2,x_D] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[x_D,x_1] & \dots & \dots & \mathrm{Cov}[x_D,x_D] \end{bmatrix} \in \mathbb{R}^{D\times D}. \tag{1.10}$$

This matrix is called the **covariance matrix** *of the random variable* $\boldsymbol{x}$*. The covariance matrix is symmetric and positive definite and tells us something about the spread of the data.*
The covariance matrix contains the variances of the marginals $p(x_i) = \int p(x_1,\dots,x_D)d\boldsymbol{x}_{\setminus i}$ *on its diagonal, where "*$\setminus i$*" denotes "all variables but* i*". The off-diagonal terms contain the* **cross-covariance** *terms* $\mathrm{Cov}[x_i,x_j]$ *for* $i,j = 1,\dots,D, i \neq j$*.*

It generally holds that

$$\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}] = \mathrm{Cov}_{\boldsymbol{x}}[\boldsymbol{x},\boldsymbol{x}]. \tag{1.11}$$

1.2.1.1 Sum of Random Variables

Consider two random variables $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^D$. It holds that

$$\mathbb{E}[\boldsymbol{x}+\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}]+\mathbb{E}[\boldsymbol{y}] \tag{1.12}$$
$$\mathbb{E}[\boldsymbol{x}-\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}]-\mathbb{E}[\boldsymbol{y}] \tag{1.13}$$
$$\mathbb{V}[\boldsymbol{x}+\boldsymbol{y}] = \mathbb{V}[\boldsymbol{x}]+\mathbb{V}[\boldsymbol{y}]+\mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}]+\mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}] \tag{1.14}$$
$$\mathbb{V}[\boldsymbol{x}-\boldsymbol{y}] = \mathbb{V}[\boldsymbol{x}]+\mathbb{V}[\boldsymbol{y}]-\mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}]-\mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}] \tag{1.15}$$

1.2.1.2 Affine Transformation

Mean and (co)variance exhibit some useful properties when it comes to affine transformation of random variables[5]. Consider a random variable $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance matrix Σ and a (deterministic) affine transformation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$ of $\boldsymbol{x}$. Then, $\boldsymbol{y}$ is itself a random variable whose mean vector and covariance matrix are given by

$$\mathbb{E}_{\boldsymbol{y}}[\boldsymbol{y}] = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}] = \boldsymbol{A}\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}]+\boldsymbol{b} = \boldsymbol{A}\boldsymbol{\mu}+\boldsymbol{b}, \tag{1.16}$$
$$\mathbb{V}_{\boldsymbol{y}}[\boldsymbol{y}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}] = \mathbb{V}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x}] = \boldsymbol{A}\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}]\boldsymbol{A}^\top = \boldsymbol{A}\Sigma\boldsymbol{A}^\top, \tag{1.17}$$

respectively.[6] Furthermore,

$$\mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] = \int \boldsymbol{x}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})^\top p(\boldsymbol{x})d\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}]^\top \tag{1.18}$$

[5] The proof is left as an exercise.
[6] This can be shown directly by using the definition of the mean and covariance.

$$= \int \boldsymbol{x} p(\boldsymbol{x}) d\boldsymbol{x} \boldsymbol{b}^\top + \int \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x} \boldsymbol{A}^\top - \boldsymbol{\mu}\boldsymbol{b}^\top - \boldsymbol{\mu}\boldsymbol{\mu}^\top \boldsymbol{A}^\top \tag{1.19}$$

$$= \boldsymbol{\mu}\boldsymbol{b}^\top - \boldsymbol{\mu}\boldsymbol{b}^\top + \Big(\int \boldsymbol{x}\boldsymbol{x}^\top p(\boldsymbol{x}) d\boldsymbol{x} - \boldsymbol{\mu}\boldsymbol{\mu}^\top \Big) \boldsymbol{A}^\top \tag{1.20}$$

$$\overset{(1.9)}{=} \boldsymbol{\Sigma}\boldsymbol{A}^\top \tag{1.21}$$

1.2.2 Statistical Independence

Definition 3 (Independence)
Two random variables $\boldsymbol{x}, \boldsymbol{y}$ are **statistically independent** *if and only if*

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x}) p(\boldsymbol{y}). \tag{1.22}$$

Intuitively, two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent if the value of $\boldsymbol{y}$ (once known) does not add any additional information about $\boldsymbol{x}$ (and vice versa).
If $\boldsymbol{x}, \boldsymbol{y}$ are (statistically) independent then

- $p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{y})$
- $p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{x})$
- $\mathbb{V}[\boldsymbol{x} + \boldsymbol{y}] = \mathbb{V}[\boldsymbol{x}] + \mathbb{V}[\boldsymbol{y}]$
- $\mathrm{Cov}[\boldsymbol{x}, \boldsymbol{y}] = \boldsymbol{0}$

Another concept that is important in machine learning is conditional independence.

Definition 4 (Conditional Independence)
Formally, $\boldsymbol{x}$ and $\boldsymbol{y}$ are **conditionally independent given $\boldsymbol{z}$** *if and only if*

$$p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{y}|\boldsymbol{z}). \tag{1.23}$$

We write $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y}|\boldsymbol{z}$.

Independence can be cast as a special case of conditional independence if we write $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y}|\emptyset$.

1.2.3 Basic Probability Distributions

In the following, we will briefly introduce basic probability distributions.

1.2.3.1 Uniform Distribution

The uniform distribution is a distribution that assigns equal probability mass in a region. For $a, b \in \mathbb{R}$ and $a < b$, the uniform distribution is defined as

$$\mathcal{U}[a, b] = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{1.24}$$

Mean and variance of the uniform distribution $\mathcal{U}[a, b]$ are $\frac{1}{2}(a + b)$ and $\frac{1}{12}(b - a)^2$, respectively.

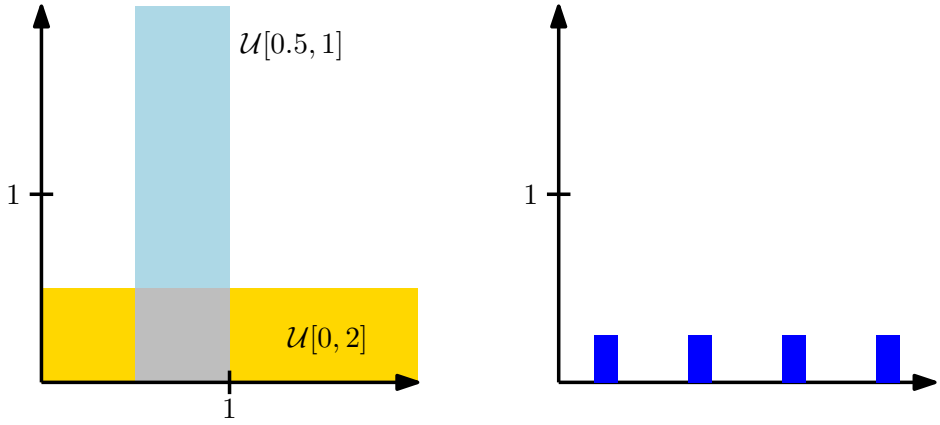


Figure 1.2: Examples of uniform distributions. Left: Continuous uniform distribution that distributes probability mass equally everywhere in a (compact) region. Right: Discrete uniform distribution that assigns equal probability to four possible (discrete) events.

Figure 1.3: The Bernoulli distribution can be used to model the binary outcome probability of a coin flip experiment.

1.2.3.2 Bernoulli Distribution

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

$$p(x|\mu) = \mu^x (1 - \mu)^{1-x}, \quad x \in \{0, 1\}, \tag{1.25}$$

$$\mathbb{E}[x] = \mu, \tag{1.26}$$

$$\mathbb{V}[x] = \mu(1 - \mu), \tag{1.27}$$

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.

1.2.3.3 Binomial Distribution

The Binomial distribution is a generalization of the Bernoulli distribution to a distribution over integers. In particular, the Binomial can be used to describe the probability of observing m occurrences of $x = 1$ in a set of N samples from a Bernoulli

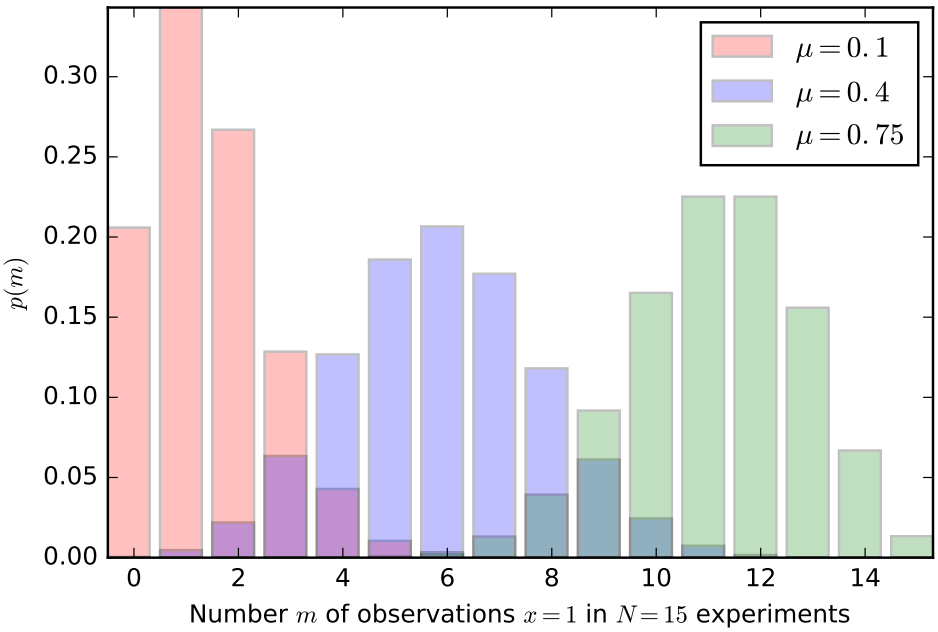


Figure 1.4: Examples of the Binomial distribution for $\mu \in \{0.1, 0.4, 0.75\}$ and $N = 15$.

distribution where $p(x = 1) = \mu \in [0, 1]$. The Binomial distribution is defined as

$$p(m|N, \mu) = \binom{N}{m} \mu^m (1 - \mu)^{N-m}, \tag{1.28}$$

$$\mathbb{E}[m] = N\mu, \tag{1.29}$$

$$\mathbb{V}[m] = N\mu(1 - \mu) \tag{1.30}$$

where $\mathbb{E}[m]$ and $\mathbb{V}[m]$ are the mean and variance of m, respectively.
An example where the Binomial could be used is if we want to describe the probability of observing m "heads" in N coin-flip experiments if the probability for observing head in a single experiment is μ?

1.2.3.4 Beta Distribution

The Beta distribution is a distribution over a continuous variable $\mu \in [0, 1]$, which is often used to represent the probability for some binary event (e.g., the parameter governing the Bernoulli distribution). The Beta distribution itself is governed by two parameters $\alpha > 0,\ \beta > 0$ and is defined as

$$p(\mu|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1} (1 - \mu)^{\beta-1} \tag{1.31}$$

$$\mathbb{E}[\mu] = \frac{\alpha}{\alpha + \beta}, \qquad \mathbb{V}[\mu] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{1.32}$$

where $\Gamma(\cdot)$ is the Gamma function defined as

$$\Gamma(t) := \int_0^\infty x^{t-1} \exp(-x) dx, \qquad t > 0. \tag{1.33}$$

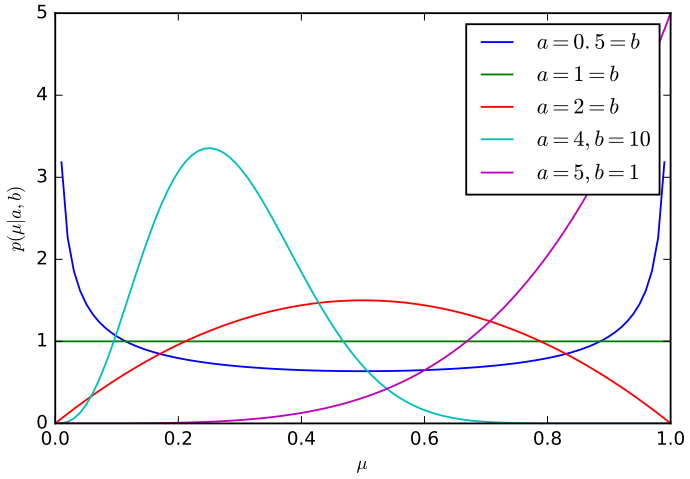


Figure 1.5: Examples of the Beta distribution for different values of α and β.

$$\Gamma(t+1) = t\Gamma(t). \tag{1.34}$$

Note that the fraction of Gamma functions in (1.31) normalizes the Beta distribution. Intuitively, α moves probability mass toward 1, whereas β moves probability mass toward 0. There are some special cases (Murphy, 2012):

- For $\alpha = 1 = \beta$ we obtain the uniform distribution $\mathcal{U}[0, 1]$.
- For $\alpha, \beta < 1$, we get a bimodal distribution with spikes at 0 and 1.
- For $\alpha, \beta > 1$, the distribution is unimodal.
- For $\alpha, \beta > 1$ and $\alpha = \beta$, the distribution is unimodal, symmetric and centered in the interval $[0, 1]$, i.e., the mode/mean is at $\frac{1}{2}$.

1.2.3.5 Gaussian Distribution

The **multivariate Gaussian distribution**[7] is fully characterized by a **mean vector** $\boldsymbol{\mu}$ and a **covariance matrix** Σ and defined as

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \Big), \tag{1.35}$$

where $\boldsymbol{x} \in \mathbb{R}^D$ is a random variable. We write $\boldsymbol{x} \sim \mathcal{N}\big(\boldsymbol{x} \,|\, \boldsymbol{\mu}, \Sigma\big)$ or $\boldsymbol{x} \sim \mathcal{N}\big(\boldsymbol{\mu}, \Sigma\big)$.
The Gaussian distribution is the most important probability distribution[8] for continuous-valued random variables. Its importance originates from the fact that it has many computationally convenient properties, which we will be discussing in the following.

[7] Also: Multivariate normal distribution

[8] We will be adopting a common, but mathematically slightly sloppy, language and call the "probability density function" a "distribution".

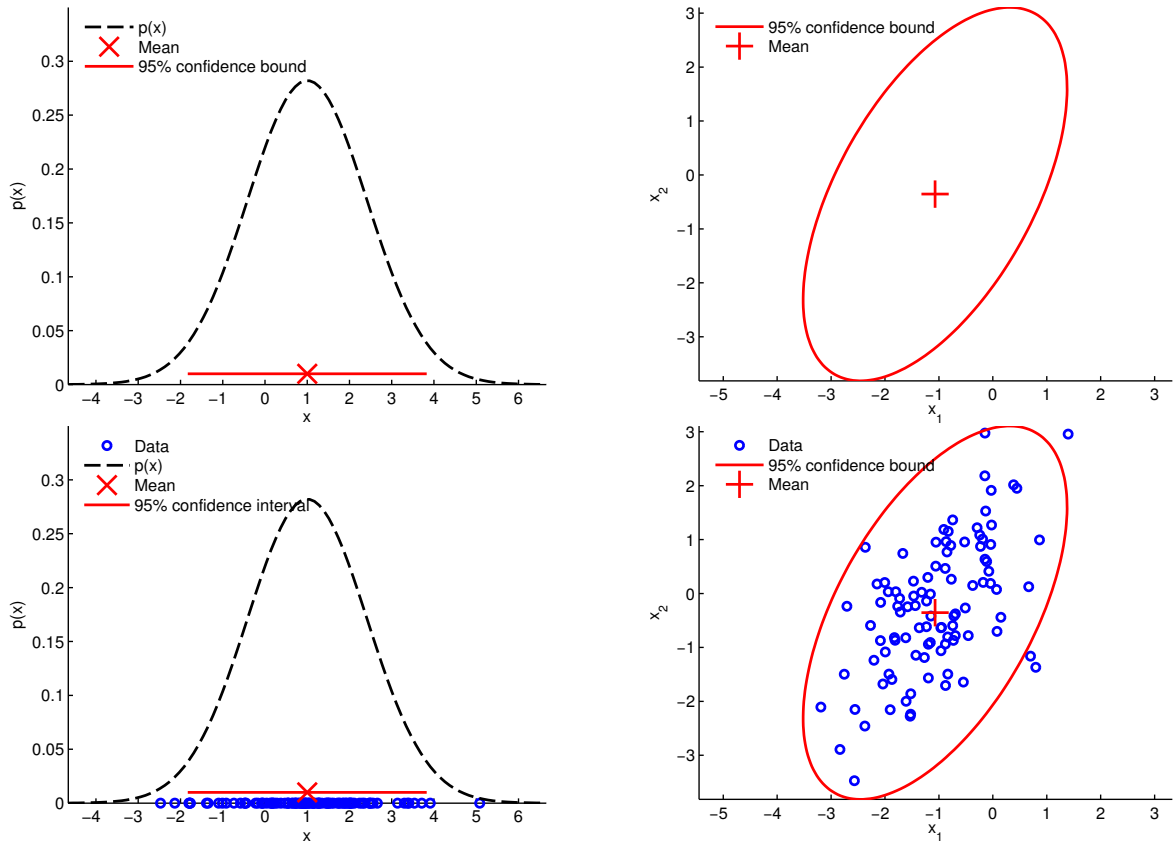


Figure 1.6:

Application areas in which the Gaussian plays a central role range from signal processing (e.g., Kalman filter) to control (e.g., linear quadratic regulator) and machine learning (e.g., Gaussian processes, principal component analysis, clustering with Gaussian mixture models and k-means, linear regression, deep learning with squared errors, variational inference, reinforcement learning).

Conditional and Marginal Consider the joint Gaussian distribution

$$p(\boldsymbol{x},\boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right) \tag{1.36}$$

of two random variables $\boldsymbol{x},\boldsymbol{y}$, where $\Sigma_{xx} = \mathrm{Cov}[\boldsymbol{x},\boldsymbol{x}]$ and $\Sigma_{yy} = \mathrm{Cov}[\boldsymbol{y},\boldsymbol{y}]$ are the marginal covariance matrices of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and $\Sigma_{xy} = \mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}]$ is the cross-covariance matrix between $\boldsymbol{x}$ and $\boldsymbol{y}$.
The conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$ is also Gaussian and given by

$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{\mu}_{x|y}, \Sigma_{x|y}\right) \tag{1.37}$$

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y) \tag{1.38}$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}. \tag{1.39}$$

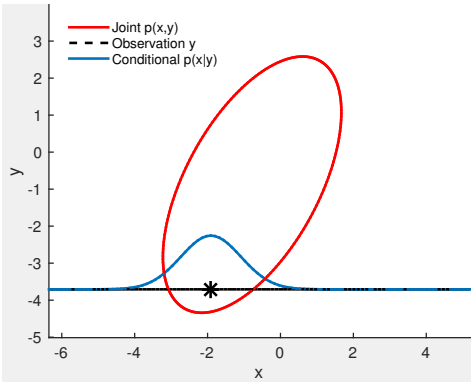


Figure 1.7: The conditional distribution of a Gaussian is also Gaussian

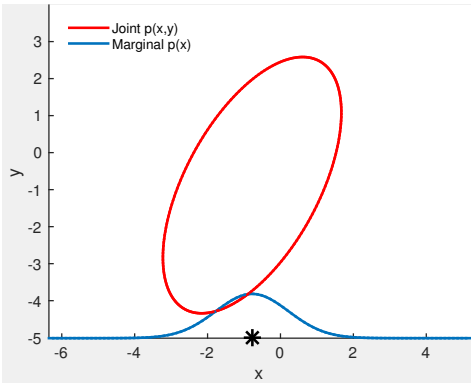


Figure 1.8: Marginal of a joint Gaussian distribution is Gaussian.

Note that in the computation of the mean in (1.38) the $\boldsymbol{y}$-value is an observation and no longer random.

Remark 3
The conditional Gaussian distribution shows up in many places, where we are interested in posterior distributions:

- *The Kalman filter (Kalman, 1960), one of the most central algorithms for state estimation in signal processing, does nothing but computing Gaussian conditionals of joint distributions (Deisenroth and Ohlsson, 2011).*
- *Gaussian processes (Rasmussen and Williams, 2006), which are a practical implementation of a distribution over functions. In a Gaussian process, we make assumptions of joint Gaussianity of random variables. By (Gaussian) conditioning on observed data, we can determine a posterior distribution over functions.*
- *Latent linear Gaussian models (Roweis and Ghahramani, 1999; Murphy, 2012), which include probabilistic PCA (Tipping and Bishop, 1999).*

The marginal distribution $p(\boldsymbol{x})$[9] of a joint Gaussian distribution $p(\boldsymbol{x},\boldsymbol{y})$, see (1.36), is

[9]The same holds for $p(\boldsymbol{y})$.

itself Gaussian and computed by applying the sum-rule in (1.4) and given by

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} = \mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{\mu}_x, \Sigma_{xx}\right). \tag{1.40}$$

Intuitively, looking at the joint distribution in (1.36), we ignore (i.e., integrate out) everything we are not interested in.

Product of Gaussians The **product** of two Gaussians $\mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{a}, \boldsymbol{A}\right)\mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{b}, \boldsymbol{B}\right)$ is an unnormalized Gaussian distribution $c\,\mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{c}, \boldsymbol{C}\right)$ with

$$\boldsymbol{C} = (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1} \tag{1.41}$$

$$\boldsymbol{c} = \boldsymbol{C}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}) \tag{1.42}$$

$$c = (2\pi)^{-\frac{D}{2}} |\boldsymbol{A} + \boldsymbol{B}|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{a} - \boldsymbol{b})^\top (\boldsymbol{A} + \boldsymbol{B})^{-1}(\boldsymbol{a} - \boldsymbol{b})\right). \tag{1.43}$$

Note that the normalizing constant c itself is a Gaussian either in $\boldsymbol{a}$ or in $\boldsymbol{b}$ with an "inflated" covariance matrix $\boldsymbol{A} + \boldsymbol{B}$, i.e., $c = \mathcal{N}\left(\boldsymbol{a} \,|\, \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B}\right) = \mathcal{N}\left(\boldsymbol{b} \,|\, \boldsymbol{a}, \boldsymbol{A} + \boldsymbol{B}\right)$.

Linear Transformation of Gaussian Random Variables A linear (or affine) transformation of a Gaussian random variable is Gaussian distributed.

- If $p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \,|\, \boldsymbol{\mu}, \Sigma\right)$ and $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ then $p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y} \,|\, \boldsymbol{A}\boldsymbol{\mu}, \boldsymbol{A}\Sigma\boldsymbol{A}^\top\right)$.
- If $p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y} \,|\, \boldsymbol{A}\boldsymbol{x}, \Sigma\right)$ then we can re-write this as a probability distribution in $\boldsymbol{x}$: If $\boldsymbol{A}$ was invertible, we could write $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$. However, $\boldsymbol{A}$ is not generally invertible. Therefore, we perform a transformation with $\boldsymbol{A}^\top$:

  $$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \Leftrightarrow \boldsymbol{A}^\top\boldsymbol{y} = \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x}. \tag{1.44}$$

  $\boldsymbol{A}^\top\boldsymbol{A}$ is symmetric and positive definite[10], i.e., it can be inverted. Then,

  $$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \Leftrightarrow (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{y} = \boldsymbol{x}. \tag{1.45}$$

  Hence, $\boldsymbol{x}$ is a linear transformation of $\boldsymbol{y}$, and we obtain

  $$p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x} \,|\, (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{y}, (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\Sigma\boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\right). \tag{1.46}$$

Sampling from Multivariate Gaussian Distributions Assume we are interested in generating samples $\boldsymbol{x}_i, i = 1, \ldots, n$, from a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ. However, we only have access to a sampler that allows us to generate samples from the standard normal $\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right)$.

To obtain samples from a multivariate normal $\mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$, we can use the properties of a linear transformation of a Gaussian random variable: If $\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}\right)$ then $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\mu}$, where $\boldsymbol{A}\boldsymbol{A}^\top = \Sigma$, is Gaussian distributed with mean $\boldsymbol{\mu}$ and covariance matrix Σ. We call $\Sigma = \boldsymbol{A}\boldsymbol{A}^\top$ the **Cholesky factorization** of Σ.[11]

[10] Actually, only positive semi-definite, but with mild assumptions we arrive at positive definite.

[11] To compute the Cholesky factorization of a matrix, it is required that the matrix is symmetric and positive definite. Covariance matrices possess this property.

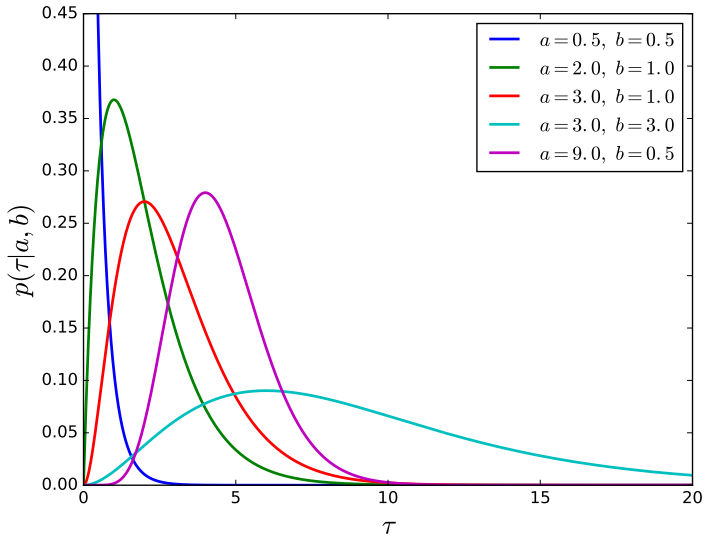


Figure 1.9: Gamma distribution for different values of a, b.

1.2.3.6 Gamma Distribution

The Gamma distribution is a distribution over positive real numbers $\tau > 0$. The Gamma distribution is itself governed by two parameters $a, b > 0$, where a is a shape parameter and b a scale parameter of the corresponding density.
The Gamma distribution is defined as

$$p(\tau|a,b) = \frac{1}{\Gamma(a)} b^a \tau^{a-1} \exp(-b\tau), \tag{1.47}$$

$$\mathbb{E}[\tau] = \frac{a}{b}, \tag{1.48}$$

$$\mathbb{V}[\tau] = \frac{a}{b^2}. \tag{1.49}$$

The Gamma distribution is often used as a prior on the precision (inverse variance) of a univariate Gaussian distribution.

Remark 4
Other distributions are special cases of the Gamma distribution (Murphy, 2012): The Exponential distribution with parameter λ is obtained for $(a,b) = (1,\lambda)$. The exponential distribution describes the time between events in a Poisson process. The Erlang distribution is a Gamma distribution with $a \in \mathbb{N}$. In the Chi-Squared distribution, which is the distribution of the sum of Gaussian random variables, the scale parameter b is fixed to $b = \frac{1}{2}$.

1.2.3.7 Wishart Distribution

The Wishart distribution is the multivariate generalization of the Gamma distribution: It is a family of probability distributions defined over symmetric, nonnegative-definite matrix-valued random variables ("random matrices").
For $D \times D$ matrices the **Wishart distribution** is defined as

$$\mathcal{W}(\Sigma|\boldsymbol{W},\nu) = B|\Sigma|^{\frac{\nu-D-1}{2}} \exp\Big(-\tfrac{1}{2}\mathrm{tr}(\boldsymbol{W}^{-1}\Sigma)\Big) \tag{1.50}$$

where ν is called the **number of degrees of freedom** of the distribution, and $\boldsymbol{W}$ is a $d \times D$ scale matrix. B is a normalization constant (Bishop, 2006, p. 102) that ensures that the distribution is normalized.
These distributions are of great importance in the estimation of covariance matrices in multivariate statistics: The Wishart distribution is the conjugate prior for the precision matrix (inverse covariance matrix) of a Gaussian-distributed random variable $\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$.

1.2.4 Conjugacy

According to Bayes' theorem (1.6), the posterior is proportional to the product of the prior and the likelihood. The specification of the prior can be tricky for two reasons: First, the prior should encapsulate our knowledge about the problem before we see some data. This is often difficult to describe. Second, it is often not possible to compute the posterior distribution analytically. However, there are some priors that are computationally convenient: **conjugate priors**.

Definition 5 (Conjugate Prior)
A prior is **conjugate** *for the likelihood function if the posterior is of the same form as the prior.*

Example (Beta-Binomial Conjugacy)
Consider a Binomial random variable $x \sim \text{Bin}(m|N, \mu)$ where

$$p(x|\mu, N) = \binom{N}{m} \mu^m (1-\mu)^{N-m} \propto \mu^a (1-\mu)^b \tag{1.51}$$

for some constants a, b. We place a Beta prior on the parameter μ:

$$\text{Beta}(\mu|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1}(1-\mu)^{\beta-1} \propto \mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{1.52}$$

If we now observe some outcomes $\boldsymbol{x} = (x_1, \ldots, x_n)$ of a repeated coin-flip experiment with h heads and t tails, we compute the posterior distribution on μ as

$$p(\mu|\boldsymbol{x} = h) \propto p(\boldsymbol{x}|\mu)p(\mu|\alpha, \beta) = \mu^h(1-\mu)^t \mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{1.53}$$

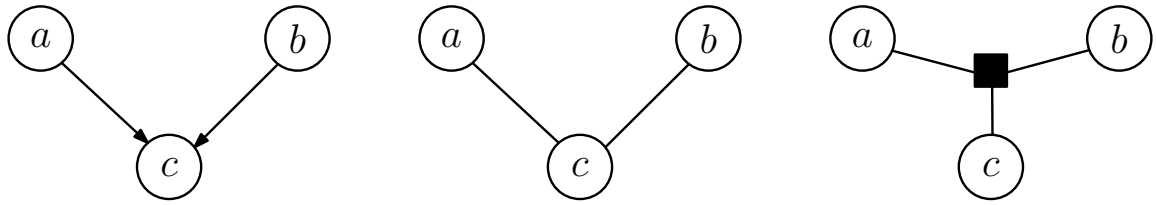
$$= \mu^{h+\alpha-1}(1-\mu)^{t+\beta-1} \propto \text{Beta}(h+\alpha, t+\beta) \tag{1.54}$$

i.e., the posterior distribution is a Beta distribution as the prior, i.e., the Beta prior is conjugate for the parameter μ in the Binomial likelihood function.

Table 1.1 lists examples for conjugate priors for the parameters of some of the standard distributions that we discussed in this section. The Beta distribution is the conjugate prior for the parameter μ in both the Binomial and the Bernoulli likelihood. For a Gaussian likelihood function, we can place a conjugate Gaussian prior

Table 1.1: Standard examples of conjugate priors.

Conjugate prior	Likelihood	Posterior
Beta	Bernoulli	Beta
Beta	Binomial	Beta
Gaussian/inverse Gamma	Gaussian	Gaussian/inverse Gamma
Gaussian/inverse Wishart	Gaussian	Gaussian/inverse Wishart
Dirichlet	Multinomial	Dirichlet



(a) Directed graphical model (b) Undirected graphical model (c) Factor graph

Figure 1.10: Three types of graphical models: (a) Directed graphical models (Bayesian network); (b) Undirected graphical models (Markov random field); (c) Factor graphs.

on the mean. The reason why the Gaussian likelihood appears twice in the table is that we need distinguish the univariate from the multivariate case. In the univariate (scalar) case, the inverse Gamma is the conjugate prior for the variance[12]. In the multivariate case, we use a conjugate inverse Wishart distribution as a prior on the covariance matrix[13]. The Dirichlet distribution is the conjugate prior for the multinomial likelihood function. For further details, we refer to Bishop (2006).

1.3 Probabilistic Graphical Models

A graphical model 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.
There are three main types of graphical models:

- Directed graphical models (Bayesian networks)
- Undirected graphical models (Markov random fields)
- Factor graphs

[12]Alternatively, the Gamma prior is conjugate for the precision (inverse variance) in the Gaussian likelihood.
[13]Alternatively, the Wishart prior is conjugate for the precision matrix (inverse covariance matrix) in the Gaussian likelihood.

Nodes are (random) variables, edges represent probabilistic relations between variables. In this course, we will focus on directed graphical models.[14]
Probabilistic graphical models have some convenient properties:

- They are a simple way to visualize the structure of a probabilistic model
- They can be used to design or motivate new kind of statistical models
- Inspection of the graph alone gives us insight into properties, e.g., conditional independence
- Complex computations for inference and learning in statistical models can be expressed in terms of graphical manipulations.

1.3.1 From Joint Distributions to Graphs

Consider the joint distribution

$$p(a,b,c) = p(c|a,b)p(b|a)p(a) \tag{1.55}$$

of three random variables a, b, c. The factorization of the joint in (1.55) tell 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

We can build the corresponding directed graphical model as follows:

1. Create a node for all random variables
2. For each conditional distribution, we add a directed link (arrow) to the graph from the nodes corresponding to the variables on which the distribution is conditioned on

Note that the graph layout depends on the choice of factorization. For the factorization in (1.55), we obtain the directed graphical model in Fig. 1.11.

1.3.2 From Graphs to Joint Distributions

In the following, we will discuss how to extract the joint distribution of a set of random variables from a given graphical model. We will immediately look at the graphical model in Fig. 1.12, and exploit two observations:

[14] "Data Analysis and Probabilistic Inference" (CO-493) will discuss all three types of graphical models.

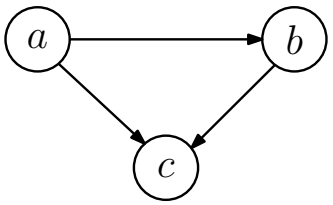


Figure 1.11: Directed graphical model for the factorization of the joint distribution in (1.55).

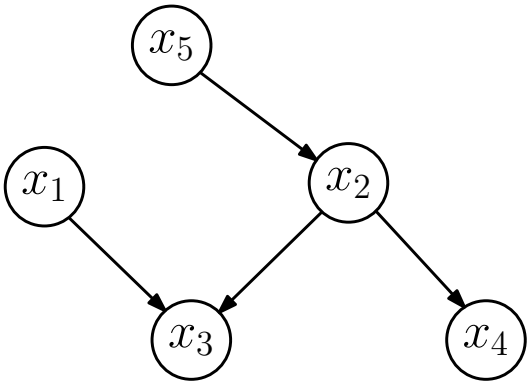


Figure 1.12: Directed graphical model for which we seek the corresponding joint distribution and its factorization.

- 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.

Using these two properties, we arrive at the desired factorization of the joint distribution

$$p(x_1,x_2,x_3,x_4,x_5) = p(x_1)p(x_5)p(x_2|x_5)p(x_3|x_1,x_2)p(x_4|x_2). \tag{1.56}$$

In general, the joint distribution $p(\boldsymbol{x}) = p(x_1,\dots,x_K)$ is given as

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k) \tag{1.57}$$

where pa_k means "the parent nodes of x_k".

Example (Linear Regression)
We seek the graphical model representation for the linear regression setting

$$y_n = \boldsymbol{x}_n^\top \boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}\left(0, \sigma^2\right), \tag{1.58}$$

for $n = 1,\dots,N$, where y_n are observed variables.
In this example, we have three different types of "variables":

- Unknown random variables $\boldsymbol{\theta}$
- Observed random variables y_n
- Deterministic parameters $\boldsymbol{x}_n, \sigma$, which are fixed

To make the distinction between these three types easier, we introduce additional nodes for graphical models:

- Shaded nodes represent observed random variables
- Dots represent deterministic parameters

To find the directed graphical model, for all (observed and unobserved) random variables we write down all probability distributions with explicit conditioning on the parameters/variables they depend on. In our case, we end up with:

- $p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}, \sigma)$
- $p(\boldsymbol{\theta})$

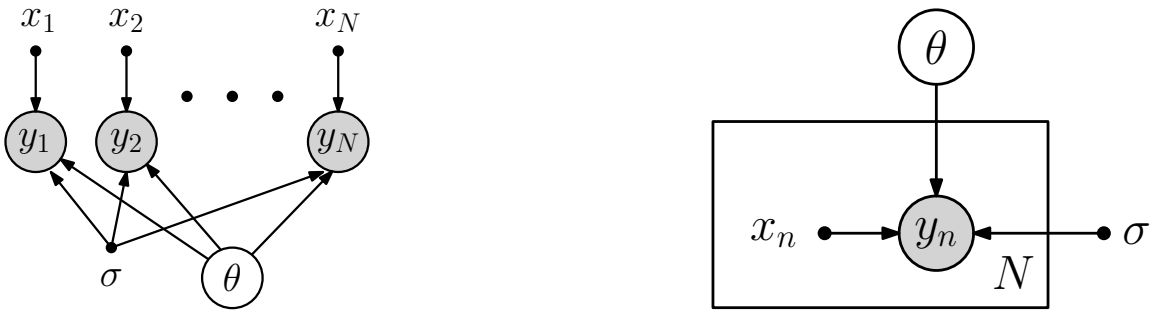
This gives us the joint distribution of all random variables

$$p(y_1, \ldots, y_N, \boldsymbol{\theta}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \sigma) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}, \sigma). \tag{1.59}$$

Now, we follow the steps Section 1.3.1 and find the graphical model in Fig. 1.13(a). Observed random variables are shaded, deterministic parameters are dots, unobserved random variables are "hollow". The graphical model is somewhat repetitive because, and we can write it in a more compact form using the **plate notation** in Fig. 1.13(b). The plate essentially can be read as "for $n = 1, \ldots, N$ locally copy/repeat everything inside the plate". Therefore, the plate replaces the dots in Fig. 1.13(a). Note that the parameter σ for the noise and the random variable $\boldsymbol{\theta}$ are "global" and, therefore, outside the plate.

1.3.3 Further Reading

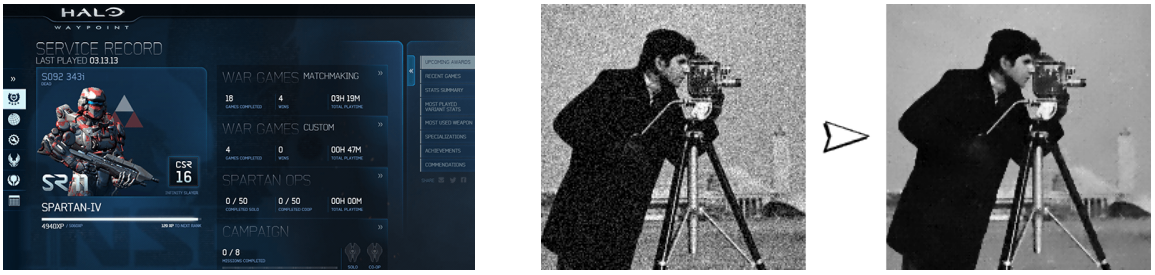
A good and extensive introduction to probabilistic graphical models can be found in the book by Koller and Friedman (2009).
Directed graphical models allow us to find conditional independence relationship properties of the joint distribution only by looking at the graph. A concept called **d-separation** (Pearl, 1988) is key to this. D-separation will be discussed in more detail in "Data Analysis and Probabilistic Inference" (CO-493).
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,



(a) Version 1

(b) Version 2 using the plate notation.

Figure 1.13: Two graphical models for linear regression. Observed random variables are shaded, deterministic parameters are dots. (a) Graphical model without plate notation; (b) Graphical model with plate notation, which allows for a more compact representation than (a).



(a) Online ranking with the TrueSkill system.

(b) Image restoration

Figure 1.14: Examples of message passing using graphical models: (a) Microsoft's TrueSkill system (Herbrich et al., 2007) is used for ranking in online video games. (b) Image restoration (Kittler and Föglein, 1984) is used to remove noise from images.

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).

1.4 Vector Calculus

Now, we return to the linear regression setting outlined in Section 1.1: We are interested in finding "good" parameters for the linear regression model. Finding good parameters can be phrased as an optimization problem[15]. We will use gradient-based approaches for solving this optimization problem. Hence, in this section, we will be looking at differentiation of multi-variate functions with respect to parameter vectors.

[15]We will do this in Section 1.5.

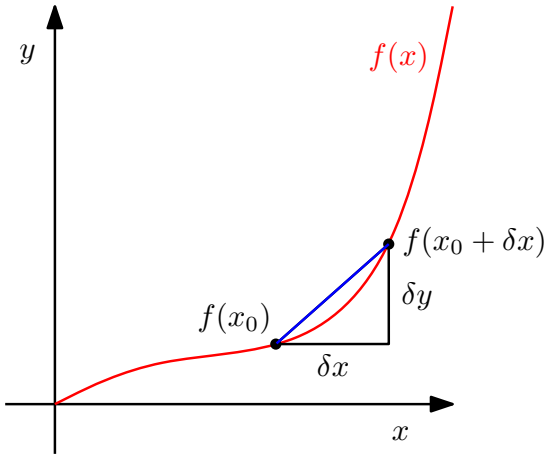


Figure 1.15: The average incline of a function f between x_0 and $x_0 + \delta x$ is the incline of the secant (blue) through $f(x_0)$ and $f(x_0 + \delta x)$ and given by $\delta y / \delta x$.

We start with the **difference quotient** of a univariate function $y = f(x),\ x, y \in \mathbb{R}$. Here, tangent on a curve $f(x)$ is approximately given by

$$\frac{\delta y}{\delta x} = \frac{f(x + \delta x) - f(x)}{\delta x}. \tag{1.60}$$

The difference quotient computes the incline of a curve at x by linearizing the function at x: It is the incline of the secant through $f(x)$ and $f(x + \delta_x)$ and the average incline of f between x and $x + \delta x$.
In the limit for $\delta x \to 0$, we obtain the derivative of f at x, if f is differentiable.

Definition 6 (Derivative)
More formally, the **derivative** *of f at x is defined as*

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{\delta x \to 0} \frac{f(x + \delta x) - f(x)}{\delta x}, \tag{1.61}$$

and the secant in Fig. 1.15 becomes a tangent.

Example
We want to compute the derivative of $f(x) = x^n, n \in \mathbb{N}$. By using (1.61), we obtain

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}x} &= \lim_{\delta x \to 0} \frac{f(x + \delta x) - f(x)}{\delta x} && (1.62)\\
&= \lim_{\delta x \to 0} \frac{(x + \delta x)^n - x^n}{\delta x} && (1.63)\\
&= \lim_{\delta x \to 0} \frac{\sum_{i=0}^{n} \binom{n}{i} x^{n-i} \delta x^i - x^n}{\delta x} && (1.64)\\
&= \lim_{\delta x \to 0} \frac{\sum_{i=1}^{n} \binom{n}{i} x^{n-i} \delta x^i}{\delta x} && (1.65)
\end{aligned}$$

$$= \lim_{\delta x \to 0} \sum_{i=1}^{n} \binom{n}{i} x^{n-i} \delta x^{i-1} \tag{1.66}$$

$$= \lim_{\delta x \to 0} \binom{n}{1} x^{n-1} + \underbrace{\sum_{i=2}^{n} \binom{n}{i} x^{n-i} \delta x^{i-1}}_{\to 0 \text{ as } \delta x \to 0} \tag{1.67}$$

$$= \frac{n!}{1!(n-1)!} x^{n-1} = n x^{n-1} \tag{1.68}$$

1.4.1 Partial Differentiation and Gradients

Ordinary differentiation $\frac{\mathrm{d}f}{\mathrm{d}x}$ applies to functions f of a scalar variable $x \in \mathbb{R}$. In the following, we consider the case where the function f depends on one or more variables $\boldsymbol{x} \in \mathbb{R}^n$, e.g., $f(\boldsymbol{x}) = f(x_1, x_2)$. The generalization of the derivative to functions of several variables is the **gradient**.
We find the gradient of the function f with respect to $\boldsymbol{x}$ by *varying one variable at a time* and keeping the others constant. The gradient is then the vector of these **partial derivatives**.

Definition 7 (Partial Derivative)
For a function $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$ of n variables $x_1, \ldots, x_n$ we define the partial derivatives as

$$\frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_n) - f(\boldsymbol{x})}{h} \tag{1.69}$$

$$\vdots$$

$$\frac{\partial f}{\partial x_n} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{n-1}, x_n + h) - f(\boldsymbol{x})}{h} \tag{1.70}$$

and collect them in the row vector

$$\nabla_{\boldsymbol{x}} f = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{1 \times n}. \tag{1.71}$$

Here, we used the compact vector notation $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$.

Remark 5
The definition of the gradient as the limit of the difference quotient can be exploited when checking gradients in computer programs: When we compute gradients and implement them, we often 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 Definition 7 with our (analytic) implementation of the gradient.

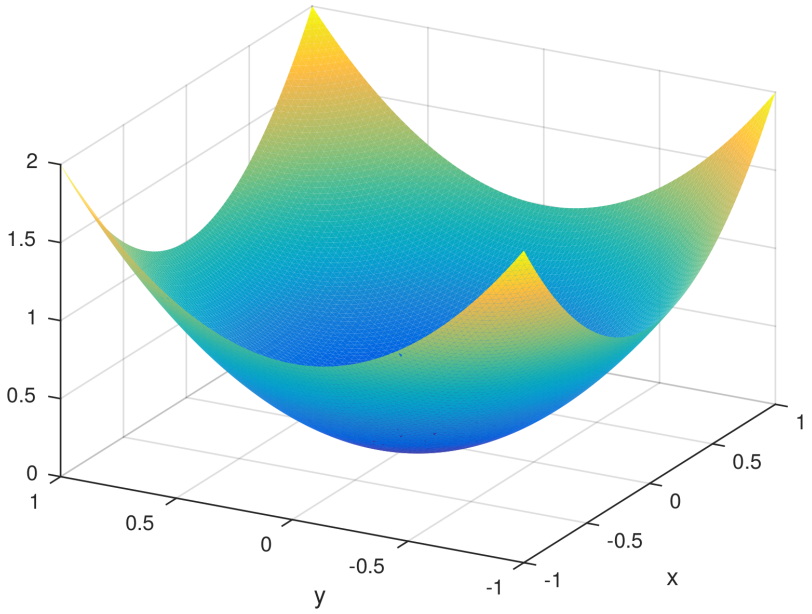


Figure 1.16: $f(x,y) = x^2 + y^2$

Example

For $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$, the **partial derivatives** (i.e., the derivatives of f with respect to x_1 and x_2) are

$$\frac{\partial f(x_1,x_2)}{\partial x_1} = 2x_1x_2 + x_2^3 \tag{1.72}$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} = x_1^2 + 3x_1x_2^2 \tag{1.73}$$

where ∂ indicates that it is a partial derivative, and the gradient is then

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}} = \left[\frac{\partial f(x_1,x_2)}{\partial x_1} \quad \frac{\partial f(x_1,x_2)}{\partial x_2}\right] = \left[2x_1x_2 + x_2^3 \quad x_1^2 + 3x_1x_2^2\right] \in \mathbb{R}^{1\times 2}. \tag{1.74}$$

Example

Consider the function $f(x,y) = x^2+y^2$ (see Fig. 1.16). We obtain the partial derivative $\partial f/\partial x$ by treating y as a constant and computing the derivative of f with respect to x. We then obtain

$$\frac{\partial f(x,y)}{\partial x} = 2x. \tag{1.75}$$

Similarly, we obtain the partial derivative of f with respect to y as

$$\frac{\partial f(x,y)}{\partial y} = 2y. \tag{1.76}$$

Example
For $f(x,y) = (x+2y^3)^2$, we get

$$\frac{\partial f(x,y)}{\partial x} = 2(x+2y^3)\frac{\partial}{\partial x}(x+2y^3) = 2(x+2y^3). \tag{1.77}$$

1.4.1.1 Jacobian

The matrix (or vector) of all first-order partial derivatives of a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is called the **Jacobian**. The Jacobian $\boldsymbol{J}$ is an $m \times n$ matrix, which is usually defined and arranged as follows:

$$\boldsymbol{J} = \nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \tag{1.78}$$

$$J(i,j) = \frac{\partial f_i}{\partial x_j}. \tag{1.79}$$

In particular, a function $f : \mathbb{R}^n \to \mathbb{R}$, which maps a vector $\boldsymbol{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ onto a scalar $y \in \mathbb{R}$ (e.g., $y = \sum_i x_i$), possesses a Jacobian that is a row vector (matrix of dimension $1 \times n$), see (1.71).

Remark 6
We sometimes see gradients defined as column vectors (and not as row vectors as we do here) and Jacobians defined as the transpose of the definition we use. The disadvantage with these definitions is that we need to pay attention to transposes when we perform matrix multiplications, which appear when we apply the chain rule in the multivariate case. With the standard definition we use here, this extra attention is not required.

Remark 7 (Gradients of Matrices)
Matrices represent (linear) mappings. We will encounter situations where we need to take gradients of matrices with respect to vectors (or other matrices), which results in a multi-dimensional tensor. For example, if we compute the gradient of an $m \times n$ matrix with respect to a $p \times q$ matrix, the resulting Jacobian would be $(p \times q) \times (m \times n)$, i.e., a four-dimensional tensor (or array). However, we can exploit the fact that there is an isomorphism between the space $\mathbb{R}^{m \times n}$ of $m \times n$ matrices and the space $\mathbb{R}^{mn}$ of mn vectors. Therefore, we can re-shape[16] our matrices into vectors of lengths mn and pq, respectively, which results in a Jacobian of size $pq \times mn$.

[16] e.g., by stacking the columns of the matrix

Notation Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x, y. Multiple partial derivatives (as for ordinary derivatives) are expressed as

- $\frac{\partial^2 f}{\partial x^2}$ is the second partial derivative of f with respect to x
- $\frac{\partial^n f}{\partial x^n}$ is the nth partial derivative of f with respect to x
- $\frac{\partial^2 f}{\partial x \partial y}$ is the partial derivative obtained by first partial differentiating by y and then x
- $\frac{\partial^2 f}{\partial y \partial x}$ is the partial derivative obtained by first partial differentiating by x and then y

If $f(x, y)$ is a twice (continuously) differentiable function then

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}, \tag{1.80}$$

i.e., the order of differentiation does not matter and the corresponding **Hessian matrix** (the matrix of second partial derivatives) is symmetric.[17]

1.4.1.2 Linearization and Taylor Series

The gradient ∇f of a function f is often used for a locally linear approximation of f around $\boldsymbol{x}_0$:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0). \tag{1.81}$$

Here $(\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)$ is the gradient of f with respect to $\boldsymbol{x}$, evaluated at $\boldsymbol{x}_0$. Note that (1.81) is equivalent to the first two terms in the **multi-variate Taylor-series expansion** of f at $\boldsymbol{x}_0$.

Definition 8 (Taylor Series)
For a function $f : \mathbb{R} \to \mathbb{R}$, the **Taylor series** *of a smooth*[18] *function $f \in \mathcal{C}^\infty$ at x_0 is defined as*

$$f(x_0) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \tag{1.82}$$

where $f^{(k)}(x_0)$ is the kth derivative of f at x_0.[19]
For the **multivariate Taylor series**, *we consider a function*

$$f : \mathbb{R}^D \to \mathbb{R} \tag{1.83}$$

$$\boldsymbol{x} \mapsto f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^D, \tag{1.84}$$

[17] The Hessian measures the local geometry of curvature.
[18] i.e., $f \in \mathcal{C}^\infty$ (infinitely often continuously differentiable)
[19] If $x_0 = 0$, we obtain the **Maclaurin series**.

that is smooth at $\boldsymbol{x}_0$. The Taylor series of f at $(\boldsymbol{x}_0)$ is defined as

$$f(\boldsymbol{x}) = \sum_{k=0}^{\infty} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} (\boldsymbol{x} - \boldsymbol{x}_0)^k , \tag{1.85}$$

where D^k is the k-th (total) derivative of f with respect to $\boldsymbol{x}$.
The **Taylor polynomial of degree** n *of f at $\boldsymbol{x}_0$ contains the first $n+1$ components of the series in* (1.85) *and is defined as*

$$T_n = \sum_{k=0}^{n} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} (\boldsymbol{x} - \boldsymbol{x}_0)^k . \tag{1.86}$$

Example
Consider the function

$$f(x, y) = x^2 + 2xy + y^3 . \tag{1.87}$$

Compute the Taylor series at $(x_0, y_0) = (1, 2)$.

$$f(1,2) = 13 \tag{1.88}$$
$$\frac{\partial f}{\partial x} = 2x + 2y \Rightarrow \frac{\partial f}{\partial x}(1,2) = 6 \tag{1.89}$$
$$\frac{\partial f}{\partial y} = 2x + 3y^2 \Rightarrow \frac{\partial f}{\partial y}(1,2) = 14 \tag{1.90}$$
$$\frac{\partial^2 f}{\partial x^2} = 2 \Rightarrow \frac{\partial^2 f}{\partial x^2}(1,2) = 2 \tag{1.91}$$
$$\frac{\partial^2 f}{\partial y^2} = 6y \Rightarrow \frac{\partial^2 f}{\partial y^2}(1,2) = 12 \tag{1.92}$$
$$\frac{\partial f^2}{\partial x \partial y} = 2 \Rightarrow \frac{\partial f^2}{\partial x \partial y}(1,2) = 2 \tag{1.93}$$
$$\frac{\partial f^2}{\partial y \partial x} = 2 \Rightarrow \frac{\partial f^2}{\partial y \partial x}(1,2) = 2 \tag{1.94}$$
$$\frac{\partial^3 f}{\partial x^3} = 0 \Rightarrow \frac{\partial^3 f}{\partial x^3}(1,2) = 0 \tag{1.95}$$
$$\frac{\partial^3 f}{\partial y^3} = 4 \Rightarrow \frac{\partial^3 f}{\partial y^3}(1,2) = 6 \tag{1.96}$$

Higher-order derivatives and the mixed derivatives of degree 3 (e.g., $\frac{\partial f^3}{\partial x^2 \partial y}$) vanish. Therefore, the Taylor series of f at $(1,2)$ is

$$f(x) = f(1,2) \tag{1.97}$$

$$+\frac{\partial f(1,2)}{\partial x}(x-1)+\frac{\partial f(1,2)}{\partial y}(y-2) \tag{1.98}$$

$$+\frac{1}{2!}\left(\frac{\partial^2 f(1,2)}{\partial x^2}(x-1)^2+\frac{\partial^2 f(1,2)}{\partial y^2}(y-2)^2+2\frac{\partial^2 f(1,2)}{\partial x \partial y}(x-1)(y-2)\right) \tag{1.99}$$

$$+\frac{1}{3!}\frac{\partial^3 f(1,2)}{\partial y^3}(y-2)^3 \tag{1.100}$$

$$=13+6(x-1)+14(y-2)+(x-1)^2+6(y-2)^2+2(x-1)(x-2)+(y-3)^3 \tag{1.101}$$

Remark 8 (Application)
In machine learning (and other disciplines), we often need to compute expectations, i.e., we need to solve integrals of the form

$$\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})]=\int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}. \tag{1.102}$$

Even if $p(\boldsymbol{x})$ is in a convenient form (e.g., Gaussian), this integral cannot be solved in general. The Taylor series expansion is one way of finding an approximate solution: Assuming $p(\boldsymbol{x})=\mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$ is Gaussian, then the first-order Taylor series expansion around $\boldsymbol{\mu}$ locally linearizes the nonlinear function f. For linear functions, we can compute the mean (and the covariance) exactly if $p(\boldsymbol{x})$ is Gaussian distributed (see Section 1.2.3.5). This property is heavily exploited by the Extended Kalman Filter (Maybeck, 1979) for online state estimation in nonlinear dynamical systems[20].

1.4.2 Useful Identities for Computing Gradients

In the following, we list some useful gradients that are frequently required in a machine learning context (Petersen and Pedersen, 2012):

$$\frac{\partial f(\boldsymbol{X})^\top}{\partial \boldsymbol{X}}=\left(\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)^\top \tag{1.103}$$

$$\frac{\partial \mathrm{tr}(f(\boldsymbol{X}))}{\partial \boldsymbol{X}}=\mathrm{tr}\left(\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{1.104}$$

$$\frac{\partial \det(f(\boldsymbol{X}))}{\partial \boldsymbol{X}}=\det(\boldsymbol{X})\mathrm{tr}\left(\boldsymbol{X}^{-1}\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{1.105}$$

$$\frac{\partial \boldsymbol{Y}^{-1}}{\partial \boldsymbol{X}}=-\boldsymbol{Y}^{-1}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}}\boldsymbol{Y}^{-1} \tag{1.106}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}^{-1}\boldsymbol{b}}{\partial \boldsymbol{X}}=-(\boldsymbol{X}^{-1})^\top \boldsymbol{a}\boldsymbol{b}^\top (\boldsymbol{X}^{-1})^\top \tag{1.107}$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{a}}{\partial \boldsymbol{x}}=\boldsymbol{a}^\top \tag{1.108}$$

[20] also called "state-space models"

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{1.109}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X} \boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^\top \tag{1.110}$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^\top (\boldsymbol{B} + \boldsymbol{B}^\top) \tag{1.111}$$

$$\frac{\partial}{\partial \boldsymbol{s}} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = -2(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}\boldsymbol{A} \quad \text{for symmetric } \boldsymbol{W} \tag{1.112}$$

1.4.3 Chain Rule

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x_1, x_2. Furthermore, x_1 and x_2 are themselves functions of t. To compute the gradient of f with respect to t, we need to apply the **chain rule** for multivariate functions as

$$\frac{\mathrm{d} f(x_1(t), x_2(t))}{\mathrm{d}t} = \frac{\partial f(x_1, x_2)}{\partial x_1} \frac{\mathrm{d}x_1}{\mathrm{d}t} + \frac{\partial f(x_1, x_2)}{\partial x_2} \frac{\mathrm{d}x_2}{\mathrm{d}t} \tag{1.113}$$

where d denotes the **total derivative**.

Example
Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin t$ and $x_2 = \cos t$ then

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial x_1} \frac{\mathrm{d}x_1}{\mathrm{d}t} + \frac{\partial f}{\partial x_c} \frac{\mathrm{d}x_2}{\mathrm{d}t} \tag{1.114}$$

$$= 2 \sin t \frac{\mathrm{d} \sin t}{\mathrm{d}t} + 2 \frac{\mathrm{d} \cos t}{\mathrm{d}t} \tag{1.115}$$

$$= 2 \sin t \cos t - 2 \sin t = 2 \sin t (\cos t - 1) \tag{1.116}$$

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 s and t, the chain rule yields

$$\frac{\mathrm{d}f}{\mathrm{d}s} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial s}, \tag{1.117}$$

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t}, \tag{1.118}$$

which can be expressed as a matrix equation

$$\nabla f := \frac{\mathrm{d}f}{\mathrm{d}(s,t)} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}(s,t)} = \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix}}_{= \frac{\partial f}{\partial \boldsymbol{x}}} \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix}}_{= \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}(s,t)}} \tag{1.119}$$

Example
Consider the function $h : \mathbb{R} \to \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$f : \mathbb{R}^2 \to \mathbb{R} \tag{1.120}$$
$$g : \mathbb{R} \to \mathbb{R}^2 \tag{1.121}$$
$$f(\boldsymbol{x}) = \exp(x_1 x_2^2), \tag{1.122}$$
$$\boldsymbol{x} = g(t) = \begin{bmatrix} t\cos t \\ t\sin t \end{bmatrix} \tag{1.123}$$

and compute the gradient of h with respect to t.
Since $f : \mathbb{R}^2 \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}^2$ we note that

$$\frac{\partial f}{\partial \boldsymbol{x}} \in \mathbb{R}^{1\times 2}, \tag{1.124}$$
$$\frac{\partial g}{\partial t} \in \mathbb{R}^{2\times 1}. \tag{1.125}$$

The desired gradient is computed by applying the chain-rule:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \tag{1.126}$$
$$= \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}x_1}{\mathrm{d}t} \\ \frac{\mathrm{d}x_2}{\mathrm{d}t} \end{bmatrix} \tag{1.127}$$
$$= \begin{bmatrix} \exp(x_1 x_2^2) x_2^2 & 2\exp(x_1 x_2^2) x_1 x_2 \end{bmatrix} \begin{bmatrix} \cos t - t\sin t \\ \sin t + t\cos t \end{bmatrix} \tag{1.128}$$
$$= \exp(x_1 x_2^2)\Big(x_2^2(\cos t - t\sin t) + 2x_1 x_2(\sin t + t\cos t)\Big) \tag{1.129}$$

Remark 9
The chain rule is useful when we perform change of variables, e.g., when we look at polar coordinates (complex numbers).

Remark 10 (Application in Machine Learning)
In machine learning, the chain rule plays an important role when optimizing parameters of a hierarchical model (e.g., for maximum likelihood estimation). An area where the chain rule is used to an extreme is Deep Learning where the function value y is computed as a nested/layered function

$$y = f_K(f_{K-1}(\cdots(f_1(\boldsymbol{x}))\cdots)), \tag{1.130}$$

where $\boldsymbol{x}$ are the inputs (e.g., images), y are the observations (e.g., class labels) and every function f_i, $i = 1,\ldots,K$ possesses its own parameters. In neural networks with multiple layers, we have functions $f_i(\boldsymbol{x}) = \sigma(\boldsymbol{A}_i\boldsymbol{x}_{i-1} + \boldsymbol{b}_i)$ in the ith layer, where $\boldsymbol{x}_{i-1}$

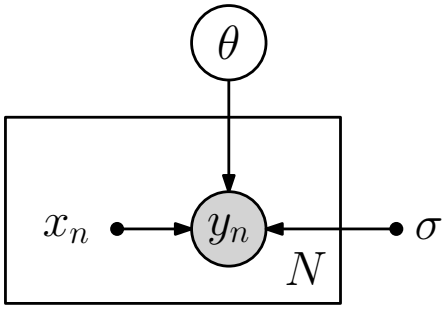


Figure 1.17: Probabilistic graphical model for linear regression.

is the output of layer $i-1$ and σ an activation function, e.g., the logistic sigmoid $\frac{1}{1+e^{-x}}$, tanh *or a rectified linear unit (ReLU). In order to train these models, we have to compute the gradient of a loss function with respect to the inputs of each layer (e.g., $\boldsymbol{x}_{i-1}$) to obtain the partial derivative with respect to the parameters of the previous layer (e.g., $\boldsymbol{A}_{i-1}, \boldsymbol{b}_{i-1}$). There are efficient ways of implementing this repeated application of the chain-rule using* **backpropagation** *(Kelley, 1960; Bryson, 1961; Dreyfus, 1962; Rumelhart et al., 1986).*

1.5 Parameter Estimation

Assume we are given a **training set** $\mathcal{D}$ consisting of N inputs $\boldsymbol{x}_i$ and corresponding observations y_i, $i = 1,\dots,N$, where y_i and y_j are conditionally independent given $\boldsymbol{x}_i, \boldsymbol{x}_j$. Our objective is to find optimal parameters $\boldsymbol{\theta}^*$ for the linear regression model (1.2). The corresponding graphical model is given in Fig. 1.17.

1.5.1 Maximum Likelihood Estimation

A widely used approach to finding the desired parameters $\boldsymbol{\theta}^*$ is maximum likelihood estimation where

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}), \qquad \boldsymbol{X} := [\boldsymbol{x}_1|\cdots|\boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times D}, \quad \boldsymbol{y} := [y_1,\dots,y_N]^\top \in \mathbb{R}^N \tag{1.131}$$

maximizes the likelihood function $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$.

Remark 11
Note that the likelihood is not a probability distribution in $\boldsymbol{\theta}$: It is simply a function but does usually not integrate to 1 (i.e., it is unnormalized), and may not even be integrable with respect to $\boldsymbol{\theta}$. However, the likelihood in (1.131) *is a probability distribution in $\boldsymbol{y}$.*

To find the desired parameters $\boldsymbol{\theta}^*$ that maximize the likelihood, we typically do gradient ascent (or gradient descent on the negative likelihood). For numerical reasons, we apply the log-transformation to the problem[21] and minimize the negative

[21] Note that the logarithm is a (strictly) monotonically increasing function.

log-likelihood

$$\theta^* \in \arg\min_{\theta} \Big(-\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) \Big) = \arg\min_{\theta} \left(-\log \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \right) \tag{1.132}$$

$$= \arg\min_{\theta} \sum_{i=1}^{N} -\log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}), \tag{1.133}$$

where we exploited that the likelihood factorizes over the number of data points due to our independence assumption on the training set.
In the linear regression model (1.2) the likelihood is Gaussian (due to the Gaussian additive noise term), such that we arrive at

$$-\log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\theta})^2 + \text{const} \tag{1.134}$$

where the constant includes all terms independent of $\boldsymbol{\theta}$. Using (1.134) in the negative log-likelihood (1.133) we obtain (ignoring the constant terms)

$$\mathcal{L}(\boldsymbol{\theta}) := -\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\theta})^2 \tag{1.135}$$

$$= \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) = \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2, \tag{1.136}$$

$$\boldsymbol{X} := \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times D}, \tag{1.137}$$

where $\boldsymbol{X}$ is called the **design matrix**.[22] In (1.136) we replaced the sum of squared with the squared norm[23] of the difference term $\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}$.

Remark 12
In machine learning, the negative log likelihood function is also called an **error function**.

Now that we have a concrete form of the negative log-likelihood function we need to optimize. We will discuss two approaches, both of which are based on gradients.

1.5.1.1 Closed-Form Solution

We immediately see that (1.136) is quadratic in $\boldsymbol{\theta}$. This means that we can find a unique global solution $\boldsymbol{\theta}^*$ for minimizing the negative log-likelihood. We can find the global optimum by computing the gradient of $\mathcal{L}$, setting it to $\boldsymbol{0}$ and solving for $\boldsymbol{\theta}$.[24]

[22] Note that there is some notation overloading: We summarize the set of training inputs in $\boldsymbol{X}$, whereas in the design matrix we additionally assume a specific "shape".

[23] Remember that $\|\boldsymbol{x}\|^2 := \langle \boldsymbol{x}, \boldsymbol{x} \rangle$ where $\langle \cdot, \cdot \rangle$ is a scalar product (inner product).

[24] In this case, setting the gradient to $\boldsymbol{0}$ is a necessary and sufficient condition. As an exercise, compute the Hessian (matrix of second derivatives) and show that it is positive definite.

We compute the gradient of $\mathcal{L}$ with respect to the parameters as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}}\left(\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\theta})\right) = \frac{1}{2\sigma^2}\frac{\partial}{\partial \boldsymbol{\theta}}\left(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}\right) \tag{1.138}$$

$$= \frac{1}{\sigma^2}(-\boldsymbol{y}^\top\boldsymbol{X} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}) \in \mathbb{R}^{1\times D}. \tag{1.139}$$

A necessary condition for $\boldsymbol{\theta}$ being optimal we set this gradient to $\mathbf{0}$ and obtain

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \mathbf{0} \overset{(1.139)}{\Leftrightarrow} (\boldsymbol{\theta}^*)^\top\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{y}^\top\boldsymbol{X} \tag{1.140}$$

$$\Leftrightarrow (\boldsymbol{\theta}^*)^\top = \boldsymbol{y}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1} \tag{1.141}$$

$$\Leftrightarrow \boldsymbol{\theta}^* = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \tag{1.142}$$

We could right-multiply the first equation by $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ because $\boldsymbol{X}^\top\boldsymbol{X}$ is positive definite (if we do not have two identical inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$ for $i \neq j$). Note that we actually do obtain a global minimum since the Hessian $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{D\times D}$ is positive definite.

1.5.1.2 Iterative Solution

We may be interested in finding parameters on which the log-likelihood depends in a more complicated way. Then, a closed-form solution is often not available. However, we can use an iterative procedure to find a local optimum. The most straightforward procedure for this is gradient descent, which we will discuss in more detail in Section 1.6.

1.5.1.3 Maximum Likelihood Estimation with Features

When we consider the linear regression model

$$y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon \tag{1.143}$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^K$ is a (nonlinear) transformation of the inputs $\boldsymbol{x}$.

Example (Polynomial Regression)
We are concerned with a regression problems $y = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta} + \epsilon$, where $x \in \mathbb{R}$. A transformation that is often used in this context is

$$\boldsymbol{\phi}(x) = \begin{bmatrix}\phi_0(x)\\ \phi_1(x)\\ \vdots \\ \phi_K(x)\end{bmatrix} = \begin{bmatrix}1\\ x\\ x^2\\ x^3\\ \vdots\\ x^K\end{bmatrix} \in \mathbb{R}^{K+1}. \tag{1.144}$$

This means, we "lift" the original 1D input space into a $K+1$-dimensional feature space consisting of monomials. With these features, we can model polynomials of degree $\leq K$ within the framework of linear regression: A polynomial of degree K is given by

$$f(x) = \sum_{i=0}^{K} \theta_i x^i = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta} \tag{1.145}$$

where $\boldsymbol{\phi}$ is defined in (1.144) and $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_K]^\top$ contains the parameters θ_i.

When we consider the training data $\boldsymbol{x}_i, y_i$, $i = 1, \ldots, N$ and define the feature (design) matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_0^\top(\boldsymbol{x}_1) & \phi_1^\top(\boldsymbol{x}_1) & \cdots & \phi_K^\top(\boldsymbol{x}_1) \\ \phi_0^\top(\boldsymbol{x}_2) & \phi_1^\top(\boldsymbol{x}_2) & \cdots & \phi_K^\top(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0^\top(\boldsymbol{x}_N) & \cdots & \cdots & \phi_K^\top(\boldsymbol{x}_N) \end{bmatrix}, \quad \Phi_{ij} = \phi_j^\top(\boldsymbol{x}_i) \tag{1.146}$$

the negative log-likelihood can be written as

$$-\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \text{const}. \tag{1.147}$$

Comparing (1.147) with (1.136) we immediately see that $\boldsymbol{X}$ is replaced by $\boldsymbol{\Phi}$. Since both $\boldsymbol{X}$ and $\boldsymbol{\Phi}$ are independent of the parameters $\boldsymbol{\theta}$ that we wish to optimize, we therefore also arrive immediately at the maximum likelihood estimate

$$\boldsymbol{\theta}^* = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} \tag{1.148}$$

for the linear regression problem with nonlinear features defined in (1.143).

Example (Feature Matrix for Polynomials)
For a second-order polynomial and N training points $x_i \in \mathbb{R}, i = 1, \ldots, N$, the feature matrix is

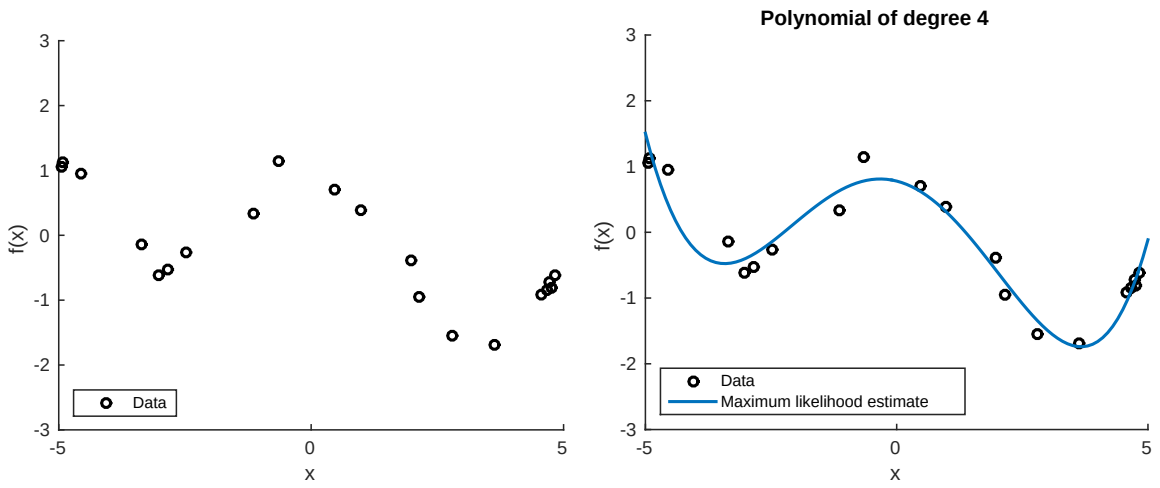
$$\boldsymbol{\Phi} = \begin{bmatrix} 0 & x_1 & x_1^2 \\ 0 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 0 & x_N & x_N^2 \end{bmatrix}. \tag{1.149}$$

1.5.1.4 Properties

The maximum likelihood estimate θ^* possesses the following properties:

- Asymptotic consistency: The MLE converges to the true value in the limit of infinitely many observations, plus a random error that is approximately normal.
- The size of the samples necessary to achieve these properties can be quite large.
- The error's variance decays in $1/N$ where N is the number of data points.
- Especially, in the "small" data regime, maximum likelihood estimation can lead to **overfitting**.

Example (Maximum Likelihood Polynomial Fit)



(a) Regression data set.

(b) Polynomial of degree 4 determined by maximum likelihood estimation.

Figure 1.18: Polynomial regression. (a) Data set consisting of (x_i, y_i) pairs, $i = 1, \ldots, 20$. (b) Maximum likelihood polynomial of degree 4.

Let us consider the data set in Fig. 1.18(a). The data set consists of $N = 20$ pairs (x_i, y_i), where $x_i \sim \mathcal{U}[-5, 5]$ and $y_i = -\sin(x_i/5) + \cos(x_i) + \epsilon$, where $\epsilon \sim \mathcal{N}\left(0, 0.2^2\right)$. We fit a polynomial of degree $M = 4$ using maximum likelihood estimation (i.e., the parameters are given in (1.148)). The result is shown in Fig. 1.18(b).

1.5.2 Overfitting

We have seen that we can use maximum likelihood estimation to fit linear models (e.g., polynomials) to data. We can evaluate the quality of the model by comput-

ing the error/loss incurred. One way of doing this is to compute the negative log-likelihood (1.133), which we minimized to determine the MLE. Alternatively, given that the noise parameter σ^2 is not a free parameter, we can ignore the scaling by $1/\sigma^2$, so that we end up with a squared-error-loss function $\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2$. Instead of using this squared loss, we often use the **root mean squared error (RMSE)**

$$\sqrt{\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2/N} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2}, \tag{1.150}$$

which (a) allows us to compare errors of data sets with different sizes[25] and (b) has the same scale and the same units as the observed function values y_i.[26] Note that the division by σ^2 makes the log-likelihood "unit-free".

We can use the RMSE (or the log-likelihood) to determine the best degree of the polynomial by finding the value M, such that the error is minimized. Given that the polynomial degree is a natural number, we can perform a brute-force search and enumerate all (reasonable) values of M.[27]

Fig. 1.19 shows a number of polynomial fits determined by maximum likelihood. 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 $M = 4, \ldots, 9$ 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 $M = N - 1 = 19$, the function passes 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.[28] The property of the polynomials fitting the noise structure is called **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 100 data points generated using exactly the same procedure used to generate the training set, but with new choices for the random noise values included in the target values. As test inputs, we chose a linear grid of 100 points in the interval of $[-5, 5]$. For each choice of M, we evaluate the RMSE (1.150) for both the training data and the test data.

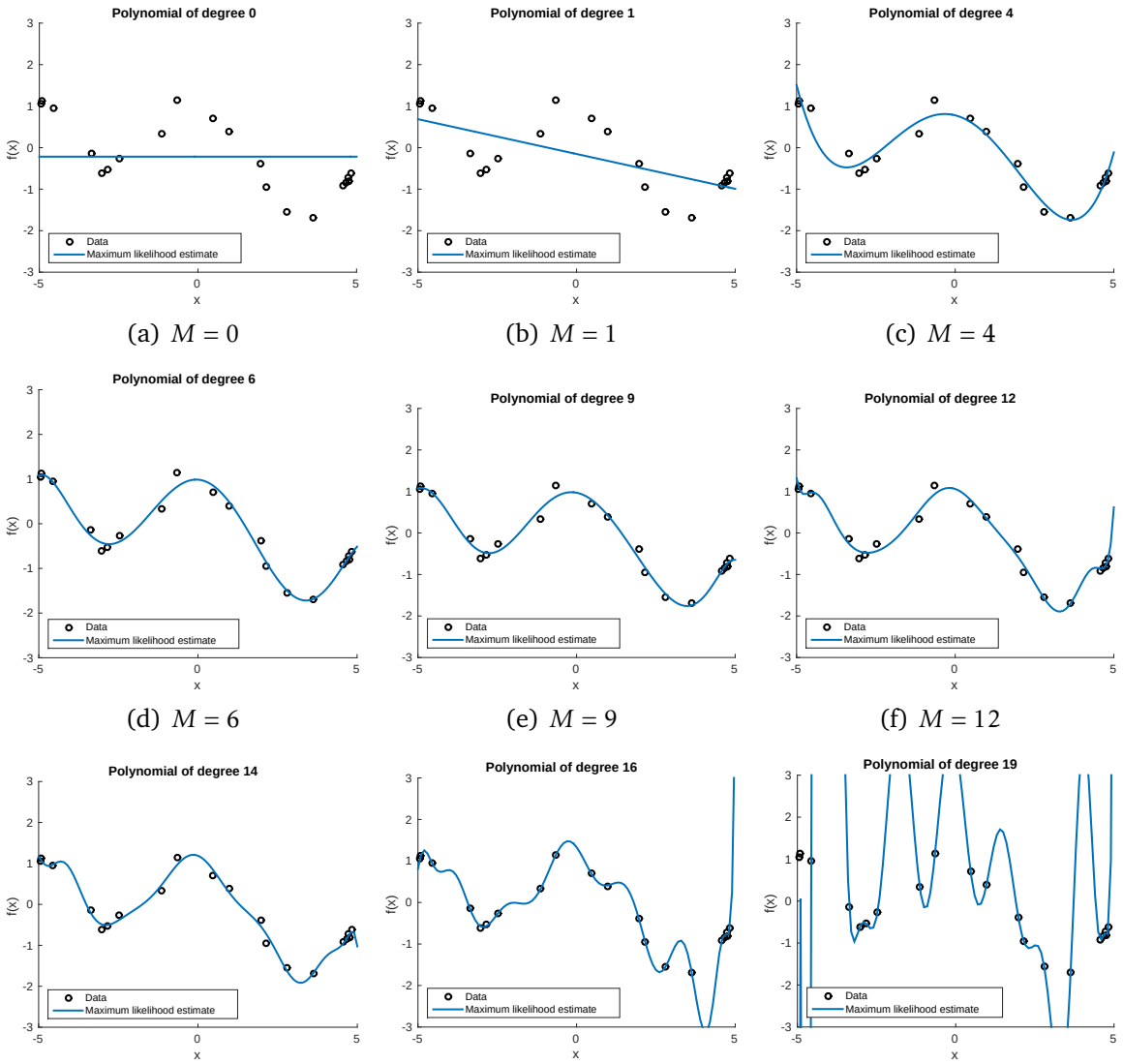
Looking now at the test error, which is a qualitive measure of the generalization properties of the corresponding polynomial, we notice that initially the test error decreases, see Fig. 1.20 (red). For fourth-order polynomials the test error is relatively low and stays relatively constant up to degree 11. However, from degree 12 onward the test error increases significantly, and high-order polynomials have very bad generalization properties. In this particular example, this also is evident from

[25] The RMSE is normalized.

[26] Assume, we fit a model that maps post-codes ($\boldsymbol{x}$ is given in latitude,longitude) to house prices (y-values are GBP). Then, the RMSE is also measured in GBP, whereas the squared error is given in GBP^2.

[27] For a training set of size N it is sufficient to test $0 \leq M \leq N - 1$.

[28] Note that the noise variance $\sigma^2 > 0$.



(a) $M = 0$ (b) $M = 1$ (c) $M = 4$

(d) $M = 6$ (e) $M = 9$ (f) $M = 12$

(g) $M = 14$ (h) $M = 16$ (i) $M = 19$

Figure 1.19: Polynomial fits for different degrees M

the corresponding maximum likelihood fits in Fig. 1.19. Note that the training error (blue curve in Fig. 1.20) never increases as a function of M. In our example, the best generalization (the point of the smallest test error) is achieved for a polynomial of degree $M = 6$.

It is a bit counter-intuitive that a polynomial of degree $M = 19$ is a worse approximation than a polynomial of degree $M = 4$, which is a special case of a 19th-order polynomial (by setting all higher coefficients to 0). However, a 19th-order polynomial can also describe many more functions, i.e., it is a much more flexible model. In the data set we considered, the observations y_n were noisy (i.i.d. Gaussian). A polynomial of a high degree will use its flexibility to model random disturbances as systematic/structural properties of the underlying function. Overfitting can be seen as a general problem of maximum likelihood estimation (Bishop, 2006). Assuming

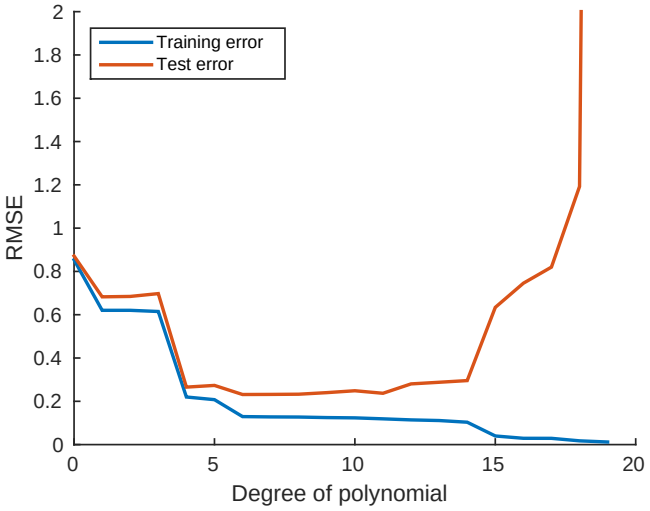


Figure 1.20: Training and test error.

we had noise-free data, overfitting does not occur, which is also revealed by the test error, see Fig. 1.21.

1.5.3 Regularization

In Section 1.5.1, we saw that maximum likelihood estimation is prone to overfitting. It often happens that the parameter values become relatively big if we run into overfitting (Bishop, 2006). One way to control overfitting is to penalize big parameter values. A technique that is often used to control overfitting is **regularization**. In regularization, we add a term to the log-likelihood that penalizes the amplitude of the parameters $\boldsymbol{\theta}$. A typical example is a "loss function" of the form

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{1.151}$$

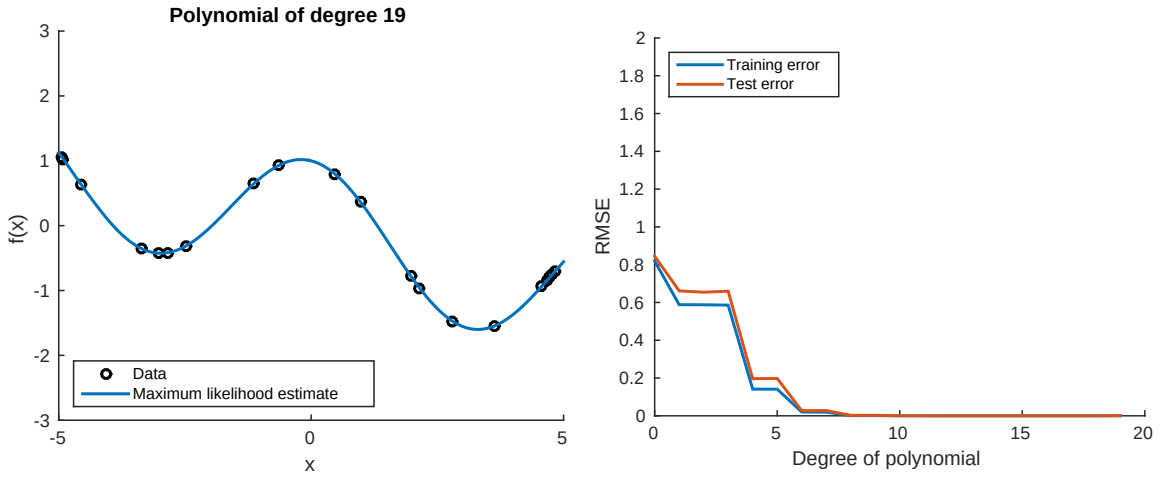
where the second term is the regularizer, and $\lambda \geq 0$ controls the "strictness" of the regularization.[29]

1.5.4 Maximum-A-Posterior (MAP) Estimation

From a probabilistic perspective, adding a regularizer is identical to using a prior distribution $p(\boldsymbol{\theta})$ on the parameters and then selecting the parameters that maximize the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})$, i.e., we choose the parameters $\boldsymbol{\theta}$ that are "most probable" given the training data: In a Bayesian setting, the posterior over the parameters $\boldsymbol{\theta}$ given the training data $\boldsymbol{X},\boldsymbol{y}$ is obtained by applying Bayes' theorem as

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}. \tag{1.152}$$

[29] Instead of the 2-norm, we can choose and p-norm $\|\cdot\|_p$. In practice, smaller values for p lead to sparser solutions. Here, "sparse" means that many parameter values $\theta_i = 0$, which is also useful for variable selection. For $p = 1$, the regularizer is called LASSO (least absolute shrinkage and selection operator) and was proposed by Tibshirani (1996).



(a) Maximum likelihood fit. (b) Training and test error

Figure 1.21: Noise-free maximum likelihood estimation with a 19th-degree polynomial. (a) Overfitting no longer occurs; (b) the test error declines to 0.

The parameter vector $\boldsymbol{\theta}_{\text{MAP}}$ that maximizes the posterior (1.152) is called the **maximum 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

$$\log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + \text{const}, \tag{1.153}$$

where the constant includes the terms independent of $\boldsymbol{\theta}$. We see that the log-posterior in (1.153) consists of the log-likelihood $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$ and the log-prior $\log p(\boldsymbol{\theta})$.

Remark 13 (Relation to Regularization)
Choosing a Gaussian parameter prior $p(\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{0}, b^2\boldsymbol{I}\right)$, $b^2 = \frac{1}{2\lambda}$, *the log-prior term will be*

$$\log p(\boldsymbol{\theta}) = \underbrace{\lambda\boldsymbol{\theta}^\top\boldsymbol{\theta}}_{=\lambda\|\boldsymbol{\theta}\|_2^2} + const, \tag{1.154}$$

and we recover exactly the regularization term in (1.151). *This means that for a quadratic regularization, the regularization parameter* λ *in* (1.151) *corresponds to twice the precision (inverse variance) of the Gaussian (isotropic) prior* $p(\boldsymbol{\theta})$. *The log-prior in* (1.153) *plays the role of a regularizer that penalizes implausible values, i.e., values that are unlikely under the prior.*

To find the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$, we minimize the negative log-posterior with respect to $\boldsymbol{\theta}$, i.e., we solve

$$\boldsymbol{\theta}_{\text{MAP}} \in \arg\min_{\boldsymbol{\theta}} -\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}). \tag{1.155}$$

We compute the gradient with respect to $\boldsymbol{\theta}$ as

$$-\frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})}{\partial \boldsymbol{\theta}} = -\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial \log p(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{1.156}$$

where we identify the first term on the right-hand-side as the gradient of the negative log-likelihood given in (1.139).

1.5.4.1 MAP Estimation for Linear Regression

We consider the linear regression problem where

$$y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}\left(0, \sigma^2\right), \tag{1.157}$$

with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{0}, b^2\boldsymbol{I}\right)$ on the parameters $\boldsymbol{\theta}$.
The negative log-posterior for this model is

$$-\log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{1}{2b^2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{const}. \tag{1.158}$$

Here, the blue term corresponds to the contribution from the log-likelihood, and the red term originates from the log-prior.
The gradient of the log-posterior with respect to the parameters $\boldsymbol{\theta}$ is

$$-\frac{\partial \log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})}{\partial \boldsymbol{\theta}} = \frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top. \tag{1.159}$$

We will find the MAP estimate $\boldsymbol{\theta}_{\text{MAP}}$ by setting this gradient to $\mathbf{0}$:

$$\frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top = \mathbf{0} \tag{1.160}$$

$$\Leftrightarrow \boldsymbol{\theta}^\top\left(\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{1}{b^2}\boldsymbol{I}\right) - \frac{1}{\sigma^2}\boldsymbol{y}^\top\boldsymbol{\Phi} = \mathbf{0} \tag{1.161}$$

$$\Leftrightarrow \boldsymbol{\theta}^\top\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right) = \boldsymbol{y}^\top\boldsymbol{\Phi} \tag{1.162}$$

$$\Leftrightarrow \boldsymbol{\theta}^\top = \boldsymbol{y}^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1} \tag{1.163}$$

$$\Leftrightarrow \boldsymbol{\theta}_{\text{MAP}} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y}. \tag{1.164}$$

Comparing the MAP estimate in (1.164) with the maximum likelihood estimate in (1.148) we see that the only difference between both solutions is the additional red term $\frac{\sigma^2}{b^2}\boldsymbol{I}$ in the inverse matrix.[30] This term ensures that the inverse exists and serves as a **regularizer**.

Example (MAP Estimation for Polynomial Regression)

[30] $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is positive semidefinite and the additional term is strictly positive definite, such that all eigenvalues of the matrix to be inverted are positive.

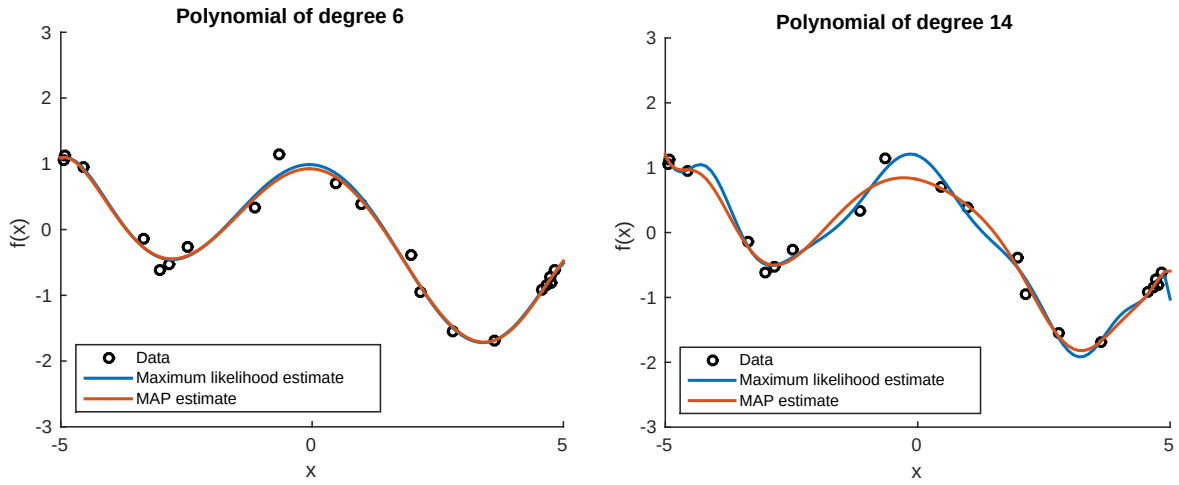


Figure 1.22: Polynomial regression: Maximum likelihood and MAP estimates.

In the polynomial regression example from Section 1.5.1, we place a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}\left(0, 0.1^2\boldsymbol{I}\right)$ on the parameters $\boldsymbol{\theta}$ and determine the MAP estimates according to (1.164). In Fig. 1.22, we show both the maximum likelihood and the MAP estimates for polynomials of degree 6 (left) and degree 14 (right). The prior (regularizer) does not play a significant role for the low-degree polynomial, but keeps the function relatively smooth for higher-degree polynomials. However, the MAP estimate is only able to push the boundaries of overfitting—it is not a general solution to this problem.

1.6 Gradient Descent

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 (or of the approximate gradient) of the function at the current point. Remember that the gradient points in the direction of the steepest ascent and it is orthogonal to the contour lines of the function we wish to optimize.
Gradient descent exploits the fact that if a multivariate function $f(\boldsymbol{x})$ is defined and differentiable in a neighborhood of a point $\boldsymbol{x}_0$ then $f(\boldsymbol{x}_0)$ decreases fastest if one moves from $\boldsymbol{x}_0$ in the direction of the negative gradient $-(\nabla f)(\boldsymbol{x}_0)$ of f at $\boldsymbol{x}_0$. Then, if

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \gamma(\nabla f)(\boldsymbol{x}_0) \tag{1.165}$$

for small **step size** $\gamma \geq 0$ then $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_0)$.
This observation allows us to define a simple gradient-descent algorithm: If we want to find a local optimum $f(\boldsymbol{x}_*)$ of a function $f : \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, we start with an initial guess $\boldsymbol{x}_0$ of the parameters we wish to optimize and then iterate according to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i(\nabla f)(\boldsymbol{x}_i). \tag{1.166}$$

For suitable γ_i, the sequence $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x}_1) \geq \ldots$ converges to a local minimum.

Remark 14
Gradient descent can be relatively slow close to the minimum: Its asymptotic rate of convergence is inferior to many other methods. For poorly conditioned convex problems, gradient descent increasingly 'zigzags' as the gradients point nearly orthogonally to the shortest direction to a minimum point.

1.6.1 Stepsize

Choosing a good stepsize is important in gradient descent: If the stepsize (also called the learning rate) is too small, gradient descent can be slow. If the stepsize is chosen too large, gradient descent can overshoot, fail to converge, or even diverge.
Robust gradient methods must permanently rescale the stepsize empirically depending on local properties of the function. There are two simple heuristics (Toussaint, 2012):

- When the function value increases after a gradient step, the step size was too large. Undo the step and decrease the stepsize.
- When the function value decreases the step could have been larger. Try to increase the stepsize.

Although the "undo" step seems to be a waste of resources, using this heuristic guarantees monotonic convergence.

Example (Solving a Linear System)
When we solve linear equations of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, in practice we solve $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$ approximately by finding $\boldsymbol{x}_*$ that minimizes the the squared error

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{1.167}$$

if we use the Euclidean norm. The gradient of (1.167) with respect to $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} = 2(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top \boldsymbol{A}. \tag{1.168}$$

Remark 15
Gradient descent is rarely used for solving linear equation systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Instead other algorithms, e.g., conjugate gradient descent, are used. The speed of convergence of gradient descent depends on the maximal and minimal eigenvalues of $\boldsymbol{A}$, while the speed of convergence of conjugate gradients has a more complex dependence on the eigenvalues, and can benefit from preconditioning. Gradient descent also benefits from preconditioning, but this is not done as commonly. For further information on gradient descent, pre-conditioning and convergence we refer to CO-477.

1.6.2 Gradient Descent with Momentum

Gradient descent with momentum (Rumelhart et al., 1986) is a method that smoothes out erratic behavior of gradient updates and dampens oscillations.
The idea is to have a gradient update with a memory to implement a moving average. The momentum-based method remembers the update $\Delta \boldsymbol{x}_i$ at each iteration i and determines the next update as a linear combination of the current and previous gradients

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i (\nabla \boldsymbol{f})(\boldsymbol{x}_i) + \alpha \Delta \boldsymbol{x}_i \tag{1.169}$$

$$\Delta \boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} = -\gamma_{i-1} (\nabla \boldsymbol{f})(\boldsymbol{x}_{i-1}), \tag{1.170}$$

where $\alpha \in [0, 1]$. Due to the moving average of the gradient, momentum-based methods are particularly useful when the gradient itself is only a (noisy) estimate. We will discuss stochastic approximations to the gradient in the following.

1.6.3 Stochastic Gradient Descent

Computing the gradient can be very time consuming. However, often it is possible to find a "cheap" approximation of the gradient. Approximating the gradient is still useful as long as it points in roughly the same direction as the true gradient.
Stochastic gradient descent (often shortened in SGD) is a stochastic approximation of the gradient descent method for minimizing an objective function that is written as a sum of differentiable functions.
In machine learning, we often consider objective functions of the form

$$L(\boldsymbol{\theta}) = \sum_k L_k(\boldsymbol{\theta}) \tag{1.171}$$

where $\boldsymbol{\theta}$ is the parameter vector of interest, i.e., we want to find $\boldsymbol{\theta}$ that minimizes L. An example is the negative log-likelihood

$$L(\boldsymbol{\theta}) = -\sum_k \log p(y_k | \boldsymbol{x}_k, \boldsymbol{\theta}) \tag{1.172}$$

in a regression setting, where $\boldsymbol{x}_k \in \mathbb{R}^D$ are the training inputs, y_k are the training targets and $\boldsymbol{\theta}$ are the parameters of the regression model.
Standard gradient descent, as introduced previously, is a "batch" optimization method, i.e., optimization is performed using the full training set by updating the parameters according to

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma_i \nabla L(\boldsymbol{\theta}_i) = \boldsymbol{\theta}_i - \gamma_i \sum_k \nabla L_k(\boldsymbol{\theta}_i) \tag{1.173}$$

for a suitable stepsize parameter γ_i. Evaluating the sum-gradient may require expensive evaluations of the gradients from all summand functions. When the training set is enormous (commonly the case in Deep Learning) and/or no simple formulas exist, evaluating the sums of gradients becomes very expensive: Evaluating the

gradient requires evaluating the gradients of all summands. To economize on the computational cost at every iteration, **stochastic gradient descent** samples a subset of summand functions ("mini-batch") at every step. In an extreme case, the sum in (1.171) is approximated by a single summand (randomly chosen), and the parameters are updated according to

$$\theta_{i+1} = \theta_i - \gamma \nabla L_k(\theta_i). \tag{1.174}$$

In practice, it is good to keep the size of the mini-batch as large as possible to (a) reduce the variance in the parameter update[31] and (b) allow for the computation to take advantage of highly optimized matrix operations that should be used in a well-vectorized computation of the cost and gradient.

1.6.4 Further Reading

For non-differentiable functions, gradient methods are ill-defined. 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).
Stochastic gradient descent is very effective in large-scale machine learning problems, 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).

1.7 Model Selection and Cross Validation

Sometimes, we need to make high-level decisions about the model we want to use in order to increase the performance. Examples include:

- The degree of a polynomial in a regression setting
- The number of components in a mixture model
- The network architecture of a (deep) neural network
- The type of kernel in a support vector machine

The choices we make (e.g., the degree of the polynomial) influence the number of free parameters in the model and thereby also the model complexity. More complex models are more flexible in the sense that they can be used to describe more data sets. For instance, a polynomial of degree 1 (a line $a_0 + a_1 x$) can only be used to describe linear relations between inputs x and observations y. A polynomial of degree 2 can additionally describe quadratic relationships between inputs and observations.[32] Higher-order polynomials are very flexible models as we have seen already in Section 1.5 in the context of polynomial regression.

[31] This often leads to more stable convergence.
[32] A polynomial $a_0 + a_1 x + a_2 x^2$ can also describe linear functions by setting $a_2 = 0$.

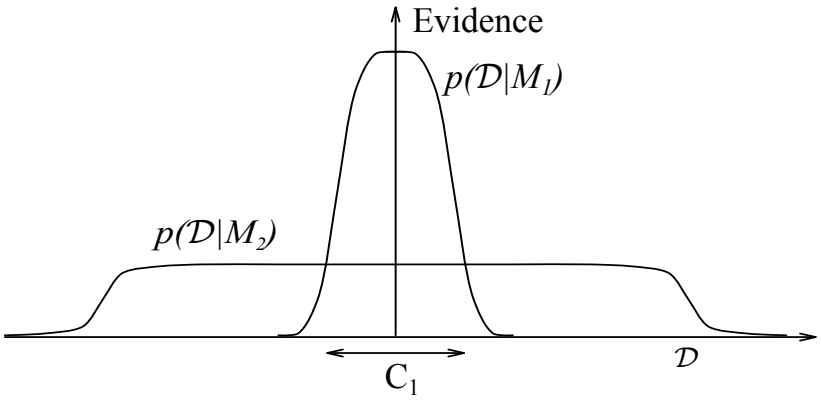


Figure 1.23: "Why Bayesian inference embodies Occam's razor. This figure gives the basic intuition for why complex models can turn out to be less probable. The horizontal axis represents the space of possible data sets $\mathcal{D}$. Bayes' theorem rewards models in proportion to how much they predicted the data that occurred. These predictions are quantified by a normalized probability distribution on $\mathcal{D}$. This probability of the data given model M_i, $p(\mathcal{D}|M_i)$, is called the evidence for M_i. A simple model M_1 makes only a limited range of predictions, shown by $p(\mathcal{D}|M_1)$; a more powerful model M_2 that has, for example, more free parameters than M_1, is able to predict a greater variety of data sets. This means, however, that M_2 does not predict the data sets in region C_1 as strongly as M_1. Suppose that equal prior probabilities have been assigned to the two models. Then, if the data set falls in region C_1 , the less powerful model M_1 will be the more probable model." (MacKay, 2003)

A general problem is that at training time we can only use the training set to evaluate the performance of the model. However, the performance on the training set is not really what we are interested in: In Section 1.5, we have seen that maximum likelihood estimation can lead to overfitting, especially when the training data set is small. Ideally, our model (also) works well on the test set (which is not available at training time). Therefore, we need some mechanisms for assessing how a model **generalizes** to unseen test data. **Model selection** is concerned with exactly this problem.

There are many approaches to model selection, some of which we will discuss later. Generally, they all attempt to trade off model complexity and data fit:[33] The objective is to find the simplest model that explains the data reasonably well.[34] This concept is also known as **Occam's Razor**. One may consider placing a prior on models that favors simpler models. However, it is not necessary to do this: An "automatic Occam's Razor" is quantitatively embodied in the application of Bayesian probability (Spiegelhalter and Smith, 1980; MacKay, 1992; Jefferys and Berger, 1992). Fig. 1.23 from MacKay (2003) illustrates this property.

[33] We assume that simpler models are less prone to overfitting than complex models.

[34] If we treat model selection as a hypothesis testing problem, we are looking for the simplest hypothesis that is consistent with the data (Murphy, 2012).

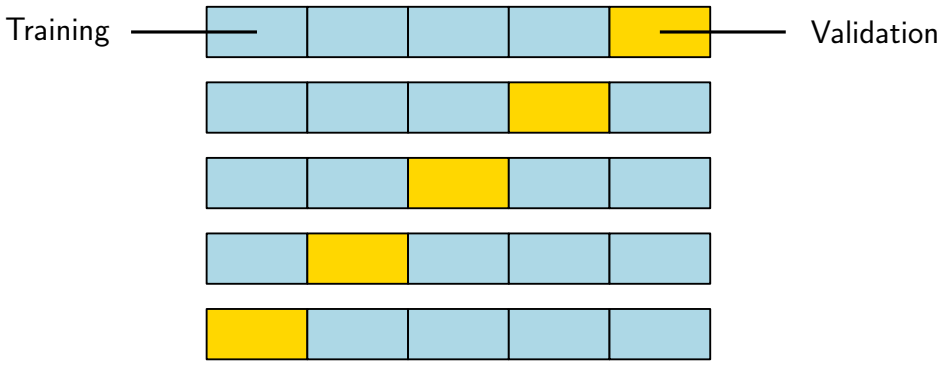


Figure 1.24: K-fold cross-validation. The data set is divided into $K = 5$ chunks, $K - 1$ of which serve as the training set (blue) and one as the validation set (yellow). This procedure is repeated for all K choices for the validation set, and the performance of the model from the K runs is averaged.

1.7.1 Cross-Validation

One way to assess the generalization performance of a model is to use a **validation set**. The validation set a small subset of the available training set that we keep aside. We partition our training set $\mathcal{D} = \tilde{\mathcal{D}} \cup \mathcal{V}$, $\tilde{\mathcal{D}} \cap \mathcal{V} = \emptyset$, where $\mathcal{V}$ is the validation set, and train our model on $\tilde{\mathcal{D}}$. After training, we assess the performance of the model on the validation set $\mathcal{V}$ (e.g., by computing RMSE values of the trained model on the validation set). We repeat this procedure for all models and choose the model that performs best. Once the model is chosen we can evaluate the final performance on the test set.

A practical issue with this approach is that the amount of data is limited, and ideally we would use as much of the data available to train the model. This would require to keep our validation set $\mathcal{V}$ small, which then would lead to a noisy estimate (with high variance) of the predictive performance. One solution to these contradictive objectives (large training set, large validation set) is to use **cross-validation**. K-fold cross-validation effectively partitions the data into K chunks, $K - 1$ of which form the training set $\tilde{\mathcal{D}}$, and the last chunk serves as the validation set $\mathcal{V}$ (similar to the idea outlined above). However, cross-validation iterates through (ideally) all combinations of assignments of chunks to $\tilde{\mathcal{D}}$ and $\mathcal{V}$, see Fig. 1.24

A disadvantage of K-fold cross-validation is the computational cost of training the model K times, which can be burdensome if the training cost is computationally expensive.

In practice, it is often not sufficient to look at the direct parameters alone. For example, we need to explore multiple complexity parameters (e.g., multiple regularization parameters), which may not be direct parameters of the model. Evaluating the quality of the model, depending on these hyper-parameters may result in a number of training runs that is exponential in the number of model parameters.

Figure 1.25: Illustration of the hierarchical generative process in Bayesian model selection. We place a prior $p(M)$ on the set of models. For each model, there is a prior $p(\boldsymbol{\theta}_k|M_k)$ on the corresponding model parameters, which are then used to generate the data $\mathcal{D}$.

1.7.2 Bayesian Model Selection

We generally assume that we have a finite number of models that we can choose from. A more efficient way than cross validation, where we have to fit each model K times (where K is the number of cross-validation folds), is to compute the **posterior distribution over models**.
Let us consider a finite number of models $M = \{M_1, \ldots, M_K\}$, where each model M_k is parametrized by $\boldsymbol{\theta}_k$. In **Bayesian model selection**, we place a prior $p(M)$ on the set of models. The corresponding **generative process** is

$$M_k \sim p(M) \tag{1.175}$$
$$\boldsymbol{\theta}_k|M_k \sim p(\boldsymbol{\theta}_k) \tag{1.176}$$
$$\mathcal{D}|\boldsymbol{\theta}_k \sim p(\mathcal{D}|\boldsymbol{\theta}_k) \tag{1.177}$$

and illustrated in Fig. 1.25. Given a training set $\mathcal{D}$, we compute the posterior over models as

$$p(M_k|\mathcal{D}) \propto p(M_k)p(\mathcal{D}|M_k). \tag{1.178}$$

Note that this posterior no longer depends on the model parameters $\boldsymbol{\theta}_k$ because they have been integrated out in the Bayesian setting.
From the posterior in (1.178), we determine the MAP estimate as

$$M^* = \arg\max_{M_k} p(M_k|\mathcal{D}). \tag{1.179}$$

With a uniform (uninformative) prior $p(M_k) = \frac{1}{K}$, determining the MAP estimate amounts to picking the model that maximizes the **model evidence (marginal likelihood)**

$$p(\mathcal{D}|M_k) = \int p(\mathcal{D}|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|M_k)d\boldsymbol{\theta}_k, \tag{1.180}$$

where $p(\boldsymbol{\theta}_k|M_k)$ is the prior distribution of the model parameters $\boldsymbol{\theta}_k$ of model M_k.

Remark 16
There are some subtle differences between a likelihood and a marginal likelihood: 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).

1.7.3 Bayes Factors for Model Comparison

Consider the problem of comparing two probabilistic models M_1, M_2, given a data set $\mathcal{D}$. If we compute the posteriors $p(M_1|\mathcal{D})$ and $p(M_2|\mathcal{D})$, we can compute the ratio of the posteriors (**posterior odds**)

$$\underbrace{\frac{p(M_1|\mathcal{D})}{p(M_2|\mathcal{D})}}_{\text{posterior odds}} = \frac{\frac{p(\mathcal{D}|M_1)p(M_1)}{p(\mathcal{D})}}{\frac{p(\mathcal{D}|M_2)p(M_2)}{p(\mathcal{D})}} = \underbrace{\frac{p(M_1)}{p(M_2)}}_{\text{prior odds}} \underbrace{\frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_2)}}_{\text{Bayes factor}} \tag{1.181}$$

The first fraction on the right-hand-side (prior odds) measures how much our prior (initial) beliefs favor M_1 over M_2. The ratio of the marginal likelihoods (second fraction on the right-hand-side) is called the **Bayes factor** and measures how well the data $\mathcal{D}$ is predicted by M_1 compared to M_2.

Remark 17
The **Jeffreys-Lindley paradox** *states that the "Bayes factor always favors the simpler model since the probability of the data under a complex model with a diffuse prior will be very small" (Murphy, 2012).*

If we choose a uniform prior over models, the prior odds term in (1.181) is 1, i.e., the posterior odds is the ratio of the marginal likelihoods (Bayes factor)

$$\frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_2)}. \tag{1.182}$$

If the Bayes factor is greater than 1, we would choose model M_1, otherwise model M_2.

1.7.4 Fully Bayesian Treatment

Instead of selecting a single "best" model, in a fully Bayesian treatment, we integrate out the corresponding model parameters $\boldsymbol{\theta}_M$ and average over all models $M_k, k = 1, \ldots, K$

$$p(\mathcal{D}) = \sum_{K=1}^{K} p(M_K) \underbrace{\int p(\mathcal{D}|\boldsymbol{\theta}_k) p(\boldsymbol{\theta}_k|M_k) d\boldsymbol{\theta}_k}_{=p(\mathcal{D}|M_k)} \tag{1.183}$$

1.7.5 Computing the Marginal Likelihood

The marginal likelihood plays an important role in model selection: We need to compute it for Bayes factors (1.181) and in the fully Bayesian setting where we additionally integrate out the model itself (1.183).
Unfortunately, computing the marginal likelihood requires us to solve an integral. This integration is generally analytically intractable, and we will have to resort to approximation techniques, e.g., numerical integration (Stoer and Burlirsch, 2002), stochastic approximations using Monte Carlo (Murphy, 2012) or Bayesian Monte Carlo techniques (O'Hagan, 1991; Rasmussen and Ghahramani, 2003).
However, there are special cases in which we can solve it. In Section 1.2.4, we discussed conjugate models. If we choose a conjugate parameter prior $p(\boldsymbol{\theta})$, we can compute the marginal likelihood in closed form.

Example (Marginal Likelihood Computation)
We consider the linear-Gaussian model with the following generative process (the

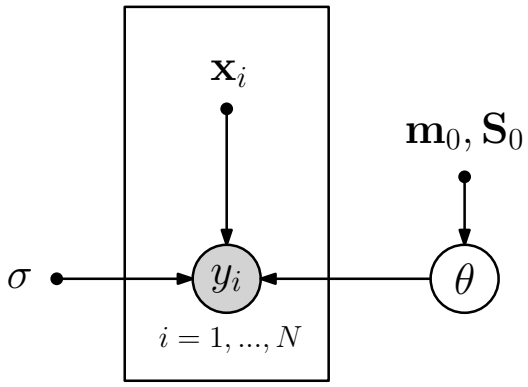


Figure 1.26: Graphical model for Bayesian linear regression.

corresponding graphical model is shown in Fig. 1.26):

$$\boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{m}_0, \boldsymbol{S}_0\right) \tag{1.184}$$

$$y_i|\boldsymbol{x}_i, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{x}_i^\top \boldsymbol{\theta}, \sigma^2\right), \tag{1.185}$$

$i = 1, \ldots, N$. We see the marginal likelihood

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{1.186}$$

$$= \int \mathcal{N}\left(\boldsymbol{y}\,|\,\boldsymbol{X}\boldsymbol{\theta}, \sigma^2 \boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{\theta}\,|\,\boldsymbol{m}_0, \boldsymbol{S}_0\right) d\boldsymbol{\theta}. \tag{1.187}$$

We compute the marginal likelihood in two steps: First, we show that the marginal likelihood is Gaussian (as a distribution in $\boldsymbol{y}$); Second, we compute the mean and covariance of this Gaussian.

1. The marginal likelihood is Gaussian: From Section 1.2.3.5 we know that (i) the product of two Gaussian random variables is an (unnormalized) Gaussian distribution, (ii) a linear transformation of a Gaussian random variable is Gaussian distributed. In (1.187), we require a linear transformation to bring $\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{\theta}, \sigma^2\boldsymbol{I}\right)$ into the form $\mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{\mu}, \Sigma\right)$ for some $\boldsymbol{\mu}, \Sigma$. Once this is done, the integral can be solved in closed form. The result is the normalizing constant of the product of the two Gaussians. The normalizing constant itself has Gaussian shape, see (1.43).

2. Mean and covariance. We compute the mean and covariance matrix of the marginal likelihood by exploiting the standard results for means and covariances of affine transformations of random variables, see Section 1.2.1.2. The mean of the marginal likelihood is computed as

$$\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{y}|\boldsymbol{X}] = \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}] = \boldsymbol{X}\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta}] = \boldsymbol{X}\boldsymbol{m}_0\,. \tag{1.188}$$

   Note that $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}\right)$ is a vector of random variables. The covariance matrix is given as

$$\mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{y}] = \mathrm{Cov}[\boldsymbol{X}\boldsymbol{\theta}] + \sigma^2\boldsymbol{I} = \boldsymbol{X}\,\mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{\theta}]\boldsymbol{X}^\top + \sigma^2\boldsymbol{I} = \boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2\boldsymbol{I} \tag{1.189}$$

Hence, the marginal likelihood is

$$p(\boldsymbol{y}|\boldsymbol{x}) = (2\pi)^{-\frac{N}{2}}|\boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2\boldsymbol{I}|^{-\frac{1}{2}}\exp\Big(-\tfrac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{m}_0)^\top(\boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{m}_0)\Big). \tag{1.190}$$

1.7.6 Further Reading

Rasmussen and Ghahramani (2001) showed that the automatic Occam's razor does not necessarily penalize the number of parameters in a model[35] but it is active in terms of the complexity of functions. They also showed that the automatic Occam's razor also holds for Bayesian non-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. The **Akaike Information Criterion (AIC)** (Akaike, 1974)

$$\ln p(\boldsymbol{x}|\boldsymbol{\theta}_{\mathrm{ML}}) - M \tag{1.191}$$

corrects for the bias of MLE by addition of a penalty term to compensate for the overfitting of more complex models (with lots of parameters). Here, M is the number of model parameters.

[35] In parametric models, the number of parameters is often related to the complexity of the model class.

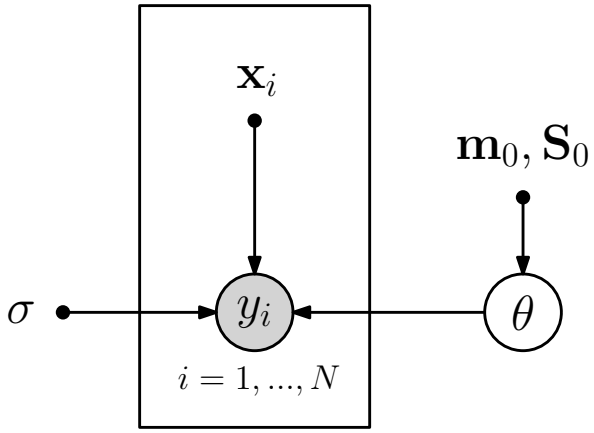


Figure 1.27: Graphical model for Bayesian linear regression.

The **Bayesian Information Criterion (BIC)** (Schwarz, 1978)

$$\ln p(\boldsymbol{x}) = \ln \int p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \approx \ln p(\boldsymbol{x}|\boldsymbol{\theta}_{\mathrm{ML}}) - \frac{1}{2} M \ln N \tag{1.192}$$

can be used for exponential family distributions. Here, N is the number of data points and M is the number of parameters. BIC penalizes model complexity more heavily than AIC.

1.8 Bayesian Linear Regression

Thus far, we looked at linear regression models where we estimated the parameters $\boldsymbol{\theta}$, e.g., by means of maximum likelihood or MAP estimation. We discovered that MLE can lead to severe overfitting, in particular, in the small-data regime. MAP addresses this issue by placing a prior on the parameters that plays the role of a regularizer.
Bayesian linear regression pushes the idea of the parameter prior a step further and does not even attempt to compute a point estimate of the parameters, but instead the full posterior over the parameters is taken into account when making predictions. This means that the parameters themselves remain uncertain.

1.8.1 Model

In Bayesian linear regression, we consider the following model

$$\begin{aligned} y &= \phi^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon \\ \epsilon &\sim \mathcal{N}\left(0, \sigma^2\right) \\ \boldsymbol{\theta} &\sim \mathcal{N}\left(\boldsymbol{m}_0, \boldsymbol{S}_0\right), \end{aligned} \tag{1.193}$$

where we now explicitly place a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{m}_0, \boldsymbol{S}_0\right)$ on the parameter vector $\boldsymbol{\theta}$.[36] The graphical corresponding graphical model is shown in Fig. 1.27.

1.8.2 Parameter Posterior

Given a training set of inputs $\boldsymbol{x}_i \in \mathbb{R}^D$ and corresponding observations $y_i \in \mathbb{R}$, $i = 1,\dots,N$, compute the posterior over the parameters using Bayes' theorem as

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}, \tag{1.194}$$

where $\boldsymbol{X}$ is the collection of training inputs and $\boldsymbol{y}$ the collection of training targets. Furthermore, $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ the parameter prior, and

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{1.195}$$

the **marginal likelihood/evidence**, which is independent of the parameters $\boldsymbol{\theta}$ and normalizes the posterior. The marginal likelihood is the likelihood averaged over all possible parameter settings (with respect to the prior distribution $p(\boldsymbol{\theta})$).
In our specific model (1.193), the posterior (1.194) can be computed in closed form as

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{\theta}\,|\,\boldsymbol{m}_N, \boldsymbol{S}_N\right), \tag{1.196}$$

$$\boldsymbol{S}_N = (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}, \tag{1.197}$$

$$\boldsymbol{m}_N = \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y}), \tag{1.198}$$

where the subscript N indicates the size of the training set. In the following, we will detail how we arrive at this posterior.
Bayes' theorem tells us that the posterior $p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})$ is proportional to the product of the likelihood $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$ and the prior $p(\boldsymbol{\theta})$:

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}, \tag{1.199}$$

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{y}\,|\,\Phi\boldsymbol{\theta}, \sigma^2\boldsymbol{I}\right), \tag{1.200}$$

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}\,|\,\boldsymbol{m}_0, \boldsymbol{S}_0\right). \tag{1.201}$$

Looking at the numerator of the posterior in (1.199), we know that the Gaussian prior times the Gaussian likelihood (where the parameters on which we place the Gaussian appears linearly in the mean) is an (unnormalized) Gaussian (see Section 1.2.3.5). If necessary, we can find the normalizing constant if we know the covariance matrix of this unnormalized Gaussian.
Before we can compute this product, e.g., by applying the results from 1.2.3.5, we need to ensure the product has the "right" form

$$\mathcal{N}\left(\boldsymbol{y}\,|\,\Phi\boldsymbol{\theta}, \sigma^2\boldsymbol{I}\right)\mathcal{N}\left(\boldsymbol{\theta}\,|\,\boldsymbol{m}_0, \boldsymbol{S}_0\right) = \mathcal{N}\left(\boldsymbol{\theta}\,|\,\boldsymbol{\mu}, \Sigma\right)\mathcal{N}\left(\boldsymbol{\theta}\,|\,\boldsymbol{m}_0, \boldsymbol{S}_0\right) \tag{1.202}$$

[36] Why is a Gaussian prior a convenient choice?

for some μ, Σ. With this form we determine the desired product immediately as

$$\mathcal{N}\left(\theta|\mu,\Sigma\right)\mathcal{N}\left(\theta|m_0,S_0\right) \propto \mathcal{N}\left(\theta|m_N,S_N\right) \tag{1.203}$$

$$S_N = (S_0^{-1}+\Sigma^{-1})^{-1} \tag{1.204}$$

$$m_N = S_N(S_0^{-1}m_0+\Sigma^{-1}\mu). \tag{1.205}$$

In the following, we discuss two ways of finding μ and Σ.

1.8.2.1 Linear Transformation of Gaussian Random Variables

To find μ and Σ, we use some basic identities for linearly transforming Gaussian random variables (see Section 1.2.3.5):

$$\mathcal{N}\left(y|\Phi\theta,\sigma^2 I\right) \overset{\text{scale by } \Phi^\top}{=} \mathcal{N}\left(\Phi y|\Phi^\top\Phi\theta,\sigma^2\Phi^\top\Phi\right) \tag{1.206}$$

$$\overset{\text{scale by } (\Phi^\top\Phi)^{-1}}{=} \mathcal{N}\left((\Phi^\top\Phi)^{-1}\Phi^\top y|\theta,\sigma^2(\Phi^\top\Phi)^{-1}\Phi^\top\Phi(\Phi^\top\Phi)^{-1}\right) \tag{1.207}$$

$$= \mathcal{N}\left(\theta|(\Phi^\top\Phi)^{-1}\Phi^\top y,\sigma^2(\Phi^\top\Phi)^{-1}\right) \tag{1.208}$$

If we now use the re-arranged likelihood (1.208) and define its mean as μ and covariance matrix as Σ in (1.205) and (1.204), respectively, we obtain

$$\mathcal{N}\left(\theta|\mu,\Sigma\right)\mathcal{N}\left(\theta|m_0,S_0\right) \propto \mathcal{N}\left(\theta|m_N,S_N\right) \tag{1.209}$$

$$S_N = (S_0^{-1}+\sigma^{-2}\Phi^\top\Phi)^{-1} \tag{1.210}$$

$$m_N = S_N(S_0^{-1}m_0+\underbrace{\sigma^{-2}(\Phi^\top\Phi)}_{\Sigma^{-1}}\underbrace{(\Phi^\top\Phi)^{-1}\Phi^\top y}_{\mu}) = S_N(S_0^{-1}m_0+\sigma^{-2}\Phi^\top y). \tag{1.211}$$

1.8.2.2 Completing the Squares

Instead of looking at the product of the prior and the likelihood, we can transform the problem into log-space and solve for μ, Σ by completing the squares.

$$\log\mathcal{N}\left(y|\Phi\theta,\sigma^2 I\right)+\log\mathcal{N}\left(\theta|m_0,S_0\right) \tag{1.212}$$

$$= -\frac{1}{2}\left(\sigma^{-2}(y-\Phi\theta)^\top(y-\Phi\theta)+(\theta-m_0)^\top S_0^{-1}(\theta-m_0)\right)+\text{const} \tag{1.213}$$

where the constant contains terms independent of θ. We will ignore the constant in the following. We now factorize (1.213), which yields

$$-\frac{1}{2}\left(\sigma^{-2}y^\top y-2\sigma^{-2}y^\top\Phi\theta+\theta^\top\sigma^{-2}\Phi^\top\Phi\theta+\theta^\top S_0^{-1}\theta-2m_0^\top S_0^{-1}\theta+m_0^\top S_0^{-1}m_0\right) \tag{1.214}$$

$$= -\frac{1}{2}\left(\theta^\top(\sigma^{-2}\Phi^\top\Phi+S_0^{-1})\theta-2(\sigma^{-2}\Phi^\top y+S_0^{-1}m_0)^\top\theta\right)+\text{const}, \tag{1.215}$$

where the constant contains the black terms in (1.214), which are independent of $\boldsymbol{\theta}$. By inspecting (1.215), we find that this equation is quadratic in $\boldsymbol{\theta}$. The fact that the unnormalized log-posterior distribution is a (negative) quadratic form implies that the posterior is Gaussian, i.e.,

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \exp(\log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})) \propto \exp(\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})) \tag{1.216}$$

$$\propto \exp\Big(-\frac{1}{2}\Big(\boldsymbol{\theta}^\top(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1})\boldsymbol{\theta} - 2(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)^\top\boldsymbol{\theta}\Big)\Big) \tag{1.217}$$

where we just used (1.215) in the last transformation.
The remaining task is it to bring this (unnormalized) Gaussian into the form that is proportional to $\mathcal{N}\big(\boldsymbol{\theta}\,|\,\boldsymbol{m}_N, \boldsymbol{S}_N\big)$ for a scaling factor, i.e., we need to identify the mean $\boldsymbol{m}_N$ and the covariance matrix $\boldsymbol{S}_N$.
Here, we use the idea of **completing the squares**. The desired log-posterior is

$$\log\mathcal{N}\big(\boldsymbol{\theta}\,|\,\boldsymbol{m}_N, \boldsymbol{S}_N\big) = -\frac{1}{2}\big((\boldsymbol{\theta}-\boldsymbol{m}_N)^\top\boldsymbol{S}_N^{-1}(\boldsymbol{\theta}-\boldsymbol{m}_N)\big) + \text{const} \tag{1.218}$$

$$= -\frac{1}{2}\big(\boldsymbol{\theta}^\top\boldsymbol{S}_N^{-1}\boldsymbol{\theta} - 2\boldsymbol{m}_N^\top\boldsymbol{S}_N^{-1}\boldsymbol{\theta} + \boldsymbol{m}_N\boldsymbol{S}_N^{-1}\boldsymbol{m}_N\big). \tag{1.219}$$

Here, we factorized the quadratic form $(\boldsymbol{\theta}-\boldsymbol{m}_N)^\top\boldsymbol{S}_N^{-1}(\boldsymbol{\theta}-\boldsymbol{m}_N)$ into a term that is quadratic in $\boldsymbol{\theta}$ alone (blue), a term that is linear in $\boldsymbol{\theta}$ (red), and a constant term (black). This allows us now to find $\boldsymbol{S}_N$ and $\boldsymbol{m}_N$ by matching the colored expressions in (1.215) and (1.219), which yields

$$\boldsymbol{S}_N^{-1} = \boldsymbol{\Phi}^\top\sigma^{-2}\boldsymbol{I}\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1} \Leftrightarrow \boldsymbol{S}_N = (\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{S}_0^{-1})^{-1}, \tag{1.220}$$

$$\boldsymbol{m}_N^\top\boldsymbol{S}_N^{-1} = (\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0)^\top \Leftrightarrow \boldsymbol{m}_N = \boldsymbol{S}_N(\sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0). \tag{1.221}$$

This is identical to the solution in (1.210)–(1.211), which we obtained by repeated linear transformation of Gaussian random variables.

Remark 18 (Completing the Squares—General Approach)
If we are given an equation

$$\boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{a}^\top\boldsymbol{x} + const_1\,, \tag{1.222}$$

where $\boldsymbol{A}$ is symmetric and positive definite, which we wish to bring into the form

$$(\boldsymbol{x}-\boldsymbol{\mu})^\top\Sigma(\boldsymbol{x}-\boldsymbol{\mu}) + const_2\,, \tag{1.223}$$

we can do this by setting

$$\Sigma = \boldsymbol{A} \tag{1.224}$$

$$\boldsymbol{\mu} = \Sigma^{-1}\boldsymbol{a} \tag{1.225}$$

and $const_2 = const_1 + \boldsymbol{\mu}^\top\Sigma\boldsymbol{\mu}$.

We can see that the terms inside the exponential in (1.217) are of the form (1.222) with

$$\boldsymbol{A} = \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{S}_0\,, \tag{1.226}$$

$$\boldsymbol{a} = \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{S}_0\boldsymbol{m}_0\,. \tag{1.227}$$

Since $\boldsymbol{A}, \boldsymbol{a}$ can be difficult to identify in equations like (1.214), it is often helpful to bring these equations into the form (1.222) that decouples quadratic term, linear terms and constants, which simplifies finding the desired solution.

1.8.3 Prediction and Inference

In practice, we are usually not so much interested in the parameter values $\boldsymbol{\theta}$. Instead, our focus often lies in the predictions we make with those parameter values. In a fully Bayesian setting, we take the full posterior distribution and average over all plausible parameter settings when we make predictions, instead of finding the maximum a-posteriori (point) estimate of the parameters (the setting $\boldsymbol{\theta}_{\text{MAP}}$ at which the posterior attains its maximum value). To predict the distribution of

$$y_* = \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta} + \epsilon\,, \quad \epsilon \sim \mathcal{N}\left(0, \sigma^2\right), \tag{1.228}$$

at a test location $\boldsymbol{x}_*$, we compute

$$p(y_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{\theta} \tag{1.229}$$

$$= \int \mathcal{N}\left(y_* \,|\, \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}, \sigma^2\right) \mathcal{N}\left(\boldsymbol{\theta} \,|\, \boldsymbol{m}_N, \boldsymbol{S}_N\right) d\boldsymbol{\theta} \tag{1.230}$$

$$= \mathcal{N}\left(y_* \,|\, \boldsymbol{m}_N^\top\boldsymbol{\phi}(\boldsymbol{x}_*), \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*) + \sigma^2\right) \tag{1.231}$$

The term $\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*)$ reflects the uncertainty associated with the parameters $\boldsymbol{\theta}$, whereas σ^2 is the noise variance. Note that $\boldsymbol{S}_N$ depends on the training inputs $\boldsymbol{X}$, see (1.210). The predictive mean coincides with the MAP estimate.

Remark 19 (Mean and Variance of Noise-Free Function Values)
In many cases, we are not interested in the predictive distribution $p(y_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*)$ of a (noisy) observation. Instead, we would like to obtain the distribution of the (noise-free) latent function values $f(\boldsymbol{x}_*) = \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}$. We determine the corresponding moments by exploiting the properties of means and variances, which yields

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{X}, \boldsymbol{y}] = \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}] = \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}] = \boldsymbol{m}_N^\top\boldsymbol{\phi}(\boldsymbol{x}_*)\,, \tag{1.232}$$

$$\mathbb{V}_{\boldsymbol{\theta}}[f(\boldsymbol{x}_*)|\boldsymbol{X}, \boldsymbol{y}] = \mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}] = \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\mathbb{V}_{\boldsymbol{\theta}}[\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}]\boldsymbol{\phi}(\boldsymbol{x}_*)\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*) \tag{1.233}$$

Remark 20 (Distribution over Functions)
The fact that we integrate out the parameters $\boldsymbol{\theta}$ induces a distribution over functions: If we sample $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$ from the parameter posterior, we obtain a single function realization $\boldsymbol{\theta}_i^\top\boldsymbol{\phi}(\cdot)$. The **mean function***, i.e., the set of all expected function values $\mathbb{E}_{\boldsymbol{\theta}}[f(\cdot)|\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}]$, of this distribution over functions is $\boldsymbol{m}_N^\top\boldsymbol{\phi}(\cdot)$. The (marginal) variances, i.e., the variance of $f(\boldsymbol{x}_*)$ at each $\boldsymbol{x}_*$, are given by $\boldsymbol{\phi}^\top(\cdot)\boldsymbol{S}_N\boldsymbol{\phi}(\cdot)$.*

1.8.3.1 Derivation

The predictive distribution $p(y_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*)$ is Gaussian: In (1.230), we multiply two Gaussians (in $\boldsymbol{\theta}$), which results in another (unnormalized) Gaussian.[37] When integrating out $\boldsymbol{\theta}$, we are left with the normalization constant, which itself is Gaussian shaped (see Section 1.2.3.5). Therefore, it suffices to determine the mean and the (co)variance of the predictive distribution, which we will do by applying the standard rules for computing means and (co)variances (see Section 1.2.1). In the following, we will use the shorthand notation $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\boldsymbol{x}_*)$.
The mean of the posterior predictive distribution $p(y_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*)$ is

$$\begin{aligned}
\mathbb{E}_\theta[y_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*] &= \mathbb{E}_\theta[\boldsymbol{\phi}_*^\top \boldsymbol{\theta} + \epsilon|\boldsymbol{X}, \boldsymbol{y}] && (1.234)\\
&= \boldsymbol{\phi}_*^\top \mathbb{E}_\theta[\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}] && (1.235)\\
&= \boldsymbol{\phi}_*^\top \boldsymbol{m}_N . && (1.236)
\end{aligned}$$

Here, we exploited that the noise is i.i.d. and that its mean is 0.
The corresponding posterior predictive variance is

$$\begin{aligned}
\mathbb{V}[y_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*] &= \mathbb{V}[\boldsymbol{\phi}_*^\top \boldsymbol{\theta} + \epsilon|\boldsymbol{X}, \boldsymbol{y}] && (1.237)\\
&\overset{i.i.d.}{=} \mathbb{V}_\theta[\boldsymbol{\phi}_*^\top \boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}] + \mathbb{V}_\epsilon[\epsilon] && (1.238)\\
&= \boldsymbol{\phi}_*^\top \mathbb{V}_\theta[\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}] \boldsymbol{\phi}_* + \sigma^2 && (1.239)\\
&= \boldsymbol{\phi}_*^\top \boldsymbol{S}_N \boldsymbol{\phi}_* + \sigma^2 . && (1.240)
\end{aligned}$$

The blue terms in the variance expression are the terms that are due to the inherent (posterior) uncertainty of the parameters, which induces the uncertainty about the latent function f. The additive green term is the variance of the i.i.d. noise variable.

Example

Fig. 1.28 shows some examples of the posterior distribution over functions, induced by the parameter posterior. The left panels show the maximum likelihood estimate, the MAP estimate (which is identical to the posterior mean function) and the 95% predictive confidence bounds, represented by the shaded area. The right panels show samples from the posterior over functions: Here, we sampled parameters $\boldsymbol{\theta}_i$ from the parameter posterior and computed the function $\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{\theta}_i$, which is a single realization of a function under the posterior distribution over functions. For low-order polynomials, the parameter posterior does not allow the parameters to vary much: The sampled functions are nearly identical. When we make the model more flexible by adding more parameters (i.e., we end up with a higher-order polynomial), these parameters are not sufficiently constrained by the posterior, and the sampled functions can be easily visually separated. We also see in the corresponding

[37] To be precise: We multiply the posterior $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$ with a distribution of a linearly transformed $\boldsymbol{\theta}$. Note that a linear transformation of a Gaussian random variable preserves Gaussianity (see Section 1.2.3.5).

panels on the left how the uncertainty increases, especially at the boundaries. Although for a 10th-order polynomial the MAP estimate yields a good fit, the Bayesian linear regression model additionally tells us that the posterior uncertainty is huge. This information can be critical, if we use these predictions in a decision-making system, where bad decisions can have significant consequences (e.g., in reinforcement learning or robotics).

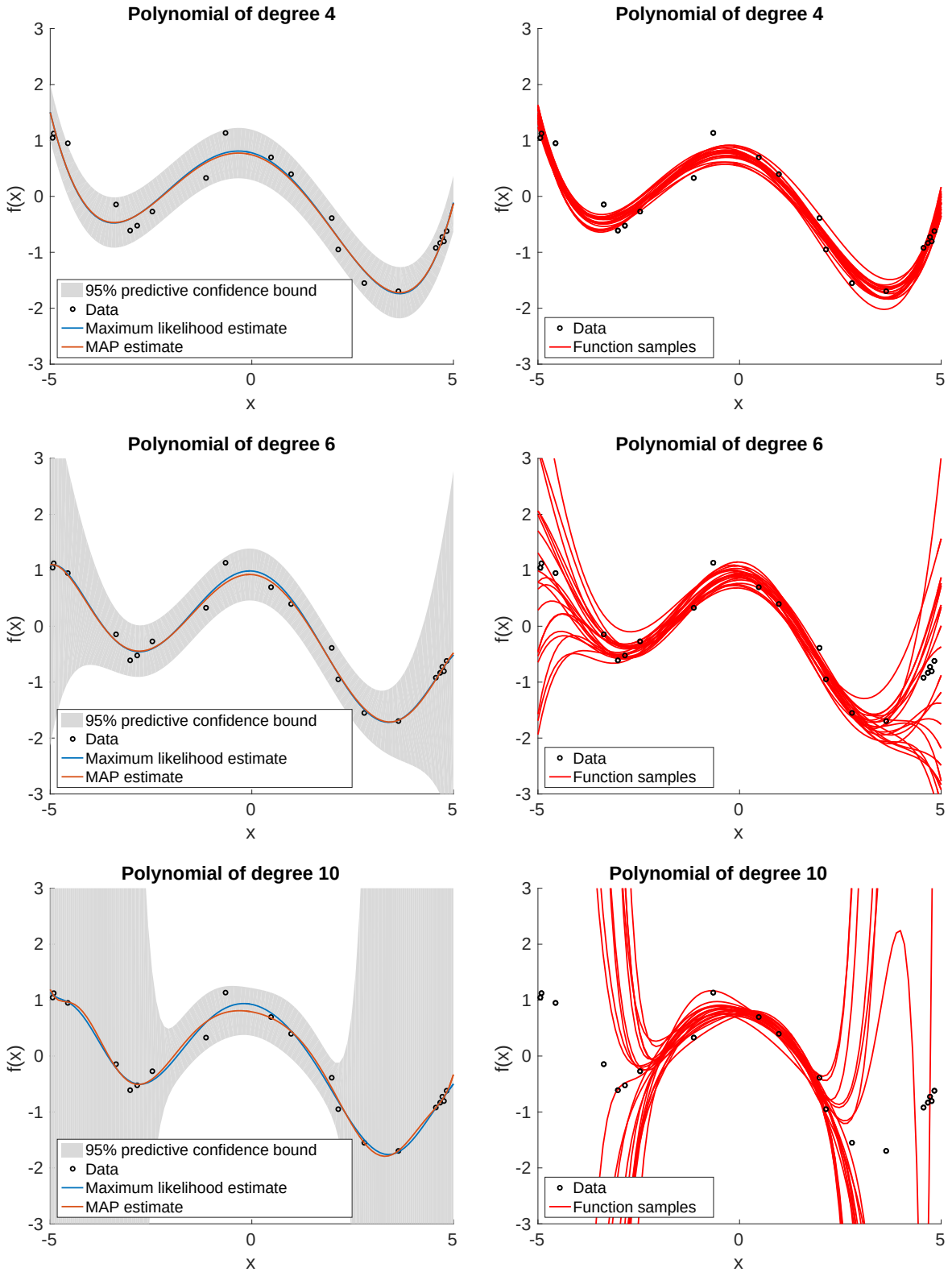


Figure 1.28: Bayesian linear regression. Left panels: The shaded area indicates the 95% predictive confidence bounds. The mean of the Bayesian linear regression model coincides with the MAP estimate. The predictive uncertainty is the sum of the noise term and the posterior parameter uncertainty, which depends on the location of the test input. Right panels: Sampled functions from the posterior distribution.

Appendix A

A.1 Scalar Products

Definition 9
For a vector space V let $\beta : V \times V \to \mathbb{R}$ be a bilinear mapping (i.e., linear in both arguments).

- *β is called* **symmetric** *if $\beta(\boldsymbol{x}, \boldsymbol{y}) = \beta(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in V$.*
- *β is called* **positive definite** *if for all $\boldsymbol{x} \neq \boldsymbol{0}$: $\beta(\boldsymbol{x}, \boldsymbol{x}) > 0$. $\beta(\boldsymbol{0}, \boldsymbol{0}) = 0$.*
- *A positive definite, symmetric bilinear mapping $\beta : V \times V \to \mathbb{R}$ is called* **scalar product/dot product/inner product** *on V. We typically write $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ instead of $\beta(\boldsymbol{x}, \boldsymbol{y})$.*
- *The pair $(V, \langle \cdot, \cdot \rangle)$ is called* **Euclidean vector space** *or (real)* **vector space with scalar product**.

Example

- For $V = \mathbb{R}^n$ we define the **standard scalar product** $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$.
- $V = \mathbb{R}^2$. If we define $\beta(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle := x_1 y_1 - (x_1 y_2 + x_2 y_1) + 2 x_2 y_2$ then β is a scalar product but different from the standard scalar product.

In a Euclidean vector space, the scalar product allows us to introduce concepts, such as lengths, distances and orthogonality.

A.1.1 Lengths, Distances, Orthogonality

Definition 10 (Norm)
Consider a Euclidean vector space $(V, \langle \cdot, \cdot \rangle)$. Then $\|\boldsymbol{x}\| := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ is the **length** *or* **norm** *of $\boldsymbol{x} \in V$. The mapping*

$$\| \cdot \| : V \to \mathbb{R} \tag{A.1}$$

$$\boldsymbol{x} \mapsto \|\boldsymbol{x}\| \tag{A.2}$$

is called **norm**.

Example (Lengths of Vectors)
In geometry, we are often interested in lengths of vectors. We can now use the scalar product to compute them. For instance, in a Euclidean vector space with standard scalar product, if $\boldsymbol{x} = [1,2]^\top$ then its norm/length is $\|\boldsymbol{x}\| = \sqrt{1^2+2^2} = \sqrt{5}$

Remark 21
The norm $\|\cdot\|$ possesses the following properties:

1. $\|\boldsymbol{x}\| \geq 0$ *for all* $\boldsymbol{x} \in V$ *and* $\|\boldsymbol{x}\| = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{0}$
2. $\|\lambda\boldsymbol{x}\| = |\lambda| \cdot \|\boldsymbol{x}\|$ *for all* $\boldsymbol{x} \in V$ *and* $\lambda \in \mathbb{R}$
3. **Minkowski inequality:** $\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ *for all* $\boldsymbol{x}, \boldsymbol{y} \in V$

Definition 11 (Distance and Metric)
Consider a Euclidean vector space $(V, \langle\cdot,\cdot\rangle)$. Then $d(\boldsymbol{x},\boldsymbol{y}) := \|\boldsymbol{x}-\boldsymbol{y}\|$ is called **distance** *of $\boldsymbol{x}, \boldsymbol{y} \in V$. The mapping*

$$d : V \times V \to \mathbb{R} \tag{A.3}$$
$$(\boldsymbol{x},\boldsymbol{y}) \mapsto d(\boldsymbol{x},\boldsymbol{y}) \tag{A.4}$$

is called **metric**.

A metric d satisfies:

1. d is positive definite, i.e., $d(\boldsymbol{x},\boldsymbol{y}) \geq 0$ for all $\boldsymbol{x},\boldsymbol{y} \in V$ and $d(\boldsymbol{x},\boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$
2. d is symmetric, i.e., $d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})$ for all $\boldsymbol{x},\boldsymbol{y} \in V$.
3. **Triangular inequality:** $d(\boldsymbol{x},\boldsymbol{z}) \leq d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z})$.

Definition 12 (Orthogonality)
Vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are **orthogonal** *if $\langle\boldsymbol{x},\boldsymbol{y}\rangle = 0$, and we write $\boldsymbol{x} \perp \boldsymbol{y}$*

Theorem 1
Let $(V, \langle\cdot,\cdot\rangle)$ be a Euclidean vector space and $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in V$. Then:

1. **Cauchy-Schwarz inequality:** $|\langle\boldsymbol{x},\boldsymbol{y}\rangle| \leq \|\boldsymbol{x}\|\,\|\boldsymbol{y}\|$
2. **Minkowski inequality:** $\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$
3. **Triangular inequality:** $d(\boldsymbol{x},\boldsymbol{z}) \leq d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z})$
4. **Parallelogram law:** $\|\boldsymbol{x}+\boldsymbol{y}\| + \|\boldsymbol{x}-\boldsymbol{y}\| = 2\|\boldsymbol{x}\|^2 + 2\|\boldsymbol{y}\|^2$

5. $4\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \|\boldsymbol{x} + \boldsymbol{y}\|^2 - \|\boldsymbol{x} - \boldsymbol{y}\|^2$
6. $\boldsymbol{x} \perp \boldsymbol{y} \Leftrightarrow \|\boldsymbol{x} + \boldsymbol{y}\|^2 = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2$

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

$$-1 \leq \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \, \|\boldsymbol{y}\|} \leq 1 \tag{A.5}$$

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

$$\cos \omega = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \, \|\boldsymbol{y}\|} \tag{A.6}$$

The number ω is the **angle** between $\boldsymbol{x}$ and $\boldsymbol{y}$.

A.1.2 Applications

Scalar products allow us to compute angles between vectors or distances. A major purpose of scalar products is to determine whether vectors are orthogonal to each other; in this case $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. This plays an important role for projections. The scalar product also allows us to determine specific bases of vector (sub)spaces, where each vector is orthogonal to all others (orthogonal bases) using the Gram-Schmidt method. These bases are important optimization and numerical algorithms for solving linear equation systems. For instance, Krylov subspace methods[1], such as Conjugate Gradients or GMRES, minimize residual errors that are orthogonal to each other (Stoer and Burlirsch, 2002).
In machine learning, scalar products are important in the context of kernel methods (Schölkopf and Smola, 2002). Kernel methods exploit the fact that many linear algorithms can be expressed purely by scalar product computations.[2] Then, the "kernel trick" allows us to compute these scalar products implicitly in a (potentially infinite-dimensional) feature space, without even knowing this feature space explicitly. This allowed the "non-linearization" of many algorithms used in machine learning, such as kernel-PCA (Schölkopf et al., 1998) for dimensionality reduction. Gaussian processes (Rasmussen and Williams, 2006) also fall into the category of kernel methods and are the current state-of-the-art in probabilistic regression (fitting curves to data points).

A.2 Useful Matrix Identities

To avoid explicit inversion of a possibly singular matrix, we often employ the following three identities:

$$(\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1} = \boldsymbol{A}(\boldsymbol{A} + \boldsymbol{B})^{-1}\boldsymbol{B} = \boldsymbol{B}(\boldsymbol{A} + \boldsymbol{B})^{-1}\boldsymbol{A} \tag{A.7}$$

[1] The basis for the Krylov subspace is derived from the Cayley-Hamilton theorem, which allows us to compute the inverse of a matrix in terms of a linear combination of its powers.

[2] Matrix-vector multiplication $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ falls into this category since b_i is a scalar product of the ith row of $\boldsymbol{A}$ with $\boldsymbol{x}$.

$$(\boldsymbol{Z}+\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\top)^{-1} = \boldsymbol{Z}^{-1} - \boldsymbol{Z}^{-1}\boldsymbol{U}(\boldsymbol{W}^{-1}+\boldsymbol{V}^\top\boldsymbol{Z}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}^\top\boldsymbol{Z}^{-1} \tag{A.8}$$

$$(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{C})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{I}+\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}. \tag{A.9}$$

The **Searle identity** in (A.7) is useful if the individual inverses of $\boldsymbol{A}$ and $\boldsymbol{B}$ do not exist or if they are ill conditioned. The **Woodbury identity** in (A.8) can be used to reduce the computational burden: If $\boldsymbol{Z} \in \mathbb{R}^{p\times p}$ is diagonal, the inverse $\boldsymbol{Z}^{-1}$ can be computed in $\mathcal{O}(p)$. Consider the case where $\boldsymbol{U} \in \mathbb{R}^{p\times q}$, $\boldsymbol{W} \in \mathbb{R}^{q\times q}$, and $\boldsymbol{V}^\top \in \mathbb{R}^{q\times p}$ with $p \gg q$. The inverse $(\boldsymbol{Z}+\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\top)^{-1} \in \mathbb{R}^{p\times p}$ would require $\mathcal{O}(p^3)$ computations (naively implemented). Using (A.8), the computational burden reduces to $\mathcal{O}(p)$ for the inverse of the diagonal matrix $\boldsymbol{Z}$ plus $\mathcal{O}(q^3)$ for the inverse of $\boldsymbol{W}$ and the inverse of $\boldsymbol{W}^{-1}+\boldsymbol{V}^\top\boldsymbol{Z}^{-1}\boldsymbol{U} \in \mathbb{R}^{q\times q}$. Therefore, the inversion of a $p\times p$ matrix can be reduced to the inversion of $q\times q$ matrices, the inversion of a diagonal $p\times p$ matrix, and some matrix multiplications, all of which require less than $\mathcal{O}(p^3)$ computations. The **Kailath inverse** in (A.9) is a special case of the Woodbury identity in (A.8) with $\boldsymbol{W} = \boldsymbol{I}$. The Kailath inverse makes the inversion of $\boldsymbol{A}+\boldsymbol{B}\boldsymbol{C}$ numerically a bit more stable if $\boldsymbol{A}+\boldsymbol{B}\boldsymbol{C}$ is ill-conditioned and $\boldsymbol{A}^{-1}$ exists.

Bibliography

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723. pages 50

Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720. pages 2

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific. pages 44

Bickson, D., Dolev, D., Shental, O., Siegel, P. H., and Wolf, J. K. (2007). Linear Detection via Belief Propagation. In *Proceedings of the Annual Allerton Conference on Communication, Control, and Computing*. pages 21

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag. pages 1, 16, 17, 37, 38

Bryson, A. E. (1961). A Gradient Method for Optimizing Multi-stage Allocation Processes. In *Proceedings of the Harvard University Symposium on Digital Computers and Their Applications*. pages 31

Bryson, A. E. and Ho, Y.-C. (1975). *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere, New York City, NY, USA. pages

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167. pages 2

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M. A., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems*, pages 1–11. pages 44

Deisenroth, M. P. and Mohamed, S. (2012). Expectation Propagation in Gaussian Process Dynamical Systems. In *Advances in Neural Information Processing Systems*, pages 2618–2626. pages 21

Deisenroth, M. P. and Ohlsson, H. (2011). A General Perspective on Gaussian Filtering and Smoothing: Explaining Current and Deriving New Algorithms. In *Proceedings of the American Control Conference*. pages 13

Dreyfus, S. (1962). The Numerical Solution of Variational Problems. *Journal of Mathematical Analysis and Applications*, 5(1):30–45. pages 31

Gal, Y., van der Wilk, M., and Rasmussen, C. E. (2014). Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In *Advances in Neural Information Processing Systems*. pages 44

Golub, G. H. and Van Loan, C. F. (2012). *Matrix Computations*, volume 4. JHU Press. pages 1

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian Processes for Big Data. In Nicholson, A. and Smyth, P., editors, *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press. pages 44

Herbrich, R., Minka, T., and Graepel, T. (2007). TrueSkill(TM): A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems*, pages 569–576. MIT Press. pages 20, 21

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic Variational Inference. *Journal of Machine Learning Research*, 14(1):1303–1347. pages 44

Jaynes, E. T. E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press. pages 5

Jefferys, W. H. and Berger, J. O. (1992). Ockham's Razor and Bayesian Analysis. *American Scientist*, 80:64–72. pages 45

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME—Journal of Basic Engineering*, 82(Series D):35–45. pages 13

Kelley, H. J. (1960). Gradient Theory of Optimal Flight Paths. *Ars Journal*, 30(10):947–954. pages 31

Kittler, J. and Föglein, J. (1984). Contextual Classification of Multispectral Pixel Data. *IMage and Vision Computing*, 2(1):13–29. pages 21

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models*. MIT Press. pages 20

Lin, C.-J. (2006). A guide to support vector machines. *Department of Computer Science & Information Engineering, National Taiwan University, Taiwan*. pages 2

MacKay, D. J. C. (1992). Bayesian Interpolation. *Neural Computation*, 4:415–447. pages 45

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK. pages 45

Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, Inc. pages 28

McEliece, R. J., MacKay, D. J. C., and Cheng, J.-F. (1998). Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152. pages 21

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-Level Control through Deep Reinforcement Learning. *Nature*, 518:529–533. pages 44

Murphy, K. P. (2012). *Machine Learning: A Proabilistic Perspective*. MIT Press, Cambridge, MA, USA. pages 11, 13, 15, 45, 48, 49

O'Hagan, A. (1991). Bayes-Hermite Quadrature. *Journal of Statistical Planning and Inference*, 29:245–260. pages 49

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. pages 20

Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. Version 20121115. pages 28

Rasmussen, C. E. and Ghahramani, Z. (2001). Occam's Razor. In *Advances in Neural Information Processing Systems 13*, pages 294–300. The MIT Press. pages 50

Rasmussen, C. E. and Ghahramani, Z. (2003). Bayesian Monte Carlo. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 489–496. The MIT Press, Cambridge, MA, USA. pages 49

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA. pages 13, 61

Roweis, S. and Ghahramani, Z. (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2):305–345. pages 13

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536. pages 31, 43

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels—Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA. pages 61

Schölkopf, B., Smola, A. J., and Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319. pages 61

Schwarz, G. E. (1978). Estimating the Dimension of a Model. *Annals of Statistics*, 6(2):461–464. pages 51

Shental, O., Bickson, D., P. H. Siegel and, J. K. W., and Dolev, D. (2008). Gaussian Belief Propagatio Solver for Systems of Linear Equations. In *IEEE International Symposium on Information Theory*. pages 21

Shor, N. Z. (1985). *Minimization Methods for Non-differentiable Functions*. Springer. pages 44

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). TextonBoost: Joint Appearance, Shape and Context Modeling for Mulit-Class Object Recognition and Segmentation. In *Proceedings of the European Conference on Computer Vision*. pages 21

Spiegelhalter, D. and Smith, A. F. M. (1980). Bayes Factors and Choice Criteria for Linear Models. *Journal of the Royal Statistical Society B*, 42(2):213–220. pages 45

Stoer, J. and Burlirsch, R. (2002). *Introduction to Numerical Analysis*. Springer. pages 49, 61

Sucar, L. E. and Gillies, D. F. (1994). Probabilistic Reasoning in High-Level Vision. *Image and Vision Computing*, 12(1):42–60. pages 21

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Vladimir Kolmogorov, A. A., Tappen, M., and Rother, C. (2008). A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-based Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080. pages 21

Tibshirani, R. (1996). Regression Selection and Shrinkage via the Lasso. *Journal of the Royal Statistical Society B*, 58(1):267–288. pages 38

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B*, 61(3):611–622. pages 13

Toussaint, M. (2012). Some Notes on Gradient Descent. pages 42

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86. pages 2