Computational Intelligence: Part 2 (SPSC)

Franz Pernkopf

June 21, 2019

Contents

1 Introduction

The field of machine learning can be divided into three categories:

- Supervised learning: Given a set of N data samples $\mathcal{X} = \{\langle \mathbf{x}_1, t_1 \rangle, \ldots, \langle \mathbf{x}_N, t_N \rangle\}$, the goal of supervized learning is to construct a *decision function* f from data $\mathcal{X}$ that maps the inputs $\mathbf{x}_n$ to the corresponding outputs t_n. The input is usually $\mathbf{x}_n \in \mathbb{R}^d$ (e.g. vector of measurements) and t_n are the corresponding outputs. The inputs are sometimes called *feature vectors* and each entry is called a *feature*. The output values are called *target values* or *labels*. Hence, the data $\mathcal{X}$ in supervised learning is also called *labeled data*. If the outputs $t_n \in \mathbb{R}$ are real valued, the task of constructing the function f is called a regression task. In case of categorical output values i.e. $t_n \in \mathbb{N}$, e.g. $t_n \in \{1, \ldots, c\}$, where c is the number of classes, it is called a classification task and the function f is called a classifier. In case of a classification problem, the output values are also called classes. Important classification approaches are neural networks, support vector machines, or the Bayes classifier.

  The process of constructing the function f from some given data examples is also called *learning* or *training* due to its similarity to the learning process of humans. In case of a classification task, one can think of the function f as dividing the feature space into c distinct decision regions and data points are predicted according to the region they fall into. Note that the decision region of a class does not need to be connected. Since in practice the decision function is typically used to predict the target values of previously unseen inputs, an important property to consider is that the function f *generalizes* well.

- Unsupervised learning: Another main field of machine learning is unsupervised learning. In unsupervised learning one has only given a data set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ without corresponding target values. Hence the data in unsupervised learning is also called unlabeled data. Unlike in supervised learning, where one seeks to find a mapping from the data vectors to their target values, the goal of unsupervised learning cannot be so easily identified. Loosely speaking, one tries to find some kind of structures within the given data. Applications examples of unsupervised learning are clustering, density estimation and dimensionality reduction (such as principal component analysis).

  The task of clustering is to partition the data into several clusters such that the distances between examples from the same clusters are in some sense small. A cluster is a subset of the given data that is in some sense similar. The number of clusters K is usually chosen by hand. In cases where the data can be visualized, i.e. when the dimension is low, the number of the clusters can often be estimated by visual inspection of the data. In larger dimensions the choice of the number of clusters is often highly nontrivial.

- Reinforcement learning: Reinforcement learning is concerned with agents taking actions in an environment so that some notion of cumulative reward is optimized. This means that there is no immediate feedback if an action is right or wrong. Example: We have a robot able to navigate in an environment and a treasure, with many hurdles in between. The robot is supposed to find the best possible path to reach the treasure. The robot obtains feedback after exploring the paths. Another prominent example is AlphaGo (https://deepmind.com/research/alphago/) where a computer plays against a human Go player.

The first two categories are relevant for this class and will be discussed in more detail in the next sections. But before, we aim to recap the main tools of probability calculus which are required for this class.

1.1 Probability Calculus

A random variable is informally described as a variable who's values depend on outcomes of a random phenomenon. A random variable has a probability distribution, which specifies the probability of its values. A random variable can be discrete or continuous. Continuous distribution functions are called *probability density functions*, while the term *probability mass function* is used in the discrete case. In the sequel, we use the term *probability distribution* for both cases. The probability that a random variable X has a certain value x is written as $P(X = x)$, or $P(x)$ for short.

Probabilities are bounded by $0 \leq P(x) \leq 1$. The sum of the probabilities of all possible values for any discrete random variable is $\sum_x P(X = x) = 1$. In analogy, in the continuous case we have: $\int_{-\infty}^{\infty} p(x)dx = 1$.

A fair die has a uniform probability distribution (probability mass function), i.e. $P(X = x) = \frac{1}{6}$, where $x \in \{1, \ldots, 6\}$ is number of potential outcomes (see Figure 1).

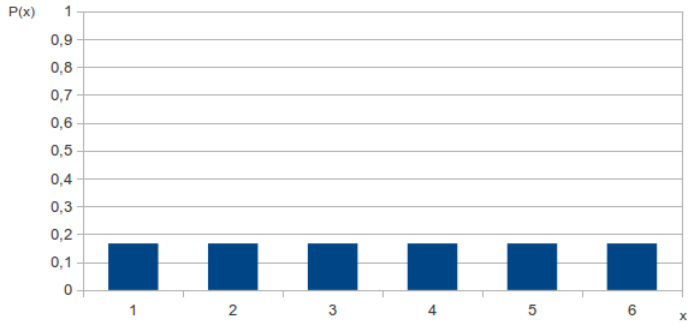


Figure 1: Discrete uniform distribution of a fair die.

When dealing with multiple random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_T\}$ we can use the joint probability distribution $P(X_1, X_2, \ldots, X_T) = P(\mathbf{X} = \mathbf{x})$ or $P(\mathbf{x})$. The joint probability distribution is defined as the probability of multiple random variables $X_1, X_2, \ldots, X_T$ jointly modeled in $P(\cdot)$. $P(\mathbf{X}, \mathbf{Y})$ is the joint probability distribution for the set $\mathbf{X} = \{X_1, \ldots, X_M\}$ and $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$ i.e. $P(\mathbf{X}, \mathbf{Y}) = P(X_1, \ldots, X_M, Y_1, \ldots, Y_N)$.

Example: Weight-Height Distribution This example provides a joint distribution of two random variables that describes the "weight-height distribution" of students. Therefore, let X be the height of students, and Y the weight of students. We can present all joint probabilities in matrix form according to Table 1. The joint probability for a specific configuration, e.g., for the probability that a

$P(X, Y)$	$X = 1$	$X = 2$	$X = 3$	$X = 4$
$Y = 1$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{20}$	0
$Y = 2$	$\frac{1}{20}$	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{10}$
$Y = 3$	0	$\frac{1}{20}$	$\frac{1}{5}$	$\frac{1}{20}$

Table 1: Joint probability distribution for the weight (X) and height (Y) of students.

student is of height $X = 2$ and of weight $Y = 1$ can be directly read from the table according to $P(X = 2, Y = 1) = 0.1$. Note that all joint probabilities must sum to one.

1.1.1 Marginalization of joint probability: Sum rule

The marginal probabilities can be acquired from joint probabilities by summing over the probabilities of all possible outcomes of the other variable, i.e.

$$P(X) = \sum_y P(X, Y = y), \tag{1}$$

and

$$P(Y) = \sum_x P(X = x, Y), \tag{2}$$

respectively. In general, marginal probabilities are defined as probability distributions over a subset of the considered random variables.

- If $\mathbf{X} = \{X_1, \ldots, X_M\}$ are discrete random variables, then $P(X_2, \ldots, X_M) = \sum_{x_1} P(X_1 = x_1, X_2, \ldots, X_M)$ or $P(X_3, \ldots, X_M) = \sum_{x_1} \sum_{x_2} P(X_1 = x_1, X_2 = x_2, X_3, \ldots, X_M)$.
- If $\mathbf{X} = \{X_1, \ldots, X_M\}$ are continuous random variables, the sum is replaced by an integral, i.e. $P(X_2, \ldots, X_M) = \int_{-\infty}^{\infty} P(X_1, X_2, \ldots, X_M) \mathrm{d}X_1$.

Example: Sum Rule Let us consider the joint probability distribution from Table 1 again. The marginal probabilities are essentially just a projection onto a single random variable (e.g., the weight) if one does not care about the realization of the other one (e.g., the height). The marginal probability for $P(X = 1)$ is consequently obtained by summing over all possible values of Y according to the sum rule so that

$$P(X = 1) = \frac{1}{10} + \frac{1}{20} = \frac{3}{20}. \tag{3}$$

1.1.2 Factorization of joint probability: Product rule

The product rule defines the factorization of $P(X, Y)$ in the following way:

$$\begin{aligned} P(X, Y) =& P(X \mid Y) P(Y) \\ =& P(Y \mid X) P(X), \end{aligned}$$

where $P(X \mid Y)$ and $P(Y \mid X)$ are conditional probabilities, while $P(X)$ and $P(Y)$ are the marginal probabilities. The conditional probability $P(X \mid Y)$ is the probability of X under the condition that Y is already known.
In analogy to the two-dimensional joint probability, the joint probability distribution of multiple random variables $X_1, \ldots, X_N$ can be factorized accordingly as:

$$\begin{aligned} P(\mathbf{X}) = P(X_1, \ldots, X_N) =& P(X_1) P(X_2 \mid X_1) \cdot \ldots \cdot P(X_N \mid X_{N-1}, \ldots, X_1) \\ =& P(X_N) P(X_{N-1} \mid X_N) \cdot \ldots \cdot P(X_1 \mid X_N, \ldots, X_2). \end{aligned}$$

This is also known as *chain rule*. The order of the variables $\mathbf{X}$ can be arbitrary in this factorization.

Example: Product rule Let us consider the joint probability distribution from Table 1 again. The conditional probability is the probability of a single random variable given that we already know the outcome of the other random variable. This would correspond to the following query: suppose we have measured a person and know that weight $X = 1$; what is the probability that this person has height $Y = 2$. The corresponding conditional probability is then obtained according to the product rule so that

$$P(Y = 2 | X = 1) = \frac{P(Y = 2, X = 1)}{P(X = 1)} \tag{4}$$

$$= \frac{1/20}{3/20} = \frac{1}{3}. \tag{5}$$

1.1.3 Bayes rule

The factorization of the joint probability $P(X, Y) = P(X \mid Y) P(Y) = P(Y \mid X) P(X)$. From this factorization the Bayes' Rule immediately follows as:

$$P(X \mid Y) = \frac{P(Y \mid X) P(X)}{P(Y)}. \tag{6}$$

1.1.4 Statistical independence

The joint probability $P(X,Y)$ factorizes as $P(X,Y) = P(X \mid Y)P(Y) = P(Y \mid X)P(X)$. Assuming statistical independence between X and Y, which is often written as $X \perp Y$, the joint probability factorizes as $P(X,Y) = P(X)P(Y)$. In this case, X doesn't contain any information about Y and vice versa, i.e. the conditional probability $P(X \mid Y) = \frac{P(X,Y)}{P(Y)} = \frac{P(X)P(Y)}{P(Y)} = P(X)$ and $P(Y \mid X) = P(Y)$. This can be extended to conditional statistical independence. Let's assume three random variables X, Y, and Z. X is statistically independent of Y conditioned on Z if and only if $P(X,Y|Z) = P(X|Y,Z)P(Y|Z) = P(X|Z)P(Y|Z)$.

2 Modeling of data with probability density functions

Let's assume data $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is given, where $\mathbf{x}_n$ is an d-dimensional random vector

$$\mathbf{x}_n = \begin{bmatrix} x_n^1 \\ \vdots \\ x_n^d \end{bmatrix}, \tag{7}$$

i.e. $\mathbf{x}_n \in \mathbb{R}^d$ where d denotes the dimension of the vector.
We aim to model $\mathcal{X}$ with a probability distribution $P(\mathbf{x})$. This probability distribution $P(\mathbf{x})$ can be parametric or non-parametric.

2.1 Non-Parametric model estimation

In contrast to a parametric model, in the non-parametric model we don't make any a-priori assumptions about the distribution of the data. Instead, the distribution is deduced from the data.
In a parametric model, we have to assume a certain kind of probability distribution beforehand, e.g. a normal distribution. Furthermore, in a parametric model the number and type of parameters is given by the selected parametric probability distribution.
For illustration, we assume that our data is in $\mathbb{R}^1$. The empiric density function is a sum of dirac functions $\delta(x)$ placed at the data samples, i.e.

$$P^e(x) = \frac{1}{N} \sum_{n=1}^{N} \delta(x - x_n),$$

where $\frac{1}{N}$ ensures that we have a normalized $P^e(x)$, i.e.

$$\int_{-\infty}^{\infty} P^e(x)dx = 1.$$

The dirac function $\delta(\cdot)$ is defined as

$$\delta(x) = \begin{cases} +\infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x)dx = 1.$$

An example of an empirical distribution is shown in Figure 4. The non-parametric model needs the data $\mathcal{X}$ for the representation of $P^e(x)$. The empirical distribution $P^e(x)$ has a high resolution, but is not smooth. To make the probability distribution more realistic, a so-called kernel can used to smoothen the distribution.

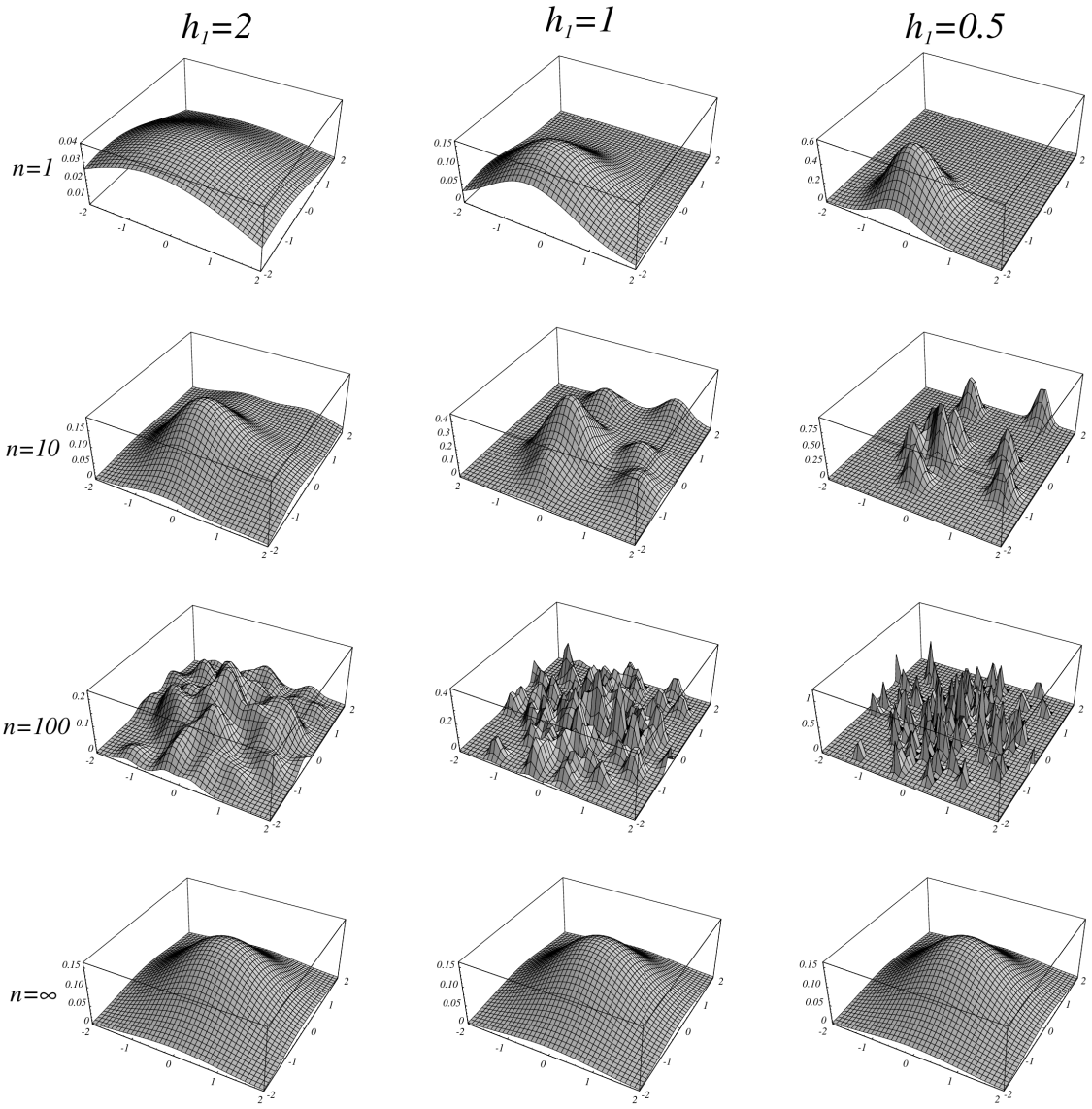


Figure 2: Smoothened non-parametric density function for a bivariate Gaussian with varying numbers of samples n and kernel widths h. For $n = \infty$ samples the estimates must always yield the true distribution (what is a bivariate Gaussian). Figure from [Duda et al., 2001].

2.1.1 Kernel density estimation

Here, we use a kernel to smooth the empirical distribution. A common choice is the Gaussian kernel $h(x)$, i.e.

$$h(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right).$$

The parameter σ controls the width (variance) of the Gaussian kernel.
The smoothened density function is determined as the convolution of the empirical density function

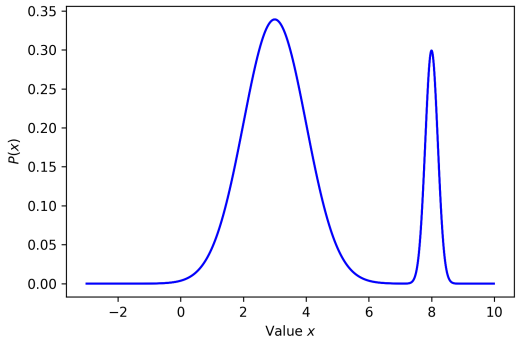


Figure 3: True distribution that we aim to approximate using kernel density estimation.

$P^e(x)$ with the kernel $h(x)$, i.e.

$$\begin{aligned} P(x) =& h(x) * P^e(x) \\ =& \int_{-\infty}^{\infty} h(x-\xi) P^e(\xi) d\xi \\ =& \int_{-\infty}^{\infty} h(x-\xi) \frac{1}{N} \sum_{n=1}^{N} \delta(\xi - x_n) d\xi \\ =& \frac{1}{N} \sum_{n=1}^{N} \int_{-\infty}^{\infty} h(x-\xi) \delta(\xi - x_n) d\xi \\ =& \frac{1}{N} \sum_{n=1}^{N} h(x-x_n). \end{aligned}$$

The kernel density function is a normalized sum of N kernels with the mean of the kernels lying on the given data samples x_n, i.e. a kernel lies on each data point x_n. Figure 5 shows a smoothened density function using a Gaussian kernel. Since this distribution is based on only 5 samples, it deviates strongly from the true distribution shown in 3. The variance σ^2 determines the intensity of smoothing; In other words, the smoothing is strong if the variance is large. If the variance is very small, the smoothened function looks similar to the empirical one in Figure 4. The influence of the variance-choice (kernel width) on the final kernel density estimation is illustrated for a bivariate Gaussian in Figure 2.

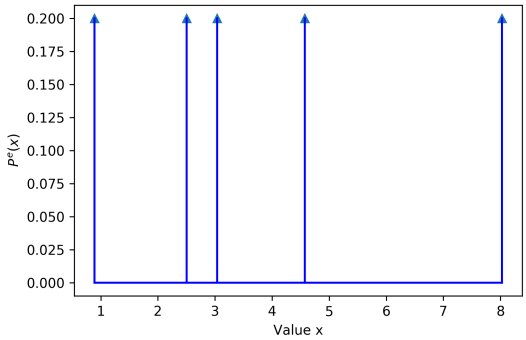


Figure 4: Empirical distribution.

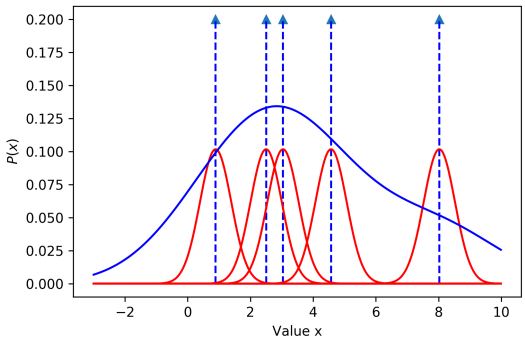


Figure 5: Kernel density estimation.

Example: Kernel Density Estimation Let us now apply kernel density estimation to the mixture of two Gaussians $P(X) = 0.85\mathcal{N}(\mu = 3, \sigma = 1) + 0.15\mathcal{N}(\mu = 8, \sigma = 0.2)$ depicted in Figure 3.
Now we are provided with 5 random samples $x_1, \ldots, x_5$ drawn according to $P(X)$. We evaluate the empirical density function $P^e(X)$ as a sum of dirac functions and a smoothed version $P(X)$ using Gaussian kernels $h(x)$ with a standard deviation of $\sigma^2 = 0.8$. The estimated non-parametric distributions are illustrated in Figure 4 and Figure 5.

2.2 Parametric model estimation

In the parametric approach, one usually makes an a-priori assumption about the density function for modeling the data, i.e. $\mathbf{x} \sim P(\mathbf{x} \mid \boldsymbol{\Theta})$. This distribution is specified by its parameters $\boldsymbol{\Theta}$. We present two estimation approaches for estimating parametric models from data: (i) Maximum-Likelihood (ML) estimation and (ii) Bayesian estimation. Both methods are introduced below. But before, we introduce the multivariate Gaussian distribution which is used throughout the lecture.

2.2.1 Multivariate Gaussian distribution

The multivariate Gaussian distribution defined as follows:

$$P(\mathbf{x} \mid \boldsymbol{\Theta}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\Theta}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right),$$

where $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix and d denotes the dimensionality of $\mathbf{x}$, i.e. $\mathbf{x} \in \mathbb{R}^d$.
The parameters of a Gaussian distribution are $\boldsymbol{\Theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, where $\boldsymbol{\mu}$ is the vector of mean values and $\boldsymbol{\Sigma}$ the covariance matrix with dimensions $d \times d$.
Every covariance matrix $\boldsymbol{\Sigma}$ has the following properties: It is symmetric, i.e. $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$ and positive semi definite (each eigenvalue of $\boldsymbol{\Sigma} \geq 0$), i.e. $\mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} \geq 0, \forall \mathbf{x} \neq 0$.

In the following examples, a Gaussian density function in $\mathbf{x} \in \mathbb{R}^2$ is shown, using three different covariance matrices. Data samples on the circles/ellipses (also known as *contours* of the probability function) have the same likelihood.

(a) Figure 6 shows the probability density function of a Gaussian with a scaled unity covariance matrix $\sigma^2 \cdot \mathbf{I}$, i.e.,

$$\Sigma = \sigma^2 \cdot \mathbf{I} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}. \tag{8}$$

The variance has the same value in both dimensions (coordinate axes). This can be seen by the concentric circles (probability density function contours) around the mean. This kind of Gaussian is

also known as *spherically-shaped* Gaussian.

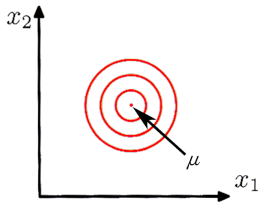


Figure 6: Gaussian distribution in $\mathbb{R}^2$ with a scaled unity matrix as its covariance matrix [Bishop, 2007].

(b) Next, we show a probability density function of a Gaussian, which has a diagonal covariance matrix with different variances on the main diagonal, i.e.

$$\Sigma = \begin{bmatrix} \sigma_{11}^2 & 0 \\ 0 & \sigma_{22}^2 \end{bmatrix}. \tag{9}$$

The density function is visualized in Figure 7. The main diagonals of the ellipses are aligned with the axes of the coordinate system. Furthermore, we can see that the variance is larger in x_1 than in x_2 (i.e. $\sigma_{11}^2 > \sigma_{22}^2$). If the covariance matrix is diagonal (as in (a) and (b)), the individual features (or dimensions) of the data are uncorrelated.

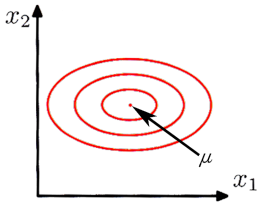


Figure 7: Gaussian distribution with a diagonal covariance matrix in $\mathbb{R}^2$ [Bishop, 2007].

(c) The third case is a general covariance matrix, i.e.

$$\Sigma = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}. \tag{10}$$

Figure 8 shows the probability density function. The main axes of the ellipses are not aligned with the coordinate system, which means that the individual features of the data are correlated.

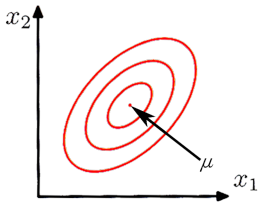


Figure 8: Gaussian distribution with a general covariance matrix in $\mathbb{R}^2$ [Bishop, 2007].

Example: Bivariate Gaussian Distribution For a 3-dimensional illustration of a bivariate Gaussian we refer back to the last line in Figure 2.

2.2.2 Maximum-Likelihood (ML) estimator

The parameters of a parametric model can be estimated from the data $\mathcal{X}$ with the ML-estimator. Assume some given data $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$; The goal is now to estimate the parameters $\boldsymbol{\Theta}$ of the model. As the name suggests, the ML estimator obtains values for parameters $\boldsymbol{\Theta}$, which maximize the likelihood of the model on the data $\mathcal{X}$. Therefore, the data $\mathcal{X}$ is fixed, and the parameters to estimate $\boldsymbol{\Theta}$ vary.
The likelihood function is defined as:

$$P(\mathcal{X} \mid \boldsymbol{\Theta}) = P(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\Theta}) = P(\mathbf{x}_1 \mid \boldsymbol{\Theta}) P(\mathbf{x}_2 \mid \mathbf{x}_1, \boldsymbol{\Theta}) \cdots P(\mathbf{x}_N \mid \mathbf{x}_{N-1}, \ldots, \mathbf{x}_1, \boldsymbol{\Theta}).$$

If the samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are independent identically distributed (**iid**[1]), we can compute the likelihood function as the product of all individual likelihoods of $\mathbf{x}_n$, i.e.

$$P(\mathcal{X} \mid \boldsymbol{\Theta}) = \prod_{n=1}^{N} P(\mathbf{x}_n \mid \boldsymbol{\Theta}).$$

In the next step, we take the logarithm of the likelihood function. The advantage is, that the product of the samples is now converted to a sum. This avoids numerical problems with very large N.
The log-likelihood function is:

$$\begin{aligned} L(\mathcal{X} \mid \boldsymbol{\Theta}) &= L(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\Theta}) = \ln P(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \boldsymbol{\Theta}) \\ &= \ln \left[\prod_{n=1}^{N} P(\mathbf{x}_n \mid \boldsymbol{\Theta}) \right] \\ &= \sum_{n=1}^{N} \ln(P(\mathbf{x}_n \mid \boldsymbol{\Theta})). \end{aligned}$$

In the Maximum-Likelihood method we want to find the parameters $\boldsymbol{\Theta}$, that maximize $L(\mathcal{X} \mid \boldsymbol{\Theta})$. Since the parameters are the log-likelihood's argument, we can write:

$$\boldsymbol{\Theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} L(\mathcal{X} \mid \boldsymbol{\Theta}).$$

To actually find these parameters $\boldsymbol{\Theta}_{\mathrm{ML}}$, the log-likelihood is differentiated w.r.t. $\boldsymbol{\Theta}$ and the derivative is set to 0

$$\frac{\partial L(\mathcal{X} \mid \boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \stackrel{!}{=} \mathbf{0}.$$

The log-likelihood function of a Gaussian distribution is:

$$L(\mathcal{X} \mid \boldsymbol{\Theta}) = \sum_{n=1}^{N} \ln \left(\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\Theta}) \right). \tag{11}$$

After differentiation of (11) w.r.t. to the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we set the derivatives to zero. This leads

[1] iid means, that the samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are drawn statistically independent from the same probability distribution.

to the following equations for parameter estimation $\boldsymbol{\Theta}_{\mathrm{ML}} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$:

$$\begin{aligned}\boldsymbol{\mu} =& \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n,\\ \boldsymbol{\Sigma} =& \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu})(\mathbf{x}_n-\boldsymbol{\mu})^T.\end{aligned}$$

Example: Maximum Likelihood Estimator Let us use the maximum likelihood method to optimize the parameters of a Gaussian distribution given some samples. We want to find a model that describes the weekly coffee consumption of students. For that purpose we are given a list of $N = 6$ iid samples (shown in Table 2) that lists the weekly coffee consumption $x_1, \ldots, x_N$ of students $S_1, \ldots, S_N$ Now we use the maximum likelihood estimator of a Gaussian as derived above for the special case of a

Student	Coffee Intake
S_1	$x_1 = 1$
S_2	$x_1 = 4$
S_3	$x_1 = 5$
S_4	$x_1 = 6$
S_5	$x_1 = 8$
S_6	$x_1 = 10$

Table 2: Samples representing the weekly coffee intake of students.

univariate Gaussian. Hence, we can estimate the mean and the variance according to

$$\mu = \frac{1}{6}34 \cong 5.5, \tag{12}$$

$$\sigma^2 = \frac{1}{6}\sum_{n=1}^{N}(\mathbf{x}_n-\mu)^2 \tag{13}$$

$$= \frac{1}{6}(4.5^2+1.5^2+0.5^2+0.5^2+2.5^2+4.5^2) \cong 8.25. \tag{14}$$

2.2.3 Bayes estimator

In the ML-estimator, $\boldsymbol{\Theta}$ is assumed to be deterministic and unknown. The Bayes estimator, on the other hand, models $\boldsymbol{\Theta}$ as a random variable, which means that there is a probability density function for $\boldsymbol{\Theta}$; The Bayes estimator makes use of Bayes' rule to estimate the probability distribution of $\boldsymbol{\Theta}$ given the data $\mathcal{X}$, i.e.

$$\underbrace{P(\boldsymbol{\Theta} \mid \mathcal{X})}_{\text{posterior probability}} = \frac{P(\overbrace{\mathcal{X} \mid \boldsymbol{\Theta}}^{\text{Likelihood; a-priori probability}}\overbrace{P(\boldsymbol{\Theta})}^{\text{Likelihood; a-priori probability}})}{P(\mathcal{X})}. \tag{15}$$

A special case of the Bayes estimator is the Maximum-a-posteriori (MAP) estimator. There, one takes the set of parameters, which maximize the posterior probability $P(\boldsymbol{\Theta} \mid \mathcal{X})$, i.e.:

$$\boldsymbol{\Theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\Theta}} P(\boldsymbol{\Theta} \mid \mathcal{X}) = \arg\max_{\boldsymbol{\Theta}} \left[P(\mathcal{X} \mid \boldsymbol{\Theta})P(\boldsymbol{\Theta})\right].$$

$P(\mathcal{X})$ in (15) can be neglected, as it is only a scaling factor, which doesn't influence the result. Furthermore, if the prior probability distribution $P(\boldsymbol{\Theta})$ is uniform, the prior is said to be non-informative. In that case, we have $\boldsymbol{\Theta}_{\mathrm{MAP}} = \boldsymbol{\Theta}_{\mathrm{ML}}$ as $\arg\max_{\boldsymbol{\Theta}} P(\boldsymbol{\Theta} \mid \mathcal{X}) = \arg\max_{\boldsymbol{\Theta}} P(\mathcal{X} \mid \boldsymbol{\Theta})$. The posterior probability distribution $P(\boldsymbol{\Theta} \mid \mathcal{X}) = f(\boldsymbol{\Theta} \mid \mathbf{y})$ is shown in Figure 9. Alternatives for the maximum of the posterior probability distribution $P(\boldsymbol{\Theta} \mid \mathcal{X})$ are the median or the mean; The median provides better results in many applications.

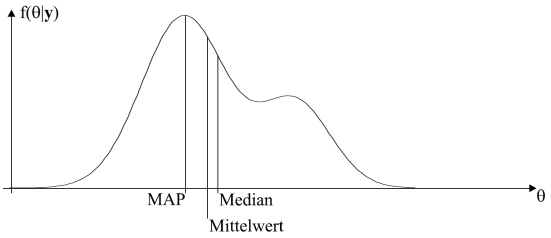


Figure 9: Posterior probability distribution of the Bayes estimator [Bishop, 2007].

3 Bayes Classifier

The aim of classification is to assign objects, represented as feature vector $\mathbf{x} \in \mathbb{R}^d$, to a particular class. Therefore, we can used the Bayes rule.
Let's assume a labeled data set:

$$\begin{aligned}
&\mathcal{X} = \{\langle \mathbf{x}_1, t_1 \rangle, \ldots, \langle \mathbf{x}_N, t_N \rangle\} \\
&\mathbf{x} \in \mathbb{R}^d \\
&t \in \mathbb{N} \qquad t \in \{1, 2, \ldots, c\} \\
&c \ldots \text{number of classes} \\
&N \ldots \text{number of samples}
\end{aligned}$$

Based on this data, the aim is to learn a decision function $g(\cdot)$ from $\mathcal{X}$ that maps a new sample $\mathbf{x}$ to a particular class (with good generalization ability).

In probabilistic classification, we exploit the posterior probability for class t given the feature vector $\mathbf{x}$ of the object, i.e. $P(t \mid \mathbf{x})$ is used. If in a two-class problem $P(t = 1 \mid \mathbf{x}) > P(t = 2 \mid \mathbf{x})$, then class one with larger probability is selected. This posterior probability can be expressed by Bayes rule according to:

$$\underbrace{P(t|\mathbf{x})}_{\text{posterior probability}} = \frac{\overbrace{P(\mathbf{x}|t)}^{\text{Likelihood; prior probability}} \overbrace{P(t)}}{\underbrace{\sum_{t'} P(\mathbf{x}|t')P(t')}_{\sum_{t'} P(\mathbf{x}|t')P(t') = \sum_{t'} P(\mathbf{x},t') = P(\mathbf{x})}}.$$

The term $P(\mathbf{x})$ (also called *evidence* term) scales the posterior probability, i.e. $P(\mathbf{x})$ scales $P(t \mid \mathbf{x})$ such that $\sum_t P(t \mid \mathbf{x}) = 1$. $P(\mathbf{x})$ is the same for each of the classes t. To obtain the most probable class, we select the class t^* with the largest posterior probability, i.e. $P(t^* \mid \mathbf{x}) > P(t \mid \mathbf{x}) \quad \forall t \neq t^*$. This can be expressed as:

$$\begin{aligned}
t^* &= \arg\max_t P(t|\mathbf{x}) = \arg\max_t \left[P(\mathbf{x}|t)P(t)\right] \\
&= \arg\max_t \left[\underbrace{\ln P(\mathbf{x}|t) + \ln P(t)}_{g_t(\mathbf{x}) \ldots \text{ decision function}} \right],
\end{aligned}$$

where we neglect $P(\mathbf{x})$ as it is the same for each of the classes t. The decision function for class t is denoted as $g_t(\mathbf{x})$. If we have a uniform class prior $P(t)$, the classification process simplifies to $t^* = \arg\max_t \ln P(t|\mathbf{x}) = \arg\max_t \ln P(\mathbf{x}|t)$. This is also known as maximum likelihood (ML) Bayes classifier.

Example: ML-Classifier for male/female students Our aim is to build a classifier for classifying students based on their weight and height into two classes, namely their gender. Let's assume that

the samples for each of the classes is Gaussian distributed, i.e. the likelihood term in the Bayes rule $P(\mathbf{x}_n|t) = \mathcal{N}(\mathbf{x}_n, |\ \mu_t, \Sigma_t)$. Hence, we model $P(\mathbf{x}_n|t)$ as parametric model. The objects $\mathbf{x}_n \in \mathbb{R}^2$ are represented by two features, i.e. weight and height, as:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} Weight \\ Height \end{bmatrix}.$$

First, the classifier has to be trained. Therefore, the data $\mathcal{X} = \{\langle \mathbf{x}_1, t_1 \rangle, \ldots, \langle \mathbf{x}_n, t_n \rangle\}$ is used. During training, the parameters $\boldsymbol{\mu}_{t=1}, \boldsymbol{\Sigma}_{t=1}, \boldsymbol{\mu}_{t=2}, \boldsymbol{\Sigma}_{t=2}, P(t=1)$ and $P(t=2)$ are determined.
To this end, the data $\mathcal{X}$ is divided according to the class label into two sets, where $\mathcal{X}^i$ denotes the samples with $t = i$. In particular, we have

$$\begin{aligned} \text{Class } t = 1: \quad & \mathcal{X}^1 = \{\mathbf{x}_n | t_n = 1\}, \\ t = 2: \quad & \mathcal{X}^2 = \{\mathbf{x}_n | t_n = 2\}. \end{aligned}$$

We can use ML estimation (see Section 2.2.2) to estimate the parameters of the Gaussian distributions $\boldsymbol{\Theta}_1 = \{\boldsymbol{\mu}_{t=1}, \boldsymbol{\Sigma}_{t=1}\}, \boldsymbol{\Theta}_2 = \{\boldsymbol{\mu}_{t=2}, \boldsymbol{\Sigma}_{t=2}\}$ from data set $\mathcal{X}^i$ of class t^i. Assuming a uniform class prior $P(t)$, the classification process is $t^* = \arg\max_t \ln P(\mathbf{x}|t) = \arg\max_t \ln \mathcal{N}(\mathbf{x}, |\ \mu_t, \Sigma_t)$. Figure 10 shows the observed samples for male and female students according to their weight/height and the decision boundary of the maximum likelihood classifier.

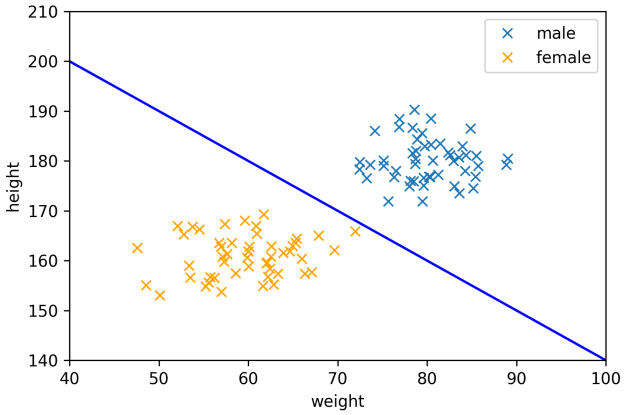


Figure 10: Distribution of data for 2 classes and the decision boundary of the maximum likelihood classifier.

Example: Bayes classifier for male/female students For the Bayes classifier we incorporate the prior class probabilities. Let us consider the male/female classification problem from above again. Therefore, we have to estimate the prior probability $P(t)$ for the classes. This can be done according to $P(t=i) = \frac{|\mathcal{X}^i|}{|\mathcal{X}|}$, where $|\mathcal{X}|$ denotes the number of samples in the set $\mathcal{X}$. This time there is a higher prior probability for a student to be of female gender, i.e., $P(t = female) > P(t = male)$; Figure 11 illustrates how the decision boundary is consequently shifted upwards.

3.1 Analysis of the decision boundary

Similar as in the example above, we use the Gaussian distribution for the likelihood term $P(\mathbf{x} \mid t)$, i.e.

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\Theta}_t) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \frac{1}{|2\pi|^{\frac{d}{2}} |\boldsymbol{\Sigma}_t|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_t)^T \boldsymbol{\Sigma}_t^{-1} (\mathbf{x} - \boldsymbol{\mu}_t)\right]. \tag{16}$$

Keep in mind that this likelihood function $P(\mathbf{x} \mid t)$ could be also modeled by a non-parametric kernel density function, a Gaussian Mixture Model (GMM) (see Section 4), an Hidden Markov Model (HMM)

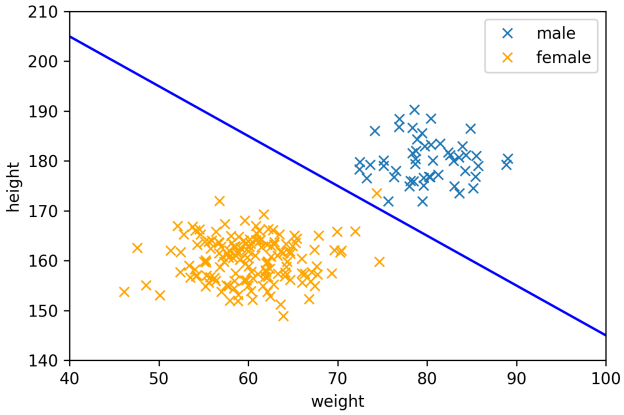


Figure 11: Distribution of Data for 2 classes and the decision boundary of the Bayes classifier.

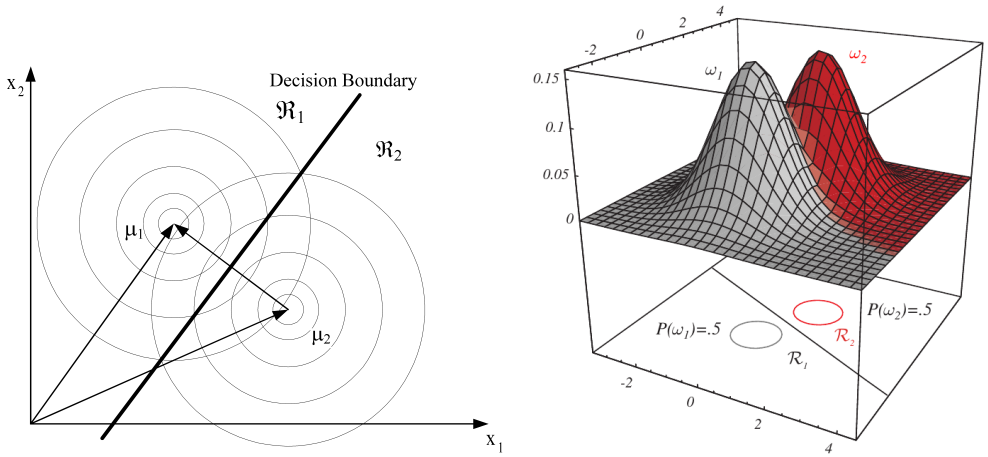
(see Section 7) or any other kind of distribution.
The decision function $g_t(\mathbf{x})$ for $P(\mathbf{x} \mid t) = \mathcal{N}(\mathbf{x} \mid \mathbf{\Theta}_t)$ is:

$$g_t(\mathbf{x}) = \ln P(t) + \ln P(\mathbf{x} \mid \mathbf{\Theta}_t) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_t)^T \mathbf{\Sigma}_t^{-1}(\mathbf{x} - \boldsymbol{\mu}_t) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{\Sigma}_t| + \ln P(t). \tag{17}$$

In the following, we show three cases of 2-class classifiers using different covariance matrices (see Section 2.2).

- Case 1: Assume same spherical covariance matrix for each class, i.e.

$$\Sigma_t = \sigma^2 \mathbf{I} \quad \forall t. \tag{18}$$



(a) (b)

Figure 12: Linear decision boundary for spherical covariance matrix [Duda et al., 2001].

Considering the same spherical covariance matrix in (18), the decision function in (17) simplifies to

$$g_t(\mathbf{x}) = -\frac{\overbrace{(\mathbf{x} - \boldsymbol{\mu}_t)^T(\mathbf{x} - \boldsymbol{\mu}_t)}^{\text{squared Euclidean distance}}}{2\sigma^2} + \ln P(t). \tag{19}$$

We see that the decision is based on the Euclidean distance and the class prior $P(t)$.
The decision boundary between class t and class i is defined as: $g_t(\mathbf{x}) = g_i(\mathbf{x}), i \neq t$, i.e, $P(i|\mathbf{x}) = P(t|\mathbf{x})$. In this case, the decision boundary is linear as shown in Figure 13. If $P(t)$ is equal for all classes, i.e uniformly distributed, then we can neglect $P(t)$ and σ^2 in (19). In this case, only the squared Euclidean distance is used for classification, i.e. we select the class represented by $\boldsymbol{\mu}_t$ which shows the smallest distance to $\mathbf{x}$. The decision function for this case is $g_t(\mathbf{x}) = -(\mathbf{x}-\boldsymbol{\mu}_t)^T(\mathbf{x}-\boldsymbol{\mu}_t)$. If both classes have different prior probabilities, i.e., $P(t_1) \neq P(t_2)$ the decision boundary is shifted as exemplified in Figure 13

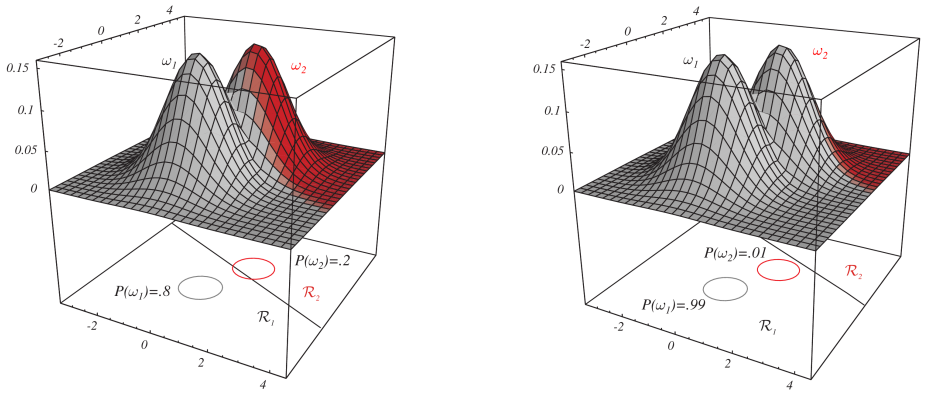


Figure 13: Shift of the decision boundary as the priors change. Note that the decision boundary may not lie between the means for certain prior distributions [Duda et al., 2001].

- Case 2: Assume same covariance matrix (not necessarily diagonal) for each class, i.e.

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}, \quad \forall t \tag{20}$$

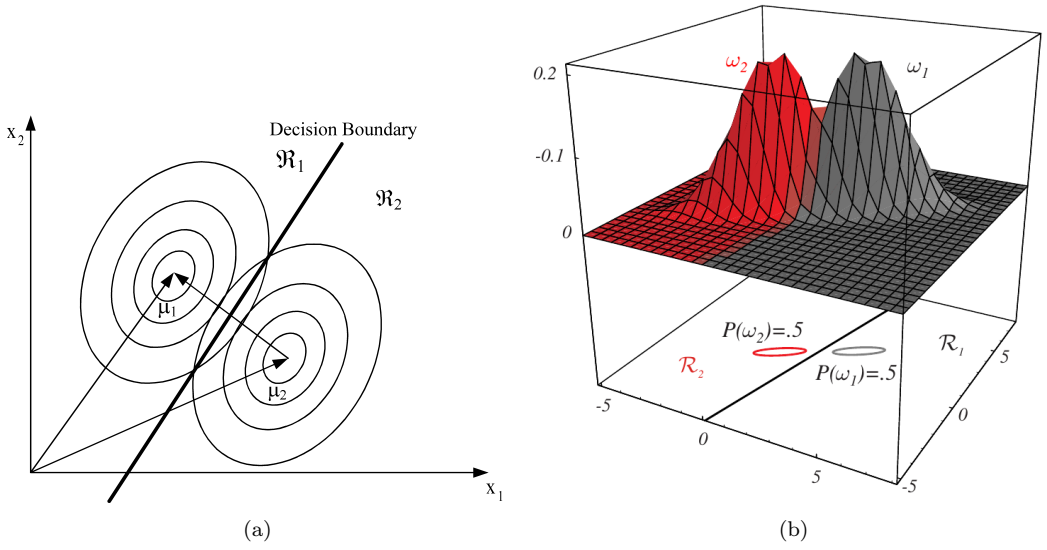


Figure 14: Linear decision boundary for same covariance matrix of both classes [Duda et al., 2001].

Considering covariances in (20), the decision function in (17) simplifies to:

$$g_t(\mathbf{x}) = -\frac{1}{2}\overbrace{(\mathbf{x}-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_t)}^{\text{Mahalanobis distance}} + \ln P(t).$$

Also in this case, the decision boundary is linear as shown in Figure 14. The decision is based on the Mahalanobis distance and $P(t)$. The Mahalanobis distance measures the distance between two points in a high-dimensional space by additionally considering the covariances.

- Case 3: Individual covariance matrix for each class $\boldsymbol{\Sigma}_t$ is assumed. This leads to the following decision function:

$$\begin{aligned} g_t(\mathbf{x}) &= \ln P(t) + \ln P(\mathbf{x} \mid \boldsymbol{\Theta}_t) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_t)^T\boldsymbol{\Sigma}_t^{-1}(\mathbf{x}-\boldsymbol{\mu}_t) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_t| + \ln P(t). \end{aligned}$$

The decision boundary is hyper-quadratic. Figure 15 shows decision boundaries for various settings of the covariance matrices. The covariance matrices for both classes are represented as ellipses or spheres.

4 Gaussian Mixture Model (GMM)

A Gaussian mixture model is used to model complex multi-modal distributions as shown in Figure 16. Multi-modal means that the distribution has more than one mode, i.e. it shows several local maxima.
A GMM is composed as a finite sum of weighted Gaussians. The probability density function is given as:

$$p(\mathbf{x} \mid \Theta) = \sum_{k=1}^{M} \alpha_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\Theta}_k),$$

where $\boldsymbol{\Theta}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$. The parameters of the distribution are $\boldsymbol{\Theta} = \{\alpha_1, \ldots, \alpha_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\}$, where K is the number of Gaussian components. The parameter $\alpha_k = P(k)$ is the weight for the k^{th} Gaussian component. For probability α_k we have the following constraints: $0 \leq \alpha_k \leq 1$ and $\sum_{k=1}^{K} \alpha_k = 1$. The scaling of $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\Theta})$ by α_k ensures that $\int_{-\infty}^{\infty} p(\mathbf{x} \mid \boldsymbol{\Theta})\mathrm{d}\mathbf{x} = 1$.

4.1 Estimation of parameters $\boldsymbol{\Theta}$

The Gaussian mixture model is a parametric model. The estimation of the model parameters $\boldsymbol{\Theta}$ can be performed by the Maximum-Likelihood (ML) method (see Section 2.2.2).

$$\begin{aligned} \text{Given are the data samples:} \quad & \mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \quad \mathbf{x} \in \mathbb{R}^d \\ \text{Aim is to estimate the parameters:} \quad & \boldsymbol{\Theta}_{\text{ML}} = \arg\max_{\boldsymbol{\Theta}} \{\ln P(\mathcal{X} \mid \boldsymbol{\Theta})\} \end{aligned}$$

First, we have to setup the log-likelihood function $L(\mathcal{X} \mid \boldsymbol{\Theta}) = \ln P(\mathcal{X}|\boldsymbol{\Theta})$ for the Gaussian mixture model. Then, we have to determine the stationary point of the log-likelihood function $L(\mathcal{X} \mid \boldsymbol{\Theta})$. This is done by $\frac{\partial \ln P(\mathcal{X}|\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \stackrel{!}{=} 0$.
Under the assumption, that the data samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ are **iid**, we obtain the following log-likelihood function

$$L(\mathcal{X} \mid \boldsymbol{\Theta}) = \ln P(\mathcal{X} \mid \boldsymbol{\Theta}) = \sum_{n=1}^{N} \ln P(\mathbf{x}_n \mid \boldsymbol{\Theta}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{21}$$

In the last step in (21), we insert $P(\mathbf{x}_n \mid \boldsymbol{\Theta})$ of the Gaussian mixture model. In the sequel, we derive the log-likelihood function for the individual parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\alpha}_k$ and set the derivative to 0.

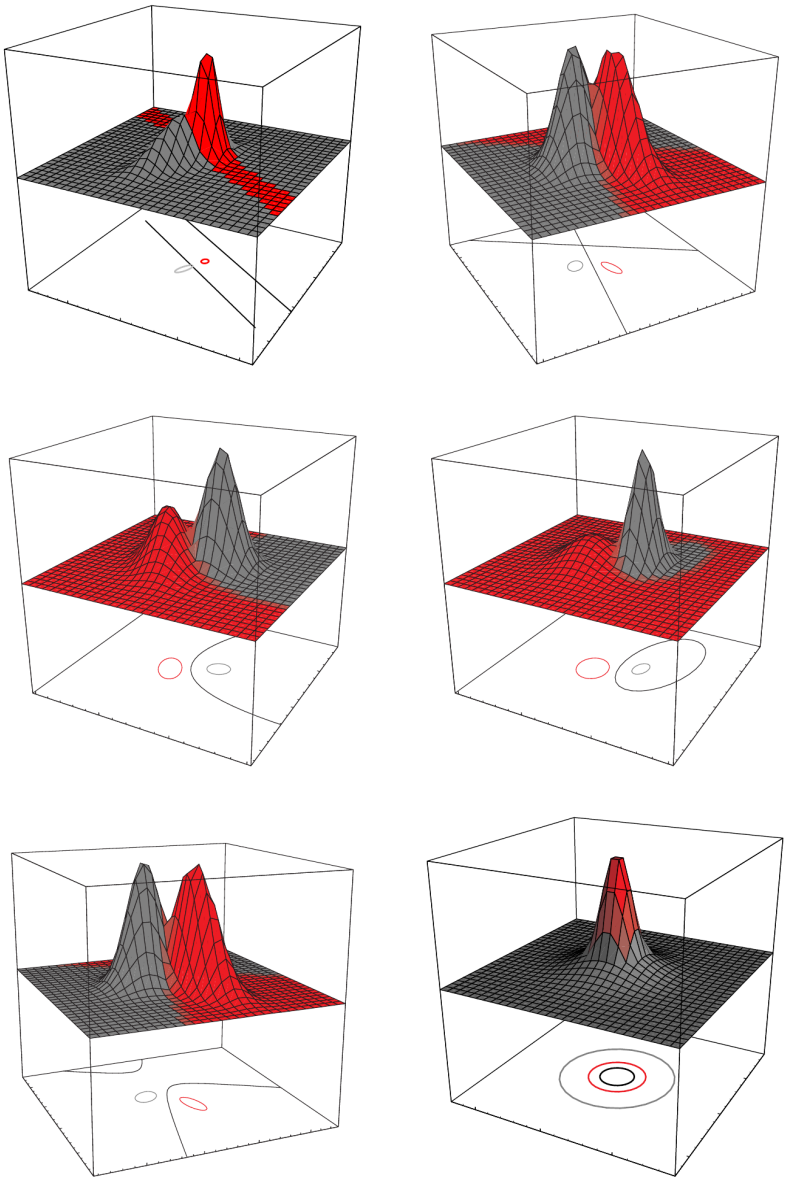


Figure 15: Hyper-quadratic decision boundaries for two Gaussian distributions with arbitrary covariance matrix. The covariance matrices are indicated by the projected ellipses [Duda et al., 2001].

4.1.1 Derivation for mean $\boldsymbol{\mu}_k$

In the following, we derive the log-likelihood function $L(\mathcal{X} \mid \boldsymbol{\Theta})$ in (21) for $\boldsymbol{\mu}_k$:

$$\frac{\partial \ln P(\mathcal{X}|\boldsymbol{\Theta})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{1}{\sum_{k'=1}^{K} \alpha_{k'} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \frac{\partial \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k}$$

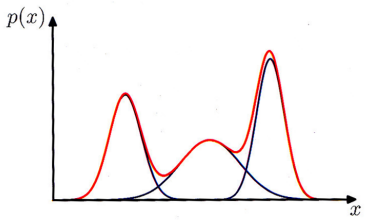


Figure 16: Gaussian mixture model [Bishop, 2007].

$$= \sum_{n=1}^{N} \underbrace{\frac{\alpha_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} \alpha_{k'} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}}_{r_k^n} \frac{\partial \left[\ln(\alpha_k) + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]}{\partial \boldsymbol{\mu}_k}. \tag{22}$$

The probability $r_k^n = P(k|\mathbf{x}_n, \Theta)$ is the posterior probability for component k given $\mathbf{x}_n$ and the parameters.
The derivation of the Gaussian distribution (see (16)) is:

$$\frac{\partial \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} = -\frac{1}{2}\left(\boldsymbol{\Sigma}_k^{-1} + (\boldsymbol{\Sigma}_k^{-1})^T\right)(\mathbf{x}_n - \boldsymbol{\mu}_k),$$

where $(\boldsymbol{\Sigma}_k^{-1} + (\boldsymbol{\Sigma}_k^{-1})^T) = 2\boldsymbol{\Sigma}_k^{-1}$, as $\boldsymbol{\Sigma}_k$ is symmetric, i.e. $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k^T$.
Hence, we obtain

$$\frac{\partial \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} = -\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k). \tag{23}$$

Inserting (23) in (22) leads to

$$\frac{\partial \ln P(\mathbf{X}|\Theta)}{\partial \boldsymbol{\mu}_k} = -\sum_{n=1}^{N} r_k^n \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k).$$

In the next step, the derivative is set to 0 and both sides of the equation are multiplied by $\boldsymbol{\Sigma}_m$, i.e. we obtain

$$\sum_{n=1}^{N}(r_k^n \mathbf{x}_n - r_k^n \boldsymbol{\mu}_k) \stackrel{!}{=} 0.$$

Furthermore, we make $\boldsymbol{\mu}_m$ explicit leading to

$$\boldsymbol{\mu}_k \sum_{n=1}^{N} r_k^n = \sum_{n=1}^{N} r_k^n \mathbf{x}_n$$

$$\boldsymbol{\mu}_k = \underbrace{\frac{\sum_{n=1}^{N} r_k^n \mathbf{x}_n}{\sum_{n=1}^{N} r_k^n}}_{N_k}.$$

Hence, the mean is computed from the data $\mathbf{x}_n$ weighted by the posterior probability r_k^n. This means that each sample $\mathbf{x}_n$ is modeled to a certain extent by each of the k Gaussian components. We introduce $N_k = \sum_{n=1}^{N} r_k^n$ as the *effective number* of data samples modeled by the Gaussian component k. For determining $\boldsymbol{\mu}_k$ we need r_k^n, which again depends on $\boldsymbol{\Theta}$. This is known as *chicken-egg* dilemma in literature. The consequence is that $\boldsymbol{\mu}_k$ is computed iteratively after initializing $\boldsymbol{\Theta}$. This enables an iterative computation of r_k^n and $\boldsymbol{\mu}_k$.

4.1.2 Derivation for $\boldsymbol{\Sigma}_k$

Here, we derive $L(\mathcal{X} \mid \boldsymbol{\Theta})$ for $\boldsymbol{\Sigma}_k$ and set the derivative to 0:

$$\frac{\partial \ln P(\mathbf{X} \mid \boldsymbol{\Theta})}{\partial \boldsymbol{\Sigma}_k} \stackrel{!}{=} 0.$$

The derivation is in full detail in [Bishop, 2007]. The solution is:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_k^n (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Again, the posterior distribution r_k^n is used to weight the samples $\mathcal{X}$. Assuming just one Gaussian component in the Gaussian mixture model, i.e. $K = 1$, results in a posterior probability $r_k^n = P(k \mid \mathbf{x}_n, \boldsymbol{\Theta}) = 1$. In this case, the parameters $\Theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ can be determined as:

$$\begin{aligned} \boldsymbol{\mu} &= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \\ \boldsymbol{\Sigma} &= \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T \end{aligned}$$

This *sanity-check* immediately leads to the ML approach for estimating a single Gaussian distribution as already shown in Section 2.2.2.

4.1.3 Derivation for α_k

Furthermore, we have to derive $L(\mathcal{X} \mid \boldsymbol{\Theta})$ for α_k and set this derivative to zero:

$$\frac{\partial \ln P(\mathcal{X} \mid \boldsymbol{\Theta})}{\partial \alpha_k} \stackrel{!}{=} 0.$$

In this case, we have an optimization problem in α with the sum-to-one constraint $\sum_{k=1}^{K} \alpha_k = 1$.
The solution can be obtained by Lagrange multipliers [Bishop, 2007]. The Lagrange function is composed by the log-likelihood function $L(\mathcal{X} \mid \boldsymbol{\Theta})$ and the constraints as follows:

$$J(k) = \ln P(\mathcal{X} \mid \boldsymbol{\Theta}) + \lambda \left(\sum_{k=1}^{K} \alpha_k - 1 \right). \tag{24}$$

In particular, the Lagrange function $J(k)$ consists of the log-likelihood function and a term consisting of the Lagrange multiplier λ and the constraint $\sum_{k=1}^{K} \alpha_k = 1$. This function $J(k)$ is derived for α_k in the sequel.
Let's first derive the log-likelihood $L(\mathcal{X} \mid \boldsymbol{\Theta})$ for α_k, i.e.

$$\frac{\partial \ln P(\mathcal{X} \mid \boldsymbol{\Theta})}{\partial \alpha_k} = \sum_{n=1}^{N} \frac{1}{\sum_{k'=1}^{K} \alpha_{k'} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \frac{\partial \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k)}{\partial \alpha_k}$$

$$= \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} \alpha_{k'} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}. \tag{25}$$

The derivative of $J(k)$ for α_k is obtained by (25) and the derivation of the constraint $\sum_{k=1}^{K} \alpha_k = 1$, i.e.,

$$\frac{\partial J(k)}{\partial \alpha_k} = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} \alpha_{k'} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'} \boldsymbol{\Sigma}_{k'})} + \lambda.$$

Let's set the derivative to zero, i.e.

$$\frac{\partial J(k)}{\partial \alpha_k} \stackrel{!}{=} 0.$$

and multiply both sides of the equation with α_k. This leads to the following equation

$$\sum_{n=1}^{N} r_k^n + \lambda \alpha_k \stackrel{!}{=} 0 \tag{26}$$

Furthermore, we sum both sides over all k components and set $N_k = \sum_{n=1}^{N} r_k^n$. This leads to the following equation:

$$\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} \lambda \alpha_k = 0.$$

Considering that $\sum_{k=1}^{K} \alpha_k = 1$, i.e. $\lambda \sum_{k=1}^{K} \alpha_k = \lambda$ and $\sum_{k=1}^{K} N_k = N$, we obtain $\lambda = -N$.
Now let's introduce $\lambda = -N$ in (26). By doing so, we obtain

$$\begin{aligned} N_k - N\alpha_k &= 0 \\ \alpha_k &= \frac{N_k}{N}. \end{aligned}$$

Similar as for the derivations of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, always the posterior probability r_k^n is necessary. This leads to an iterative algorithm for estimating the parameters $\boldsymbol{\Theta}_{\text{ML}}$ of the Gaussian mixture model. This algorithm is known as *expectation maximization* algorithm for GMMs. It is introduced in the next section.

4.2 Expectation-Maximization (EM) algorithm for learning Gaussian mixture models

The EM algorithm for learning GMMs is iterative. First the parameters $\boldsymbol{\Theta}$ are going to be initialized. In the E-Step, we can determine the posterior probabilities r_k^n based on the parameters $\boldsymbol{\Theta}$. In the maximization step (M-Step), the parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ and α_k can be updated based on r_k^n. The E and the M-Step are executed in an alternating fashion until the log-likelihood function $L(\mathcal{X}|\boldsymbol{\Theta})$ converges.
The individual steps of the EM algorithm are shown in the following.

1. Initialization: $t \ldots$ iteration counter

   $\boldsymbol{\Theta}^{t=0} = \{\alpha_k^{t=0}, \boldsymbol{\mu}_k^{t=0}, \boldsymbol{\Sigma}_k^{t=0}\}_{k=1}^{K}$

2. E-Step: Compute the posterior probability

$$r_k^n = \frac{\alpha_k^t \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k^t, \boldsymbol{\Sigma}_k^t)}{\sum_{k'=1}^{K} \alpha_{k'}^t \cdot \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_{k'}^t, \boldsymbol{\Sigma}_{k'}^t)} = P(k \mid \mathbf{x}_n, \boldsymbol{\Theta}^t)$$

$P(k \mid \mathbf{x}_n, \boldsymbol{\Theta}^t)$ is the posterior probability of k given $\mathbf{x}_n$ and $\boldsymbol{\Theta}^t$. This posterior is related to the posterior of our Bayesian classifier derived in Section 3 under the assumption that a Gaussian distribution is used to model the likelihood term.

3. M-Step: Compute the parameters Θ

$$\boldsymbol{\mu}_k^{t+1} = \frac{1}{N_k} \sum_{n=1}^{N} r_k^n \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{t+1} = \frac{1}{N_k} \sum_{n=1}^{N} r_k^n (\mathbf{x}_n - \boldsymbol{\mu}_k^{t+1})(\mathbf{x}_n - \boldsymbol{\mu}_k^{t+1})^T$$

$$\alpha_k^{t+1} = \frac{N_k}{N}$$

$$t = t + 1$$

4. Evaluate:

$$L(\mathcal{X} \mid \boldsymbol{\Theta}^t) = \ln(P(\mathcal{X} \mid \boldsymbol{\Theta}^t))$$
$$\rightarrow \text{ if } L(\mathcal{X} \mid \boldsymbol{\Theta}^t) \text{ converged: Terminate the iterations and provide } \boldsymbol{\Theta}_{\text{ML}} = \boldsymbol{\Theta}^t$$
$$\rightarrow \text{ if } L(\mathcal{X} \mid \boldsymbol{\Theta}^t) \text{ not converged} \Rightarrow \text{continue with E-Step}$$

Example – EM Algorithm for Mixture of Two Gaussian's: The EM algorithm is illustrated for a mixture of two Gaussian's applied to the Old Faithful data set in Figure 17 [Bishop, 2007]. In (a) two Gaussian's with a spherical covariance matrix are initialized. Figure (b) shows the result of the first E-step. The color of each data sample reflects the posterior probability of having been generated from either the red or the blue component. Thus, points that have a significant probability for belonging to either cluster appear purple. The situation after the first M step is shown in (c). Here the parameters of the components have been updated, i.e. the mean of the blue Gaussian has moved to the mean of the data set (i.e. center of gravity), weighted by the probabilities of each data point belonging to the blue cluster. Similarly, the covariance of the blue Gaussian has been updated. Analogously, the red Gaussian component is updated. Figure (d), (e), and (f) show the results of the EM algorithm after 2, 5, and 20 iterations, respectively. After 20 iterations the algorithm is close to convergence. A video clip of the EM algorithm for a 1-dimensional distribution can be found at https://www.youtube.com/watch?v=l1W3BvjJnmY.

4.2.1 Initialization

The parameters $\boldsymbol{\Theta}^0$ can be initialized in the following way:

1. α_k^0 is assumed to be uniformly distributed, i.e. it is set to $\alpha_k^0 = \frac{1}{K}$

2. $\boldsymbol{\Sigma}_k^0$ is set to the covariance matrix $\boldsymbol{\Sigma}$ of the data $\mathcal{X}$, i.e. $\boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T$, where $\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} x_n$.

3. For $\boldsymbol{\mu}_k^0$ we either select k arbitrary samples from $\mathcal{X}$ or the k-means algorithm (see Chapter 5) can be used.

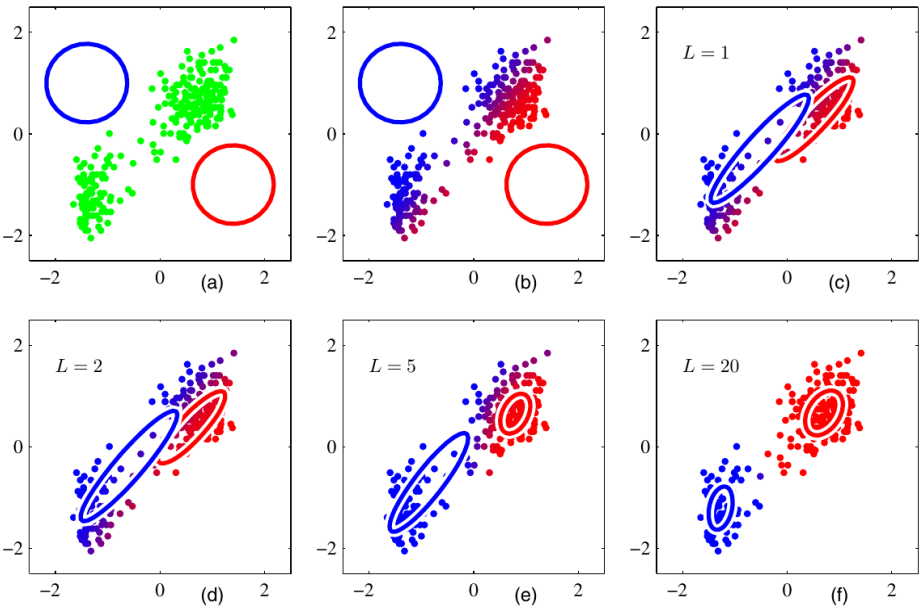


Figure 17: Illustration of the EM algorithm for GMMs over several EM iterations (from [Bishop, 2007]).

4.2.2 Properties of the EM algorithm

1. The log-likelihood $\ln P(\mathcal{X} \mid \mathbf{\Theta}^t)$ is usually monotonically increasing with each iteration. This is shown in Figure 18.

2. The EM algorithm finds a local optimal solution, i.e. a local maximum of the likelihood function. It cannot be guaranteed that the global optimum is found.

3. This means, that the solution depends on the initialization of $\mathbf{\Theta}^0$.

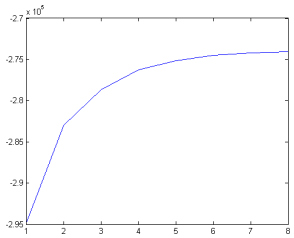


Figure 18: Log-likelihood $\ln P(\mathcal{X} \mid \mathbf{\Theta}^t)$ over the iterations of the EM algorithm.

5 The K-means algorithm

The aim of the K-means algorithm is to cluster the data, where the number of cluster is denoted as K. As an application of the K-means algorithm, we consider image segmentation [Bishop, 2007]. The goal of segmentation is to partition an image into regions each of which has a reasonably homogeneous visual appearance. Each pixel in an image is a sample $\mathbf{x}$ in a 3-dimensional space comprising the intensities of the red, blue, and green channels. The segmentation algorithm treats each pixel in the image as a separate data point. We can apply the K-means algorithm for clustering the pixels into K clusters. We illustrate the result of running K-means to convergence, for any particular value of K. Results of image

segmentation for various values of K are shown in Figure 19. We see that for a given value of K, the algorithm is representing the image using only K colors.

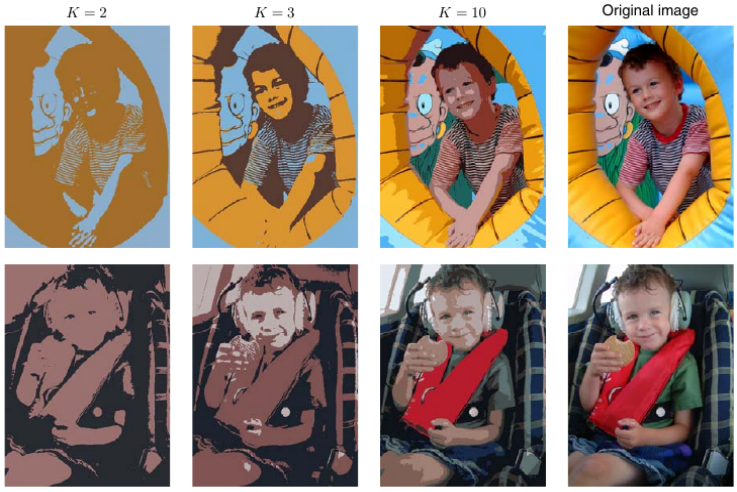


Figure 19: Two examples of the application of K-means for image segmentation. On the right-hand side the original images are shown. The remaining images show the segmentation using various values of K (from [Bishop, 2007]).

We can modify the EM-algorithm for GMMs in Section 4.2 to derive the K-means algorithm. Under the following assumptions we obtain the K-means algorithm from the EM algorithm:

1. The samples $\mathbf{x}_n$ are classified to component k; i.e. $k = \arg\max_{k'}[r_{k'}^n]$; Each sample is modeled by just *one* component.

2. $\alpha_k = P(k) = 1/K \qquad \forall k$; We assume that α_k is uniformly distributed and not modified over the iterations, i.e. α_k can be neglected; see Case 1 in Section 3.1.

3. $\mathbf{\Sigma}_k = \sigma^2\mathbf{I} \qquad \forall k$; We assume for all components the same spherical covariance matrix; see Case 1 in Section 3.1.

In the K-means algorithm the samples are classified to the most probable component, i.e. we modify the E-step and determine the class as follows.

$$\begin{aligned} k = \arg\max_{k'}[r_{k'}^n] &= \arg\max_{k'} \frac{\alpha_{k'}\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'}, \mathbf{\Sigma}_{k'})}{\sum_{k''=1}^{n} \alpha_{k''}\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k''}, \mathbf{\Sigma}_{k''})} \\ &= \arg\max_{k'}[\ln \alpha_{k'} + \ln \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'}, \mathbf{\Sigma}_{k'})]. \end{aligned}$$

Under assumption 2 and 3 (see Case 1 in Section 3.1) we obtain

$$k = \arg\max_{k'} \left[-(\mathbf{x}_n - \boldsymbol{\mu}_{k'})^T(\mathbf{x}_n - \boldsymbol{\mu}_{k'})\right] = \arg\min_{k'}(\mathbf{x}_n - \boldsymbol{\mu}_{k'})^T(\mathbf{x}_n - \boldsymbol{\mu}_{k'}). \tag{27}$$

This means that we classifier each sample $\mathbf{x}_n$ to the closest cluster center $\boldsymbol{\mu}_k$ based on the Euclidean distance.

5.1 K-means algorithm

Here, we show the K-means algorithm for clustering the data $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ into K cluster. We denote the set of all data samples classified to component k as $\mathbf{Y}_k$, i.e. $\mathbf{Y}_k = \{\mathbf{x}_n \mid k = \arg\min'_k[(\mathbf{x}_n - \boldsymbol{\mu}'_k)^T(\mathbf{x}_n - \boldsymbol{\mu}'_k)]\}$. The algorithm consists of the following steps.

1. Initialization: $t \ldots$ iteration counter

   Select K samples arbitrarily for each cluster center $\boldsymbol{\mu}_k$, i.e. $\boldsymbol{\Theta}^0 = \{\boldsymbol{\mu}_k^{t=0}\}_{k=1}^K$, $t = 0$.

2. Step 1: Classification of the samples to the clusters. This replaces the E-step.

$$\mathbf{Y}_k = \{\mathbf{x}_n \mid k = \arg\min_{k'} \left[(\mathbf{x}_n - \boldsymbol{\mu}_{k'}^t)^T(\mathbf{x}_n - \boldsymbol{\mu}_{k'}^t)\right]\} \qquad \forall k = 1, \ldots, K$$

3. Step 2: Update the parameters: Compute the cluster center (this corresponds to the center of gravity) based on the classification in $\mathbf{Y}_k$

$$\boldsymbol{\mu}_k^{t+1} = \frac{1}{\mid \mathbf{Y}_k \mid} \sum_{\mathbf{x}_n \in \mathbf{Y}_k} \mathbf{x}_n$$
$$t = t + 1$$

4. Evaluation of the cumulative distance

$$J^t = \sum_{k=1}^{K} \sum_{\mathbf{x}_n \in \mathbf{Y}_k} (\mathbf{x}_n - \boldsymbol{\mu}_k^t)^T(\mathbf{x}_n - \boldsymbol{\mu}_k^t)$$

   $\rightarrow$ If J^t is converged, i.e. $|J^t - J^{t-1}| < \epsilon$, then the optimal cluster centers are $\{\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_K^t\}$.

   $\rightarrow$ If J^t not converged, i.e. $|J^t - J^{t-1}| > \epsilon$, $\Rightarrow$ continue with Step 1

The K-means algorithm for $K = 2$ is shown in Figure 20. The samples (in green) and the initialization of the two clusters is shown in (a). Figure (b) shows the classification after step 1, each data sample is assigned either to the red cluster or to the blue cluster. This is equivalent to classifying the samples according to which side of the perpendicular bisector of the two cluster centers, shown by the magenta line, they lie on. In (c) parameters are updated, i.e. each cluster center is determined as the mean of the samples in $\mathbf{Y}_k$. Figure (d)–(i) show successive iterations until the convergence of the algorithm. Applications for the K-means algorithm are data compression, non-uniform data discretization, or the initialization of the EM algorithm for estimating GMMs.

5.2 Properties of K-means

1. K-means converges to a local minimum of the cumulative distance J^t.

2. In each iteration, the cumulative distance J^t gets smaller, i.e. $J^t \geq J^{t+1}$.

3. The result of the clustering depends on the initialization of $\boldsymbol{\Theta}^0$, i.e. usually, the global optimum is not found by the algorithm.

4. The decision boundary between the clusters is piece-wise linear.

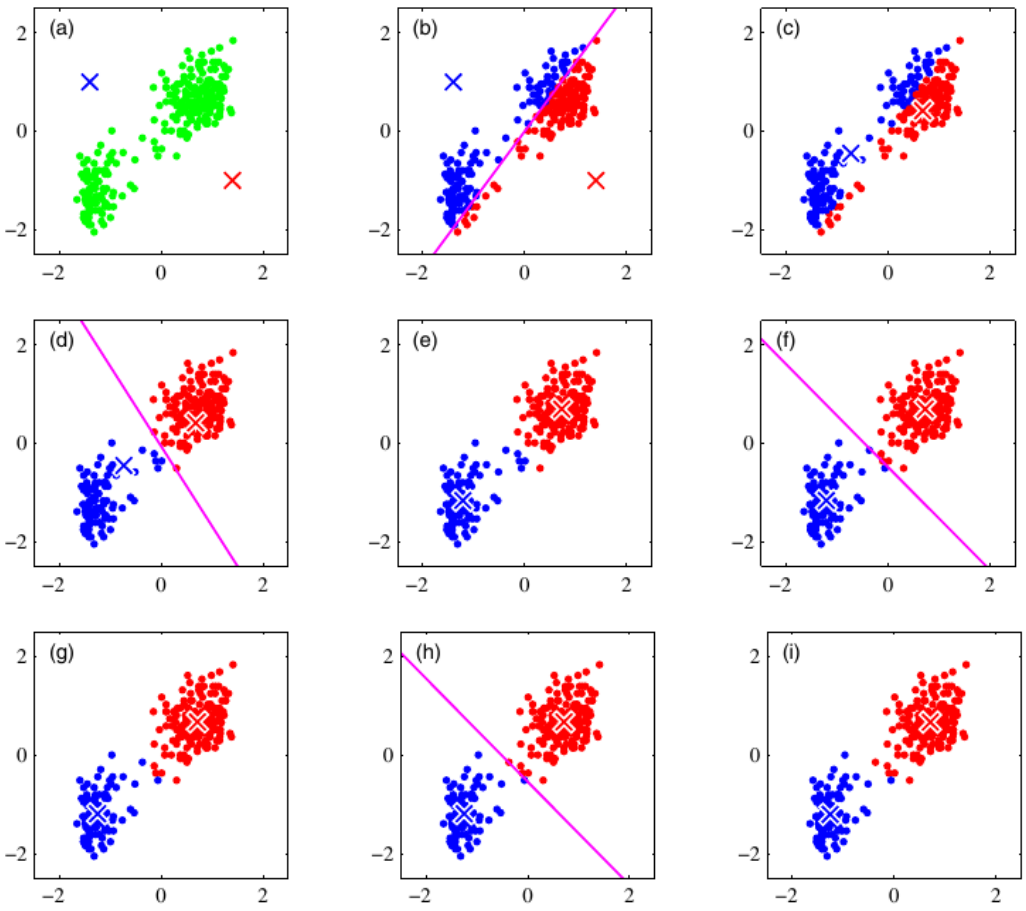


Figure 20: Illustration of the K-means algorithm over several iterations (from [Bishop, 2007]).

6 Markov Model (MM)

The Markov model is used to model sequential data, i.e. data where we aim to explicitly model the dependency among the samples. This means that the **iid** assumption is partially neglected.

6.1 Example: Language model for speech recognition

Let's introduce Q_n as state variable, where n denotes the time index (or index of the sequence elements).
We assume that Q_n is a discrete random variable, where $Q_n \in \mathcal{W}$ is a state of $\mathcal{W}$ with finite size.
In this example, $\mathcal{W}$ is the word dictionary and each state is one particular word.

An example for $\mathcal{W}$ is $\mathcal{W} = \begin{pmatrix} \text{ich} \\ \text{gehe} \\ \vdots \end{pmatrix}$, where $|\mathcal{W}|$ denotes the number of words in $\mathcal{W}$.

The aim of a language model is to assign a probability $P(Q_1, ..., Q_N)$ to every conceivable word string of length N. In particular, the likelihood of sentence with correct grammar should be larger as for sentences with grammatical errors.

This joint probability can be factorized by application of the chain-rule (product rule) as:
$P(Q_1, ..., Q_N) = P(Q_1)P(Q_2|Q_1)P(Q_3|Q_2, Q_1) \cdot \ldots \cdot P(Q_N|Q_{N-1}, \ldots, Q_1)$.
The number of parameters of the probability distributions for each of those terms increases with the number of conditioning variables. In particular, we have the following number of parameters:

$$
\begin{aligned}
\#P(Q_1) &= |\mathcal{W}| \\
\#P(Q_2|Q_1) &= |\mathcal{W}|^2 \\
\#P(Q_3|Q_2, Q_1) &= |\mathcal{W}|^3 \\
\#P(Q_N|Q_{N-1}, \ldots, Q_1) &= |\mathcal{W}|^N
\end{aligned}
$$

In particular, the number of probabilities in each term increases exponentially with the length N of the sentence. We would have $(1000)^{10}$ probabilities if we assume a dictionary of size $|\mathcal{W}| = 1000$ and a sentence of length $N = 10$. It is obvious that a reliable estimation and storage of these probabilities is unfeasible. The limitation of the context of words in each term reduces the complexity. Furthermore, this has the advantage that sequences of varying length can be modeled.
Therefore, we will make a simplifying assumption, called Markov assumption. For a sequence $\{Q_1, Q_2, ..., Q_n\}$:

$$
P(Q_n = q_n|Q_{n-1} = q_{n-1}, Q_{n-2} = q_{n-2}, ..., Q_1 = q_1) = P(Q_n = q_n|Q_{n-1} = q_{n-1}). \tag{28}
$$

This is called a *first-order Markov assumption*: we say that the probability of a certain observation at time n only depends on the observation q_{n-1} at time $n-1$. A second-order Markov assumption would assume that the probability of an observation at time n depends on q_{n-1} and q_{n-2}. In general, when people talk about Markov assumption, they usually mean the first-order Markov assumption. A model for which Eqn. 28 holds is a *(first-order) Markov model*.

This simplifying assumption is exploited in language modeling, i.e. the n-gram language model (LM).

- 1-gram LM: Here we make the assumption that there is no statistical dependency between the words, i.e. the individual words are iid:
  $P(Q_1, \ldots, Q_N) = \prod_{i=1}^{N} P(Q_i)$.
  The main problem of this model is that we obtain the same probability for all permutations of the words of the sentence e.g. "Ich gehe bald einkaufen", i.e. P(Ich gehe bald einkaufen) = P(einkaufen bald Ich gehe). Hence, the sentence with the better grammar does not get a larger probability assigned.

- 2-gram (bigram) LM: A bigram model decomposes as

  $$
  P\left(Q_1, \ldots, Q_N\right) = P(Q_1) \prod_{i=2}^{N} P\left(Q_i|Q_{i-1}\right). \tag{29}
  $$

  This means that we make the 1^{st} order Markov assumption. The probability distributions (= parameters) of this model are the prior probability $P(Q_1)$ for the very first word at the beginning of a sentence and the transition probability $P(Q_i|Q_{i-1})$. These parameters can be determined by maximum-likelihood estimation using a text corpus. Let $P(q_2|q_1)$ be the probability that word q_2 follows word q_1. We can estimate this probability from text by simply dividing the number of times we see the phrase "q_1, q_2" (i.e. the count $C(q_1, q_2)$) by the number of times we see the word "q_1" (i.e. $C(q_1)$) as

  $$
  P(q_2|q_1) = \frac{C(q_1, q_2)}{C(q_1)}, \tag{30}
  $$

  where $C(\cdot)$ denotes the number of occurrences of a particular word sequence. $P(q_2|q_1)$ is called a conditional bigram probability. Each distinct $P(q_2|q_1)$ is called a parameter.
  An example is:

  $$
  P(\text{I like snakes that are not poisonous}) \propto P(\text{I})P(\text{like}|\text{I})P(\text{snakes}|\text{like}) \cdots P(\text{poisonous}|\text{not}). \tag{31}
  $$

In other words, what is the chance that a sentence starts with the word "I"? If you did say "I", what is the chance that you would say the word "like" immediately after? And if you did say "like", is "snakes" a reasonable next word? And so on.

- 3-gram (trigram) LM: For 3-grams we assume that the joint probability factorizes as $P(Q_1, \ldots, Q_N) = P(Q_1)P(Q_2|Q_1)\prod_{i=3}^{N} P(Q_i|Q_{i-1}, Q_{i-2})$. This corresponds to a 2^{nd} order Markov model, where the context is extended to the two previous words.

  An example is:

$$P(\text{I like snakes that are not poisonous}) \propto P(\text{I})P(\text{like}|\text{I})P(\text{snakes}|\text{like,I}) \cdots P(\text{poisonous}|\text{not,are}). \tag{32}$$

6.2 Example: Modeling the weather

Let's talk about the weather. Let's say in Graz, there are three types of weather: sunny, rainy, and foggy. Let's assume for the moment that the weather lasts all day, i.e., it doesn't change from rainy to sunny in the middle of the day.
Weather *prediction* is about trying to guess what the weather will be like tomorrow based on the observations of the weather in the past (the *history*). Let's set up a *statistical model* for weather prediction: We collect statistics on what the weather Q_n is like today (on day n) depending on what the weather was like yesterday Q_{n-1}, the day before Q_{n-2}, and so forth. We want to find the following conditional probabilities

$$P(Q_n|Q_{n-1}, Q_{n-2}, ..., Q_1), \tag{33}$$

meaning, the probability of the unknown weather at day n, $Q_n \in \{$, , $\}$, depending on the (known) weather $Q_{n-1}, Q_{n-2}, \ldots$ of the past days.
Using the probability in Eqn. 33, we can make probabilistic predictions of the type of weather for tomorrow and the next days using the observations of the weather history. For example, if we knew that the weather for the past three days was $\{$, , $\}$ in chronological order, the probability that tomorrow would be is given by:

$$P(Q_4 = |Q_3 = , Q_2 = , Q_1 =). \tag{34}$$

This probability could be inferred from the relative frequency (the *statistics*) of past observations of weather sequences $\{$, , , $\}$.
Again, the larger n is, the more observations we must collect. Suppose that $n = 6$, then we have to collect statistics for $3^{(6-1)} = 243$ past histories. Therefore, we will make again the 1^{st}-order Markov assumption. For a sequence $\{Q_1, Q_2, ..., Q_n\}$:

$$P(Q_n|Q_{n-1}, Q_{n-2}, ..., Q_1) = P(Q_n|Q_{n-1}). \tag{35}$$

We can also express the joint probability of a certain sequence $\{Q_1, Q_2, \ldots, Q_n\}$ using the first-order Markov assumption:[2]

$$P(Q_1, ..., Q_n) = \prod_{i=1}^{n} P(Q_i|Q_{i-1}). \tag{36}$$

The Markov assumption has an effect on the number of histories that we have to find statistics for – we now only need $3 \cdot 3 = 9$ numbers ($P(Q_n|Q_{n-1})$ for every possible combination of $Q_n, Q_{n-1} \in \{$, , $\}$) to characterize the probabilities of all possible sequences. The Markov assumption may or may not be a valid assumption depending on the situation (in the case of weather, it's probably not valid), but it is often used to simplify modeling.
So let's arbitrarily pick some numbers for $P(Q_{\text{tomorrow}}|Q_{\text{today}})$, as given in Table 3 (note, that – whatever the weather is today – there *certainly is some kind of weather* tomorrow, so the probabilities in every *row* of table 3 sum up to one).

[2] One question that comes to mind is "What is Q_0?" In general, one can think of Q_0 as the *start word*, so $P(Q_1|Q_0)$ is the probability that Q_1 can start a sentence. We can also just multiply a *prior probability* of Q_1 with the product of $\prod_{i=2}^{n} P(Q_i|Q_{i-1})$, it's just a matter of definitions.

Today's weather	Tomorrow's weather		
	0.8	0.05	0.15
	0.2	0.6	0.2
	0.2	0.3	0.5

Table 3: Probabilities $P(Q_{n+1}|Q_n)$ of tomorrow's weather based on today's weather.

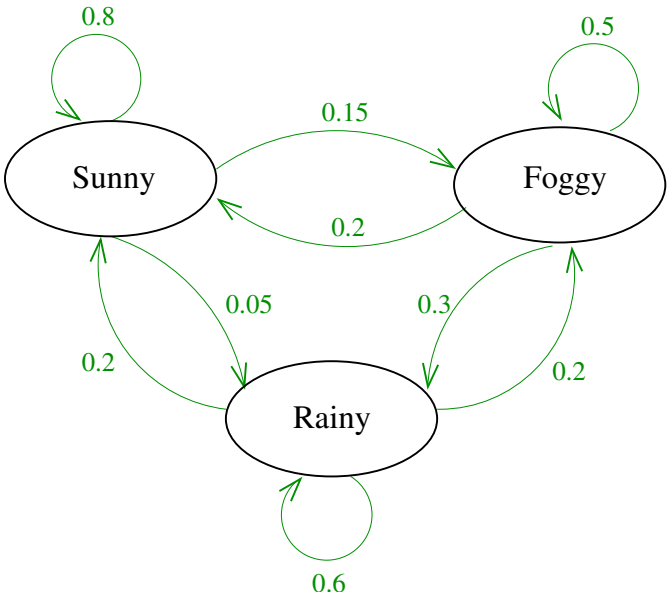


Figure 21: Markov model for the Graz weather with state transition probabilities according to Table 3.

For first-order Markov models, we can use these probabilities to draw a probabilistic finite state automaton. For the weather domain, we would have three states, $\mathcal{S} = \{$, , $\}$, and every day there would be a possibility $P(Q_n|Q_{n-1})$ of a transition to a (possibly different) state according to the probabilities in Table 3. Such an automaton would look like shown in Figure 21.

6.2.1 Examples

1. Given that today the weather is , what's the probability that tomorrow is and the day after is ?
   Using the Markov assumption and the probabilities in table 3, this translates into:

$$\begin{aligned} P(Q_2 = , Q_3 = |Q_1 =) &= P(Q_3 = |Q_2 = , Q_1 =) \cdot P(Q_2 = |Q_1 =) \\ &= P(Q_3 = |Q_2 =) \cdot P(Q_2 = |Q_1 =) \qquad \text{(Markov assumption)} \\ &= 0.05 \cdot 0.8 \\ &= 0.04 \end{aligned}$$

   You can also think about this as moving through the automaton (Figure 21), multiplying the probabilities along the path you go.

2. Assume, the weather yesterday was $Q_1 =$, and today it is $Q_2 =$, what is the probability that tomorrow it will be $Q_3 =$?

$$\begin{aligned} P(Q_3 = |Q_2 = , Q_1 =) &= P(Q_3 = |Q_2 =) \qquad \text{(Markov assumption)} \\ &= 0.2. \end{aligned}$$

3. Given that the weather today is $Q_1 =$, what is the probability that it will be two days from now: $Q_3 =$. (Hint: There are several ways to get from today to two days from now. You have to sum over these paths.)

6.3 Parameters of the Markov model

A Markov model is specified by the parameters $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \mathbf{A}\}$:

- The *set of states* $\mathcal{S} = \{1, 2, \ldots, N_s\}$, (corresponding to the three possible weather conditions above),
- The *prior probabilities* $\pi_i = P(Q_1 = i)$ are the probabilities of $i \in \mathcal{S}$ being the *first state* of a state sequence. We can collect these probabilities in a vector $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{N_S}\}$. (The prior probabilities are sometimes assumed equi-probable, i.e. $\pi_i = 1/N_s$.)
- The *transition probabilities* are the probabilities to go from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$: $a_{i,j} = P(Q_n = j | Q_{n-1} = i)$. They are collected in the matrix $\mathbf{A}$. If all elements in $\mathbf{A}$ are larger than zero we are able to switch the state to any other state at each time step. Such models are also called *ergodic* models.

7 Hidden Markov Model

So far we heard of the Markov assumption and Markov models. So, what is a *Hidden Markov Model*? Well, suppose you were locked in a room for several days, and you were asked about the weather outside. The only piece of evidence you have is whether the person who comes into the room bringing your daily meal is carrying an umbrella ( ) or not ( ).
Let's suppose the probabilities shown in Table 4: The probability that your caretaker carries an umbrella is 0.1 if the weather is sunny, 0.8 if it is actually raining, and 0.3 if it is foggy.

Weather	Probability of umbrella
Sunny	0.1
Rainy	0.8
Foggy	0.3

Table 4: Probability $P(X_n|Q_n)$ of carrying an umbrella ($x_n = \text{true}$) based on the weather Q_n on some day n.

The actual weather is *hidden*. Finding the probability of a certain weather $Q_n \in \{$, , $\}$ can only be based on the observation X_n, with $X_n =$, if your caretaker brought an umbrella on day n, and $x_i =$ if the caretaker did not bring an umbrella. As shown in Table 4, the observations X_n are statistically related to the state Q_n at time n. The observations $\mathbf{X}_n$ at each n can be discrete or continuous as well as scalar X_n or a set of values $\mathbf{X}_n$. The probability $P(\mathbf{X}_n|Q_n)$ is called observation or emission probability. For the observations we have the assumption that $\mathbf{X}_n$ only depends on the current state Q_n, i.e it is statistically independent of all other states and observations.
In particular, $P(\mathbf{X}_n|Q_1, \ldots, Q_n, \ldots, Q_N, \mathbf{X}_1, \ldots, \mathbf{X}_{n-1}, \mathbf{X}_{n+1}, \ldots, \mathbf{X}_N) = P(\mathbf{X}_n|Q_n)$.

7.1 Parameters of the HMM

The parameters can be summarized as: $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}\}$. In addition to the parameters of the Markov model (see Section 6.3), the HMM has the following parameters.

- The HMM additionally requires emission or observation probabilities. The observations/emissions can be discrete or continuous and scalar values or vectors of observation values. The observation probability distribution is denoted by $\mathbf{B}$.
  - Discrete observation: $\mathbf{x}_n \in \{\vartheta_1, \ldots, \vartheta_k\}$: $b_{Q_n=i, \mathbf{X}_n=\mathbf{x}_n} = b_{i,\mathbf{x}_n} = P(\mathbf{X}_n = \mathbf{x}_n | Q_n = i)$

– Continuous observation: $\mathbf{x}_n \in \mathbb{R}^d$: $b_{Q_n=i,\mathbf{x}_n} = b_{i,\mathbf{x}_n} = P(\mathbf{x}_n|Q_n = i)$; Continuous observations can be e.g. modeled by a Gaussian distribution or a Gaussian mixture model.

Furthermore, the HMM assumes the following statistical independence assumptions:

1. Markov Model (MM) of 1st order: The current states only depend on the previous state, i.e. $P(Q_n|Q_{n-1}, \ldots, Q_1) = P(Q_n|Q_{n-1})$.

2. Observation $\mathbf{X}_n$ just depends on Q_n, i.e. $P(\mathbf{X}_n|Q_1, \ldots, Q_N, \mathbf{X}_1, \ldots, \mathbf{X}_{n-1}, \mathbf{X}_{n+1}, \ldots, \mathbf{X}_N) = P(\mathbf{X}_n|Q_n)$.

In the HMM, we have the following conditions for all parameters $\boldsymbol{\Theta}$:

$$
\begin{aligned}
0 \leq \pi_i \leq 1 \quad &\text{and} \quad \sum_{i=1}^{N_s} \pi_i = 1 \\
0 \leq a_{ij} \leq 1 \quad &\text{and} \quad \sum_{j=1}^{N_s} a_{ij} = 1 \quad \forall i \\
0 \leq b_{i,\mathbf{x}_n} \leq 1 \quad &\text{and} \quad \mathbf{x}_n \text{ is continuous: } \int_{-\infty}^{\infty} b_{i,\mathbf{x}_n} \mathrm{d}\mathbf{x}_n = 1 \qquad \forall i \\
&\qquad\quad \mathbf{x}_n \text{ is discrete: } \sum_{\mathbf{x}_n \in \{\vartheta_1 \ldots \vartheta_k\}} b_{i,\mathbf{x}_n} = 1 \qquad \forall i
\end{aligned}
$$

In the sequel, we require the following symbols. A state sequence is denoted as $\mathbf{Q} = \{Q_1, \ldots, Q_N\}$, an observation sequence as $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$, their realization as $\mathbf{Q} = \{Q_1 = q_1, \ldots, Q_N = q_N\} = \{q_1, \ldots, q_N\}$ and $\mathbf{X} = \{\mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_N = \mathbf{x}_N\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, respectively.

7.2 Three basic problems

There are 3 fundamental problems of interest:

1. Evaluation problem (or classification problem): Let's assume that we have a model $\boldsymbol{\Theta}_t$ and an observation sequence $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. The aim is to determine the likelihood $P(\mathbf{X}|\boldsymbol{\Theta}_t)$ of the observation sequence. This likelihood can be used for classifying sequences with the Bayes classifier, i.e. the Bayes classifier is able to classify sequences $\mathbf{X}$ with *varying* length:

$$
P(t|\mathbf{X}) = \frac{P(\mathbf{X}|t)P(t)}{\sum_{t'=1}^{T} P(\mathbf{X}|t')P(t')}.
$$

We can use the HMM for the likelihood function $P(\mathbf{X}|t)$, i.e. $P(\mathbf{X}|t) = P(\mathbf{X}|\boldsymbol{\Theta}_t)$, where $\boldsymbol{\Theta}_t$ is the HMM model for class t. Again, we select the class t^* with the largest posterior probability $P(t|\mathbf{X})$:

$$
t^* = \arg\max_t \left[P(\mathbf{X}|t)P(t)\right] = \arg\max_t \left[(\mathbf{X}|\boldsymbol{\Theta}_t)P(t)\right].
$$

For calculating $P(\mathbf{X}|\boldsymbol{\Theta}_t)$ the Forward/Backward algorithm can be used. This algorithm is discussed in more detail in Section 7.2.1.

2. Decoding problem: Let's assume that we have the model $\boldsymbol{\Theta}$ and the observation sequence $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
   The aim is to determine the state sequence $\mathbf{Q}^* = \arg\max_{\mathbf{Q}} [P(\mathbf{Q}|\mathbf{X}, \boldsymbol{\Theta})]$ which best *explains* the observation sequence $\mathbf{X}$. Usually, the Viterbi algorithm (details are provided in Section 7.2.2) determines the optimal state sequence $\mathbf{Q}^*$.

3. Learning- or estimation problem: Let's assume that we have R observation sequences: $\mathbf{X}^{1:R} = \{\mathbf{X}^1, \ldots, \mathbf{X}^R\}$. The aim is to determine the parameters of the model, i.e.

$$\boldsymbol{\Theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} \left[P(\mathbf{X}^{1:R}|\boldsymbol{\Theta}) \right] = \arg\max \left[\prod_{r=1}^{R} P(\mathbf{X}^r|\boldsymbol{\Theta}) \right], \tag{37}$$

   where $P(\mathbf{X}^r|\boldsymbol{\Theta})$ denotes the likelihood obtained by the evaluation problem. The ML Solution $\boldsymbol{\Theta}_{\mathrm{ML}}$ is determined by an EM algorithm (similar as in Section 4.2).

In the following, we discuss approaches to solve each problem.

7.2.1 Evaluation problem / Classification problem

We have given an observation sequence $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and the model parameters $\boldsymbol{\Theta}$. The aim is to determine the probability $P(\mathbf{X}|\boldsymbol{\Theta})$. The probability for the state sequence $\mathbf{Q} = \{q_1, \ldots, q_N\}$ ($\hat{=}$ Markov Model 1$^{\mathrm{st}}$ order) given the model is:

$$\begin{aligned} P(\mathbf{Q}|\boldsymbol{\Theta}) &= P(q_1, \ldots, q_N|\boldsymbol{\Theta}) = \\ &= P(q_1) \prod_{n=2}^{N} P(q_n|q_{n-1}) = \pi_{q_1} \prod_{n=2}^{N} a_{q_{n-1},q_n}. \end{aligned}$$

The probability of the observation sequence $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ given the state sequence $\mathbf{Q} = \{q_1, \ldots, q_N\}$ and the model $\boldsymbol{\Theta}$ is:

$$P(\mathbf{X}|\mathbf{Q}, \boldsymbol{\Theta}) = P(\mathbf{x}_1, \ldots, \mathbf{x}_N|q_1, \ldots, q_N, \boldsymbol{\Theta}) = \prod_{n=1}^{N} P(\mathbf{x}_n|q_n) = \prod_{n=1}^{N} b_{q_n,\mathbf{x}_n}.$$

The joint probability is

$$\begin{aligned} P(\mathbf{X}, \mathbf{Q}|\boldsymbol{\Theta}) &= P(\mathbf{X}|\mathbf{Q}, \boldsymbol{\Theta}) P(\mathbf{Q}|\boldsymbol{\Theta}) \\ &= \prod_{n=1}^{N} P(\mathbf{x}_n|q_n) P(q_1) \prod_{n=2}^{N} P(q_n|q_{n-1}) \\ &= P(q_1) P(\mathbf{x}_1|q_1) \prod_{n=2}^{N} P(q_n|q_{n-1}) P(\mathbf{x}_n|q_n) \\ &= \pi_{q_1} b_{q_1,\mathbf{x}_1} \prod_{n=2}^{N} a_{q_{n-1},q_n} b_{q_n,\mathbf{x}_n}. \end{aligned}$$

The probability $P(\mathbf{X}|\boldsymbol{\Theta})$ can be determined by marginalization (sum-rule) of $P(\mathbf{X}, \mathbf{Q}|\boldsymbol{\Theta})$, i.e. $P(\mathbf{X}|\boldsymbol{\Theta}) = \sum_{\mathbf{Q} \in \mathcal{Q}} P(\mathbf{X}, \mathbf{Q}|\boldsymbol{\Theta})$, where $\mathcal{Q}$ denotes the set of all possible state sequences of the HMM. The trellis diagram visualizes the unrolling of the HMM over time (see Figure 22).

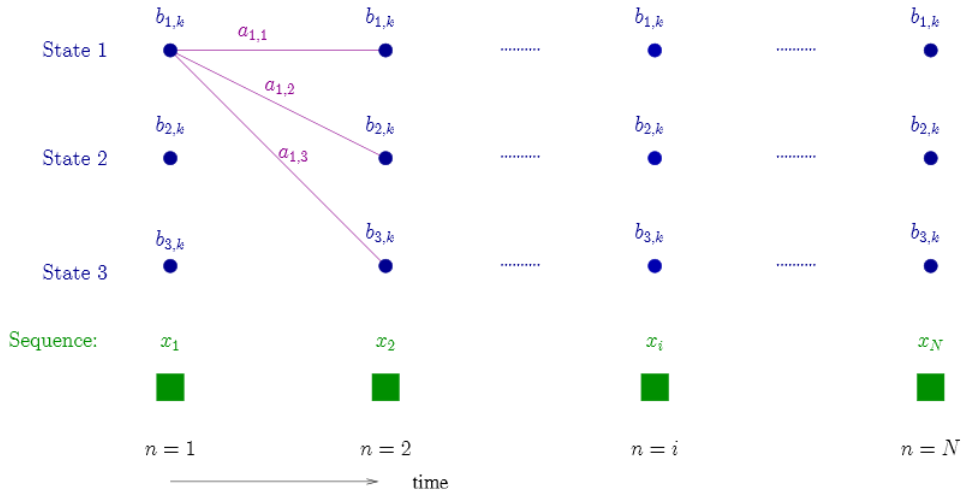


Figure 22: Trellis Diagram [Rank and Pernkopf, 2004].

Each column in the trellis shows the possible states of the weather at a certain time n. Each state in one column is connected to each state in the adjacent columns by the transition likelihood given by the the elements $a_{i,j}$ of the transition matrix $\mathbf{A}$ (shown for state 1 at time 1 in Figure 22). At the bottom is the observation sequence $X = \{X_1, \ldots, X_N\}$. $b_{i,k}$ is the likelihood of the observation $X_n = k$ in state $Q_n = i$ at time n.

Example: Suppose you do not know how the weather was when your were locked in. The following three days the caretaker always comes without an umbrella. Calculate the likelihood for the weather on these three days to have been $\{Q_1 = ☀, Q_2 = ☁, Q_3 = ☀\}$. As we do not know how the weather is on the first day, we assume a *prior probability* for sun on day one as $P(Q_1 = ☀) = 1/3$. The joint likelihood is

$$\begin{aligned} L(Q_1 = ☀, Q_2 = ☁, Q_3 = ☀, X_1 = 🕶, X_2 = 🕶, X_3 = 🕶) = \\ P(X_1 = 🕶|Q_1 = ☀) \cdot P(X_2 = 🕶|Q_2 = ☁) \cdot P(X_3 = 🕶|Q_3 = ☀) \cdot \\ P(Q_1 = ☀) \cdot P(Q_2 = ☁|Q_1 = ☀) \cdot P(Q_3 = ☀|Q_2 = ☁) = \\ 0.9 \cdot 0.7 \cdot 0.9 \cdot 1/3 \cdot 0.15 \cdot 0.2 = 0.0057. \end{aligned} \tag{38}$$

Figure 23 shows the trellis diagram for this example. The likelihood of the state sequence given the observation sequence can be found by simply following the path in the trellis diagram, multiplying the observation and transition likelihoods, i.e.:

$$L = \pi_{☀} \cdot b_{☀,🕶} \cdot a_{☀,🌧} \cdot b_{🌧,🕶} \cdot a_{🌧,☀} \cdot b_{☀,🕶} \tag{39}$$

$$= 1/3 \cdot 0.9 \cdot 0.15 \cdot 0.7 \cdot 0.2 \cdot 0.9. \tag{40}$$

The number of different state sequences is $|\mathcal{Q}| = (N_s)^N$. This means that a naive marginalization of $P(\mathbf{X}, \mathbf{Q}|\mathbf{\Theta})$ has a computational complexity of $O(2N(N_s)^N)$, i.e. it is exponential in N. This is not practical.
The forward/backward algorithm exploits the factorization property of $P(\mathbf{X}, \mathbf{Q}|\Theta)$ which reduces the computational complexity to $O(2(N_s)^2 N)$ which is linear in N. The main reason for this massive reduction is the utilization of intermediate results on partial paths (i.e. exploiting the distributive law $a(b + c) = ab + ac$). In the following, the computation of the *forward probability* with the help of the recursive forward algorithm is presented. The marginalization of the forward probability over the states at time N provides the probability $P(\mathbf{X}|\mathbf{\Theta})$.

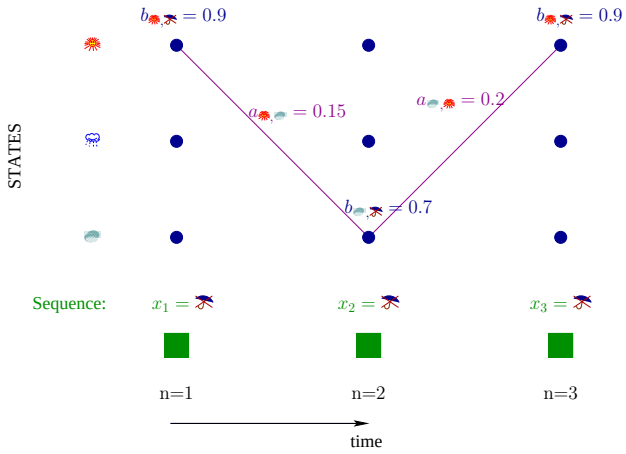


Figure 23: Trellis diagram.

The forward probability is defined as:

$$\alpha_n(j) = P(\mathbf{x}_1, \ldots, \mathbf{x}_n, Q_n = j|\mathbf{\Theta}) = \sum_{Q_1,\ldots,Q_{n-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_n, Q_1, \ldots, Q_{n-1}, Q_n = j|\Theta). \tag{41}$$

It can be computed in a recursive way, shown in the following algorithm:

Forward algorithm:

- Initialization

$$\alpha_1(j) = \pi_j b_{j,\mathbf{x}_1} \quad \forall j = 1, \ldots, N_s$$

- Recursion

$$\alpha_n(j) = \left(\sum_{i=1}^{N_s} \alpha_{n-1}(i) a_{i,j} \right) b_{j,\mathbf{x}_n} \quad \forall n = 2, \ldots, N; \quad \forall j = 1 \ldots N_s$$

- Computation of probability $P(\mathbf{X}|\mathbf{\Theta})$:

$$P(\mathbf{X}|\mathbf{\Theta}) = P(\mathbf{x}_1, \ldots, \mathbf{x}_N|\mathbf{\Theta}) = \sum_{j=1}^{N_s} \alpha_N(j)$$

Backward algorithm: In a similar fashion, the probability $P(\mathbf{X}|\mathbf{\Theta})$ can be also computed by the backward algorithm. The backward probability is defined as: $\beta_n(j) = P(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|Q_n = j, \mathbf{\Theta})$.

- Initialization

$$\beta_N(i) = 1 \quad \forall i = 1, \ldots, N_s$$

- Recursion

$$\beta_n(i) = \sum_{j=1}^{N_s} a_{i,j} b_{j,\mathbf{x}_{n+1}} \beta_{n+1}(j) \quad \forall n = N-1, \ldots, 1; \quad \forall i = 1, \ldots, N_s$$

- Computation of probability $P(\mathbf{X}|\mathbf{\Theta})$:

$$P(\mathbf{X}|\mathbf{\Theta}) = P(\mathbf{x}_1, \ldots, \mathbf{x}_N|\mathbf{\Theta}) = \sum_{j=1}^{N_s} \pi_j b_{j,\mathbf{x}_1} \beta_1(j)$$

7.2.2 Decoding problem

The decoding problem is solved by the Viterbi algorithm. Given is the model $\mathbf{\Theta}$ and the observation sequence $\mathbf{X}$. The aim is to determine the state sequence $\mathbf{Q}^* = \{q_1^*, \ldots, q_N^*\}$ which best explains the observation sequence $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, i.e.

$$Q^* = \arg\max_{\mathbf{Q}\in\mathcal{Q}} P(\mathbf{Q}|\mathbf{X}, \mathbf{\Theta}) = \arg\max_{\mathbf{Q}\in\mathcal{Q}} \frac{P(\mathbf{X}, \mathbf{Q}|\mathbf{\Theta})}{P(\mathbf{X}|\mathbf{\Theta})} = \arg\max_{\mathbf{Q}\in\mathcal{Q}} P(\mathbf{X}, \mathbf{Q}|\mathbf{\Theta}).$$

The term $P(\mathbf{X}|\mathbf{\Theta})$ is independent of $\mathbf{Q}$. This means that it scales $P(\mathbf{Q}|\mathbf{X}, \mathbf{\Theta})$, i.e. it does not change the argument of the maximum operation (i.e. can be neglected). We define the maximal attainable probability along a path $\mathbf{Q}^*$ as $P^*(\mathbf{X}|\mathbf{\Theta})$, i.e.

$$P^*(\mathbf{X}|\mathbf{\Theta}) \stackrel{!}{=} P(\mathbf{Q}^*, \mathbf{X}|\mathbf{\Theta}) = \underbrace{\max_{\mathbf{Q}\in\mathcal{Q}} P(\mathbf{X}, \mathbf{Q}|\mathbf{\Theta})}_{\text{this is recursively computed in the Viterbi algorithm}}$$

Let us introduce $\delta_n(i)$ for the *highest* likelihood of a single path among all the paths ending in state i at time n observing the partial observation sequence $\mathbf{x}_1, \ldots, \mathbf{x}_n$, i.e.

$$\delta_n(i) = \max_{Q_1,\ldots,Q_{n-1}} P(Q_1, \ldots, Q_{n-1}, Q_n = i, \mathbf{x}_1, \ldots, \mathbf{x}_n|\Theta). \tag{42}$$

The calculation of $\delta_n(i)$ is performed in a recursive manner, i.e.

$$\delta_n(j) = \max\left[\delta_{n-1}(i)a_{ij}\right] b_{j,\mathbf{x}_n}, \tag{43}$$

where $\delta_1(i) = \pi_1 b_{i,\mathbf{x}_1}$. In the case of $n = N$, we obtain the maximal attainable probability $P^*(\mathbf{X}|\mathbf{\Theta})$:

$$\begin{aligned}\delta_N(i) &= \max_{Q_1,\ldots,Q_{N-1}} P(Q_1, \ldots, Q_N = i, \mathbf{x}_1, \ldots, \mathbf{x}_N|\Theta)\\ P^*(\mathbf{X}|\mathbf{\Theta}) &= \max_{1\le j\le N_s} \delta_N(j).\end{aligned}$$

For the extraction of the most likely state sequence $\mathbf{Q}^*$ an additional variable is required, which memorizes the argument of the maximum operator. This is necessary for facilitating the extraction of the best path $\mathbf{Q}^*$ at time $n = N$. For that the variable $\psi_n(i)$

$$\psi_n(i) \stackrel{!}{=} \arg\max_{Q_1,\ldots,Q_{n-1}} \left[P(Q_1, \ldots, Q_{n-1}, Q_n = i, \mathbf{x}_1, \ldots, \mathbf{x}_n|\mathbf{\Theta})\right]$$

is used. It allows to keep track of the *best path* ending in state i at time n The best path $\mathbf{Q}^*$ is extracted by *backtracking*

$$q_n^* = \psi_{n+1}(q_{n+1}^*) \qquad n = N-1, \ldots, 1.$$

Viterbi algorithm This section summarizes the Viterbi algorithm.

- Initialization

$$\begin{aligned}\delta_1(i) &= \pi_i b_{i,\mathbf{x}_1} \quad \forall i = 1, \ldots, N_s\\ \psi_1(i) &= 0 \quad \forall i = 1, \ldots, N_s\end{aligned}$$

- Recursion

$$\begin{aligned}\delta_n(j) &= \max_{1\le i\le N_s} \left[\delta_{n-1}(i)a_{ij}\right] b_{j,\mathbf{x}_n} \quad \forall j = 1, \ldots, N_s \quad \forall n = 2, \ldots, N\\ \psi_n(j) &= \arg\max_{1\le i\le N_s} \left[\delta_{n-1}(i)a_{ij}\right] \quad \forall j = 1, \ldots, N_s \quad \forall n = 2, \ldots, N\end{aligned}$$

Find the path that leads to a maximum likelihood considering the best likelihood at the previous step and the transitions from it; then multiply by the current observation likelihood of the current state.

- Termination

$$P^*(\mathbf{X}|\mathbf{\Theta}) = \max_{1\le j\le N_s} \delta_N(j)$$
$$q_N^* = \arg \max_{1\le j\le N_s} \delta_N(j)$$

  Find the best likelihood when the end of the observation sequence $n = N$ is reached.

- Backtracking The best path $\mathbf{Q}^*$ is at time step N in q_N^*. Backtracking extracts $q_{N-1}^*, \ldots, q_1^*$ according to

$$q_n^* = \psi_{n+1}(q_{n+1}^*), \qquad n = N-1, N-2, \ldots, 1$$

Example: Viterbi algorithm We would like to exploit the weather example again. Consider that we do not know the weather when we have been locked in. In the first 3 days the umbrella observations are: {no umbrella, umbrella, umbrella} ({☂, ☂, ☂}). The aim is to find the most probable weather-sequence using the Viterbi algorithm (assuming that the 3 weather situations are equi-probable on day 1).

1. Initialization
   $n = 1$:

$$\delta_1(☀) = \pi_{☀} \cdot b_{☀,☂} = 1/3 \cdot 0.9 = 0.3$$
$$\psi_1(☀) = 0$$
$$\delta_1(☔) = \pi_{☔} \cdot b_{☔,☂} = 1/3 \cdot 0.2 = 0.0667$$
$$\psi_1(☔) = 0$$
$$\delta_1(☁) = \pi_{☁} \cdot b_{☁,☂} = 1/3 \cdot 0.7 = 0.233$$
$$\psi_1(☁) = 0$$

2. Recursion
   $n = 2$:
   We calculate the likelihood of getting to state '☀' from all possible 3 predecessor states, and choose the most likely one to go on with:

$$\delta_2(☀) = \max(\delta_1(☀) \cdot a_{☀,☀}, \delta_1(☔) \cdot a_{☔,☀}, \delta_1(☁) \cdot a_{☁,☀}) \cdot b_{☀,☂}$$
$$= \max(0.3 \cdot 0.8, 0.0667 \cdot 0.2, 0.233 \cdot 0.2) \cdot 0.1 = 0.024$$
$$\psi_2(☀) = ☀$$

   The likelihood is stored in δ, the most likely predecessor in ψ. See figure 24.

   The same procedure is executed with states ☔and ☁:

$$\delta_2(☔) = \max(\delta_1(☀) \cdot a_{☀,☔}, \delta_1(☔) \cdot a_{☔,☔}, \delta_1(☁) \cdot a_{☁,☔}) \cdot b_{☔,☂}$$
$$= \max(0.3 \cdot 0.05, 0.0667 \cdot 0.6, 0.233 \cdot 0.3) \cdot 0.8 = 0.056$$
$$\psi_2(☔) = ☁$$
$$\delta_2(☁) = \max(\delta_1(☀) \cdot a_{☀,☁}, \delta_1(☔) \cdot a_{☔,☁}, \delta_1(☁) \cdot a_{☁,☁}) \cdot b_{☁,☂}$$
$$= \max(0.3 \cdot 0.15, 0.0667 \cdot 0.2, 0.233 \cdot 0.5) \cdot 0.3 = 0.0350$$
$$\psi_2(☁) = ☁$$

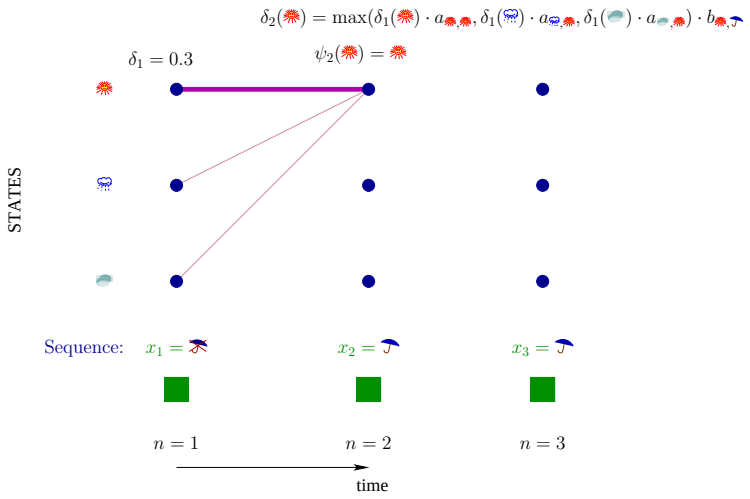


Figure 24: Viterbi algorithm to find the most likely weather sequence. Finding the most likely path to state 'sunny' at $n = 2$.

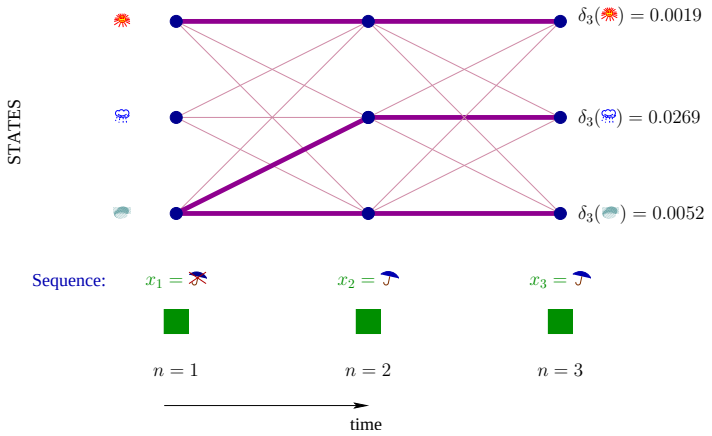


Figure 25: Viterbi algorithm to find most likely weather sequence at $n = 3$.

$n = 3$:

$$
\begin{aligned}
\delta_3(\text{☀}) &= \max(\delta_2(\text{☀}) \cdot a_{\text{☀},\text{☀}}, \delta_2(\text{🌧}) \cdot a_{\text{🌧},\text{☀}}, \delta_2(\text{☁}) \cdot a_{\text{☁},\text{☀}}) \cdot b_{\text{☀},\text{☂}} \\
&= \max(0.024 \cdot 0.8, 0.056 \cdot 0.2, 0.035 \cdot 0.2) \cdot 0.1 = 0.0019 \\
\psi_3(\text{☀}) &= \text{☀} \\
\delta_3(\text{🌧}) &= \max(\delta_2(\text{☀}) \cdot a_{\text{☀},\text{🌧}}, \delta_2(\text{🌧}) \cdot a_{\text{🌧},\text{🌧}}, \delta_2(\text{☁}) \cdot a_{\text{☁},\text{🌧}}) \cdot b_{\text{🌧},\text{☂}} \\
&= \max(0.024 \cdot 0.05, 0.056 \cdot 0.6, 0.035 \cdot 0.3) \cdot 0.8 = 0.0269 \\
\psi_3(\text{🌧}) &= \text{🌧} \\
\delta_3(\text{☁}) &= \max(\delta_2(\text{☀}) \cdot a_{\text{☀},\text{☁}}, \delta_2(\text{🌧}) \cdot a_{\text{🌧},\text{☁}}, \delta_2(\text{☁}) \cdot a_{\text{☁},\text{☁}}) \cdot b_{\text{☁},\text{☂}} \\
&= \max(0.0024 \cdot 0.15, 0.056 \cdot 0.2, 0.035 \cdot 0.5) \cdot 0.3 = 0.0052 \\
\psi_3(\text{🌧}) &= \text{☁}
\end{aligned}
$$

Finally, we get one most likely path ending in each state of the model. See figure 25.

3. Termination
   The globally most likely path is determined, starting by looking for the last state of the most likely

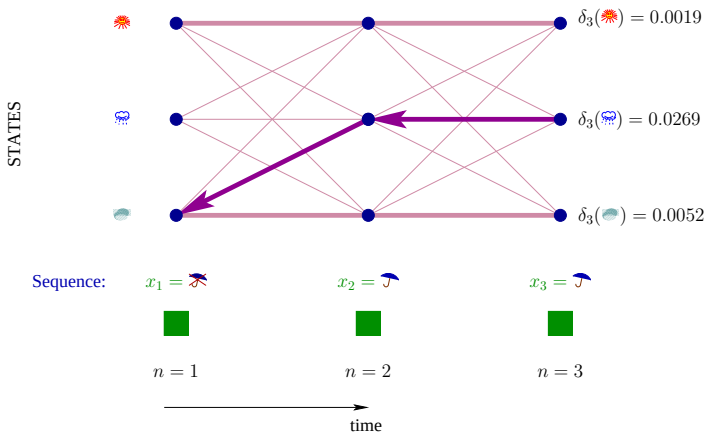


Figure 26: Viterbi algorithm to find most likely weather sequence. Backtracking.

sequence.

$$P^*(X|\mathbf{\Theta}) = \max(\delta_3(i)) = \delta_3(☔) = 0.0269$$
$$q_3^* = \arg\max(\delta_3(i)) = ☔$$

4. Backtracking
   The best sequence of states can be read from the ψ vectors. See figure 26.
   $n = N - 1 = 2$:
   $$q_2^* = \psi_3(q_3^*) = \psi_3(☔) = ☔$$
   $n = N - 1 = 1$:
   $$q_1^* = \psi_2(q_2^*) = \psi_2(☔) = ☁$$
   Thus the most likely weather sequence is: $Q^* = \{q_1^*, q_2^*, q_3^*\} = \{☁, ☔, ☔\}$.

8 Linear Transformations

8.1 Projection of x

Let's assume data samples $\{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^D$, where D is the dimension in the original space. We aim to project sample $\mathbf{x}_n$ into a lower dimension (e.g., to a 1-dimensional space $M = 1$ as shown in Figure 27) using the D-dimensional vector $\mathbf{u}_1$ which provides the direction for the projection.

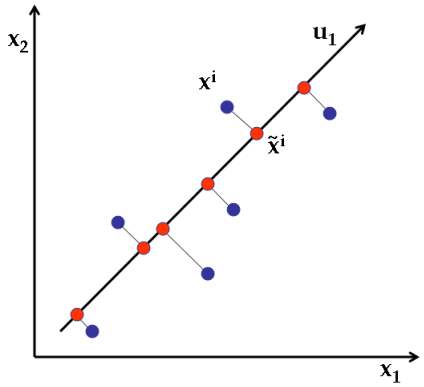


Figure 27: Dimensionality reduction; $\mathbf{x}_n$ is projected onto $\mathbf{u}_1$ [Learning, 2013].

Sample $\mathbf{x}_n$ is projected according to

$$y_n = \frac{\mathbf{u}_1^T \mathbf{x}_n}{\| \mathbf{u}_1 \|}, \tag{44}$$

where $y_n \in \mathbb{R}^M$, $M = 1$. Assuming a unit length vector $\mathbf{u}_1$, i.e. $\| \mathbf{u}_1 \| = \sqrt{\mathbf{u}_1^T \mathbf{u}_1} = 1$, the linear projection simplifies to $y_n = \mathbf{u}_1^T \mathbf{x}_n$. This projection into the one-dimensional space (i.e. $M = 1$) can be easily extended to $1 < M \leq D$ according to:

$$\mathbf{y}_n = \mathbf{U}^T \mathbf{x}_n \tag{45}$$

$$= \begin{pmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_M^T \end{pmatrix} \mathbf{x}_n, \tag{46}$$

where $\mathbf{y}_n \in \mathbb{R}^M$ and $\| \mathbf{u}_m \| = 1 \quad \forall m$.

8.1.1 Statistical properties of projected/transformed data ($M = 1$)

The mean of the projected/transformed data μ_y can be determined as:

$$\mu_y = \frac{1}{N} \sum_{n=1}^{N} \mathbf{u}_1^T \mathbf{x}_n$$

$$= \mathbf{u}_1^T \boldsymbol{\mu}_x,$$

where $\boldsymbol{\mu}_x = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$ is the mean of data $\mathbf{x}_n$ in the original space (denoted by the subindex x). The variance of the transformed data σ_y^2 can be determined as:

$$\begin{aligned}\sigma_y^2 &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\boldsymbol{\mu}_x)^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T(\mathbf{x}_n - \boldsymbol{\mu}_x))^2 \\ &= \mathbf{u}_1^T \underbrace{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_x)(\mathbf{x}_n - \boldsymbol{\mu}_x)^T}_{\boldsymbol{\Sigma}_x}\mathbf{u}_1 \\ &= \mathbf{u}_1^T\boldsymbol{\Sigma}_x\mathbf{u}_1, \end{aligned} \tag{47}$$

where $\boldsymbol{\Sigma}_x = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_x)(\mathbf{x}_n - \boldsymbol{\mu}_x)^T$ is the covariance matrix of the data $\mathbf{x}_n \in \mathbb{R}^D$.

8.2 Principal Component Analysis (PCA) for M=1

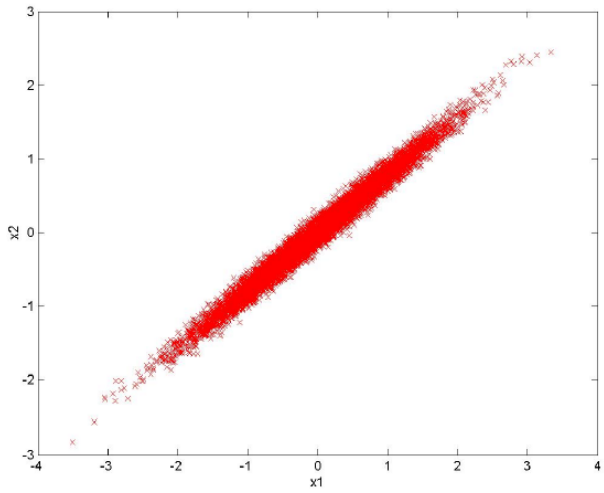


Figure 28: Data in $\mathbb{R}^2$ [Wohlmayr, 2013].

PCA is basically a linear transformation of data to decorrelate the features. The fundamental idea of PCA is that the useful *information* in the data is represented in the variance of the data samples. This means that we are looking for a projection vector $\mathbf{u}_1$ that accounts for as much of the variability in the data as possible after projection. Hence the vector $\mathbf{u}_1$ is determined with the objective of maximizing the variance of the projected data σ_y^2, i.e.

$$\begin{aligned}\mathbf{u}_1 &= \arg\max_{\mathbf{u}_1}[\sigma_y^2] \\ &= \arg\max_{\mathbf{u}_1}[\mathbf{u}_1^T\boldsymbol{\Sigma}_x\mathbf{u}_1].\end{aligned}$$

We need to constrain $\mathbf{u}_1$ as

$$\mathbf{u}_1^T \mathbf{u}_1 = 1$$

to avoid that $\| \mathbf{u}_1 \| \rightarrow \infty$. This leads to an optimization problem $\arg\max_{\mathbf{u}_1} [\mathbf{u}_1^T \mathbf{\Sigma}_x \mathbf{u}_1]$ with the constraint $\mathbf{u}_1^T \mathbf{u}_1 = 1$ which can be solved by the Lagrangian approach. The solution can be obtained by Lagrange multipliers [Bishop, 2007]. The Lagrange function is composed by

$$\mathcal{L}(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^T \mathbf{\Sigma}_x \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1).$$

The derivative of $\mathcal{L}(\mathbf{u}_1, \lambda_1)$ for $\mathbf{u}_1$ is obtained as

$$\frac{\partial \mathcal{L}(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = 2\mathbf{\Sigma}_x \mathbf{u}_1 - \lambda_1 2\mathbf{u}_1 \stackrel{!}{=} 0 \tag{48}$$

and set to zero. Equation (48) can be reformulated as

$$\mathbf{\Sigma}_x \mathbf{u}_1 = \lambda_1 \mathbf{u}_1 \tag{49}$$

which is an Eigenvalue/Eigenvector problem. This means that $\mathbf{u}_1$ is the Eigenvector of the covariance matrix $\mathbf{\Sigma}_x$ with the Eigenvalue λ_1. By multiplying both sides of (49) with $\mathbf{u}_1^T$, we obtain

$$\lambda_1 = \mathbf{u}_1^T \mathbf{\Sigma}_x \mathbf{u}_1 = \sigma_y^2. \tag{50}$$

This means that the Eigenvalue λ_1 corresponds to the variance of the transformed (projected) data σ_y^2. Due to this fact, we obtain the largest variance by projecting onto the Eigenvector $\mathbf{u}_1$ with the largest Eigenvalue λ_1. Usually $\mathbf{u}_1$ is called principal component.

8.3 PCA for $M > 1$

PCA is mostly used to visualize data in 3D or to reduce the dimensionality in data. We can project the data onto M dimensions as already shown in (45), i.e.

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_M \end{pmatrix}}_{\mathbf{y}_n} = \underbrace{\begin{pmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_M^T \end{pmatrix}}_{\mathbf{U}^T} \cdot \underbrace{\begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix}}_{\mathbf{x}_n}. \tag{51}$$

Therefore, several vectors $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]$ for projection are necessary. All these vectors are obtained from the Eigenvector/Eigenvalue decomposition of the covariance matrix $\mathbf{\Sigma}_x$ i.e. we solve the Eigenvalue/Eigenvector problem $\mathbf{\Sigma}_x \mathbf{U} = \mathbf{U}\mathbf{L}$, where the diagonal matrix $\mathbf{L}$ consists of the Eigenvalues of the decomposition:

$$\mathbf{L} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_M \end{pmatrix}. \tag{52}$$

As we already know, the Eigenvalues correspond to the variances of the projected data in the respective dimensions (see (50)). Hence, for projecting $\mathbf{x}_n$ into the M-dimensional space $\mathbf{y}_n$, we select the M Eigenvectors of $\mathbf{\Sigma}_x$ corresponding to the M largest Eigenvalues. These *principal components* form the basis $\mathbf{U}$ for projection in (51). The Eigenvectors are orthogonal to each other $\mathbf{u}_i^T \mathbf{u}_j = 0$, $i \neq j$, $\forall i, j$ and $\mathbf{u}_i^T \mathbf{u}_i = 1$, $\forall i$. Furthermore, the covariance matrix of the transformed data $\mathbf{y}_n$ is a diagonal matrix, i.e. $\mathbf{\Sigma}_y = \mathbf{L}$. This means that the dimensions in the transformed data are uncorrelated, i.e. y_i and y_j, $i \neq j$, $\forall i, j$ are uncorrelated.

Example: PCA for Handwritten Digits The dimensionality reduction of the data simultaneously serves as a way of lossy compression. PCA provides a way to compress the data while preserving most of its variance (and, according to the underlying assumption, most of the information). To understand why this is often a sensible thing to do, let us consider a data set of hand-written digits, represented by 28×28 pixel gray-level images. Each image consequently lies in a $28 \times 28 = 784$-dimensional space. Each digit, however, has only a significantly lower intrinsic dimension, corresponding to e.g., translation, rotation, scaling, and other more complex deformations.

We, then, consider all images of one specific digit, e.g., the digit three. We then compute the covariance matrix and perform the Eigenvalue/Eigenvector decomposition; Figure 29 illustrates the rapid decay of the eigenvalues.

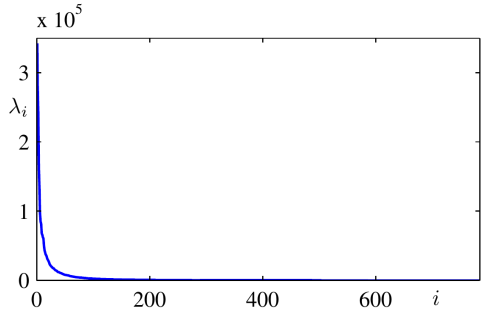


Figure 29: Decay of the eigenvalues for the digit three [Bishop, 2007].

Figure 30 finally illustrates one specific instance of the digits and the result of the compression that results from the projection onto a lower-dimensional subspace spanned by the M eigenvectors with the largest eigenvalues.

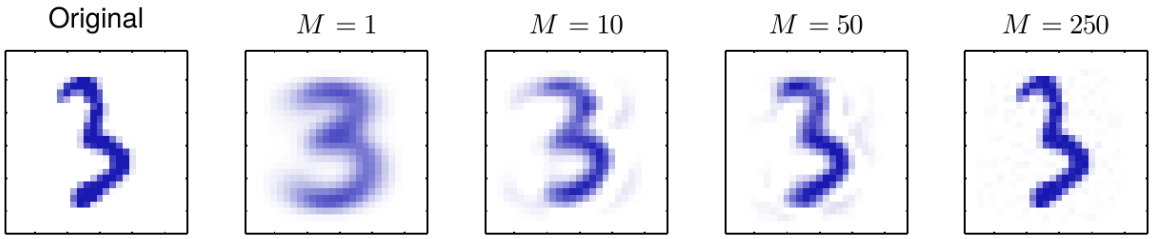


Figure 30: Original image of the considered image and the compressed images by projection onto the M principal components [Bishop, 2007].

Whitening: We have used PCA to reduce the dimension of the data. There is a closely related preprocessing step called *whitening*. The goal of whitening is that (i) the features are not correlated with each other (as already shown in PCA), (ii) the features all have the same variance (spherical covariance matrix), and (iii) that the data is mean centered, i.e. has zero mean $\boldsymbol{\mu}_y$.
Therefore, we slightly modify the projection of PCA in (51) by projecting the data as follows:

$$\mathbf{y}_n = \mathbf{L}^{-\frac{1}{2}}\mathbf{U}^T(\mathbf{x}_n - \boldsymbol{\mu}_x).$$

This leads to the desired properties of the transformed data, i.e. they have a mean of zero and a unity matrix $\mathbf{I}$ as covariance matrix, i.e.

- $\boldsymbol{\mu}_y = \frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n = 0$

- $\boldsymbol{\Sigma}_y = \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n \mathbf{y}_n^T$

$$= \mathbf{L}^{-\frac{1}{2}}\mathbf{U}^T \underbrace{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mathbf{m}_x)(\mathbf{x}_n - \mathbf{m}_x)^T}_{\boldsymbol{\Sigma}_x} \mathbf{U}\mathbf{L}^{-\frac{1}{2}}$$

$$= \mathbf{L}^{-\frac{1}{2}} \underbrace{\mathbf{U}^T\boldsymbol{\Sigma}_x\mathbf{U}}_{\mathbf{L}} \mathbf{L}^{-\frac{1}{2}} = \mathbf{I}$$

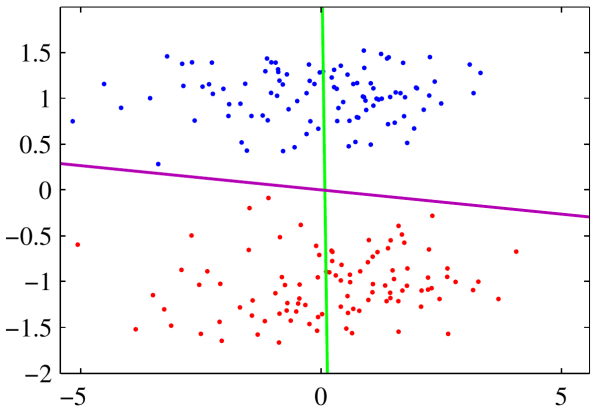


Figure 31: Comparison of PCA (projection onto magenta line) with LDA (projection onto green line) for dimensionality reduction [Bishop, 2007].

8.4 Linear Discriminant Analysis (LDA)

Projecting the data into a space of lower dimension M ($M < D$) leads to a loss of information in the data. It could happen that well-separated classes in the original feature space with dimension D are strongly overlapping in the low-dimensional space. This is shown in Figure 31. Here the data in two dimensions, belonging to two classes shown in red and blue, is projected onto a single dimension. As PCA is unsupervized, it depends only on the values $\mathbf{x}_n$ and no class labels are considered for determining the basis for projection. Hence PCA chooses the direction of maximum variance shown by the magenta line. This leads to a strong class overlap. LDA takes the class labels into account. This leads to a projection onto the green line which leads to much better class separation.

Let's introduce LDA in more detail. LDA uses the class labels (supervized technique) to maintain the information of the classes in the transformation of the data. This means that we may select $\mathbf{u}$ in such a way that the distance between the classes, i.e. the respective mean values of each class, is maximized.
In the sequel, we assume 2 classes. Let us divide the labeled data $\mathcal{X}$ into set T_1 and T_2 for class label 1 and 2, respectively, i.e.

$$\mathcal{X} = \{\underbrace{\mathbf{x}_1, \ldots, \mathbf{x}_{N_1}}_{T_1}; \underbrace{\mathbf{x}_{N_1+1}, \ldots, \mathbf{x}_N}_{T_2}\}.$$

The mean vectors $\boldsymbol{\mu}_x$ for both classes are:

$$\boldsymbol{\mu}_{x,1} = \frac{1}{|T_1|}\sum_{\mathbf{x}_n \in T_1} \mathbf{x}_n \qquad \text{and} \qquad \boldsymbol{\mu}_{x,2} = \frac{1}{|T_2|}\sum_{\mathbf{x}_n \in T_2} \mathbf{x}_n.$$

After linear projection (transformation) the means of the projected data are (see also Section 8.1.1):

$$\mu_{y,1} = \mathbf{u}^T \boldsymbol{\mu}_{x,1} \qquad \text{and} \qquad \mu_{y,2} = \mathbf{u}^T \boldsymbol{\mu}_{x,2}.$$

Case 1: For improving the class separability after projection, the simplest solution is to select $\mathbf{u}$ such that the distance between the class means of the projected data $(\mu_{y,2} - \mu_{y,1})^2$ is maximal. This can be expressed mathematically as:

$$\begin{aligned}\mathbf{u} &= \arg\max_{\mathbf{u}} (\mu_{y,2} - \mu_{y,1})^2 \\ &= \arg\max_{\mathbf{u}} \mathbf{u}^T \left(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}\right) \left(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}\right)^T \mathbf{u} \\ &\text{subject to: } \mathbf{u}^T \mathbf{u} = 1\end{aligned}$$

This constraint optimization problem can be solved by the Lagrangian approach [Bishop, 2007]. The Lagrange function is composed as

$$\mathcal{L}(\mathbf{u}, \lambda) = \mathbf{u}^T \left(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}\right) \left(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}\right)^T \mathbf{u} + \lambda(1 - \mathbf{u}^T \mathbf{u}).$$

In the next step, we set the derivative of $\mathcal{L}(\mathbf{u}, \lambda)$ to zero, i.e.

$$\frac{\partial \mathcal{L}(\mathbf{u}, \lambda)}{\partial \mathbf{u}} = 2 \left(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}\right) \underbrace{\left(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}\right)^T \mathbf{u}}_{\text{scalar}} - \lambda 2\mathbf{u} \stackrel{!}{=} 0 \tag{53}$$

and we obtain:

$$\mathbf{u} \propto (\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}).$$

This means that $\mathbf{u}$ has the direction of the line joining $\boldsymbol{\mu}_{x,2}$ and $\boldsymbol{\mu}_{x,1}$. An example of this projection is shown in the left plot of Figure 32. This means that there is still a significant overlap of the classes after the transformation/projection. Due to this fact, the result is not optimal when considering only the means of the classes.

Case II: Fisher criterion In this case, the LDA additionally considers the variances of the data for each class $k \in \{1, 2\}$ after the projection:

$$\sigma_{y,k}^2 = \sum_{n \in T_k} (y_n - \mu_{y,k})^2 = \mathbf{u}^T \boldsymbol{\Sigma}_k \mathbf{u}.$$

This allows us to define the so-called *within-class covariance* of data $\mathcal{X}$. The within-class-covariance specifies the covariance of the data within a particular class, i.e.

$$\sigma_y^2 = \sigma_{y,1}^2 + \sigma_{y,2}^2 = \mathbf{u}^T \boldsymbol{\Sigma}_1 \mathbf{u} + \mathbf{u}^T \boldsymbol{\Sigma}_2 \mathbf{u} = \mathbf{u}^T \boldsymbol{\Sigma}_w \mathbf{u},$$

where $\boldsymbol{\Sigma}_w$ denotes the within-class covariance matrix. The within-class covariance matrix $\boldsymbol{\Sigma}_w$ can be determined as:

$$\boldsymbol{\Sigma}_w = \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 = \sum_{\mathbf{x}_n \in T_1} (\mathbf{x}_n - \boldsymbol{\mu}_{x,1})(\mathbf{x}_n - \boldsymbol{\mu}_{x,1})^T + \sum_{x_n \in T_2} (\mathbf{x}_n - \boldsymbol{\mu}_{x,2})(\mathbf{x}_n - \boldsymbol{\mu}_{x,2})^T.$$

Furthermore, we can defined the between-class covariance matrix $\mathbf{\Sigma}_b$. The between-class covariance matrix represents the separation of the projected class means (see Case I), i.e.

$$(\mu_{y,2} - \mu_{y,1})^2 = \left[\mathbf{u}^T(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})\right]^2 = \mathbf{u}^T(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})^T\mathbf{u} = \mathbf{u}^T\mathbf{\Sigma}_b\mathbf{u},$$

where

$$\mathbf{\Sigma}_b = (\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})^T.$$

In the next step, we aim to maximize the ratio of the between-class covariance and within-class covariance matrix, i.e. the distance between the projected class means should be as large as possible (maximizing between-class covariance) while the within-class covariance of the projected data should be as small as possible (minimizing within-class covariance). This ratio is well-known as Fisher criterion:

$$J(\mathbf{u}) = \frac{(\mu_{y,2} - \mu_{y,1})^2}{\sigma_{y,1}^2 + \sigma_{y,2}^2} = \frac{\mathbf{u}^T\mathbf{\Sigma}_b\mathbf{u}}{\mathbf{u}^T\mathbf{\Sigma}_w\mathbf{u}}. \tag{54}$$

Hence, we are interested in

$$\mathbf{u} = \arg\max_{\mathbf{u}} J(\mathbf{u})$$

For maximizing $J(\mathbf{u})$ with respect to $\mathbf{u}$ we derive $J(\mathbf{u})$ in (54) for $\mathbf{u}$ and set the derivative to zero. This leads to the following equation:

$$(\mathbf{u}^T\mathbf{\Sigma}_b\mathbf{u})\mathbf{\Sigma}_w\mathbf{u} = (\mathbf{u}^T\mathbf{\Sigma}_w\mathbf{u})\mathbf{S}_b\mathbf{u}.$$

The terms $(\mathbf{u}^T\mathbf{\Sigma}_b\mathbf{u})$ and $(\mathbf{u}^T\mathbf{\Sigma}_w\mathbf{u})$ are scalar values and can be expressed by λ. Hence, we obtain a generalized Eigenvalue/Eigenvector problem

$$\lambda\mathbf{\Sigma}_w\mathbf{u} = \mathbf{\Sigma}_b\mathbf{u}. \tag{55}$$

By multiplying both sides of the equation with $\mathbf{\Sigma}_w^{-1}$ and noting that $\mathbf{\Sigma}_b\mathbf{u}$ points in the direction of $(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})$, i.e. $(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})^T\mathbf{u} \propto (\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1})$ we obtain the following solution:

$$\mathbf{u} \propto \mathbf{\Sigma}_w^{-1}(\boldsymbol{\mu}_{x,2} - \boldsymbol{\mu}_{x,1}).$$

This is also known as *Fisher's linear discriminant analysis*. The projection using the Fisher criterion is shown in the right plot of Figure 32. This means that by considering the covariances we are able to significantly reduce the class overlap after the transformation/projection. In general, LDA can be extended to multiple classes and high-dimensional projections.

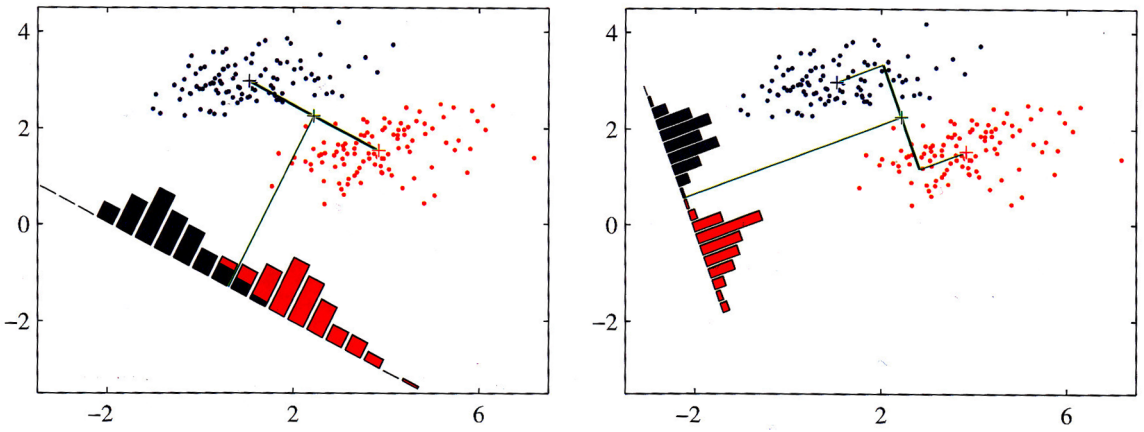


Figure 32: Dimensionality reduction by LDA; left plot: Case I; right plot: Case II - Fisher criterion [Bishop, 2007].

List of Figures

References

[Bishop, 2007] Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing 2011 edition.

[Duda et al., 2001] Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification*. Pattern Classification and Scene Analysis: Pattern Classification. Wiley.

[Köhler, 2005] Köhler, B.-U. (2005). *Konzepte der statistischen Signalverarbeitung*. Springer, Berlin [u.a.].

[Learning, 2013] Learning, C. M. (2013). Dimensionality reduction and principal component analysis. `https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.PCA`. Letzter Aufruf am: 10.06.2013.

[Pernkopf, 2013] Pernkopf, F. (2013). Graphical models. `https://www.spsc.tugraz.at/sites/default/files/CI_EW_GMs.pdf`. Letzter Aufruf am: 10.06.2013.

[Pernkopf et al., 2013] Pernkopf, F., Peharz, R., and Tschiatschek, S. (2013). Introduction to probabilistic graphical models. `https://www.spsc.tugraz.at/sites/default/files/PGM.pdf`. Letzter Aufruf am: 10.06.2013.

[Rank and Pernkopf, 2004] Rank, E. and Pernkopf, F. (2004). Hidden markov models. `https://www.spsc.tugraz.at/sites/default/files/Specomm04_3.pdf`. Letzter Aufruf am: 10.06.2013.

[Wikipedia, 2013] Wikipedia (2013). Mixture distribution. `http://en.wikipedia.org/wiki/Mixture_distribution`. Letzter Aufruf am: 06.06.2013.

[Wohlmayr, 2013] Wohlmayr, M. (2013). Blind source separation: Eine einführung. `http://www.spsc.tugraz.at/system/files/bss.pdf`. Letzter Aufruf am: 10.06.2013.