TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

CARINTHIAN TECH RESEARCH

Masterarbeit

zum Thema

Statistical Classification of Multispectral Imaging Data

ausgeführt am
Institut für Statistik und Wahrscheinlichkeitstheorie
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser

und Firmenbetreuer
Dipl.-Ing. Dr.techn. Raimund Leitner

durch
Andrea Appe, BSc.
Robert-Musil-Straße 18
9500 Villach

Villach, am 9. Februar 2012

Abstract

Multispectral data which have been collected by a hand-held device are classified into two groups by employing different statistical classification methods. The data used for the analysis are pixel images at nine wavelenghts of six objects which have been reshaped to matrices: the columns are the wavelengths used for data acquisition, the rows are the number of observations resulting from the dimension of the images. The performance of the following supervised classification methods is evaluated: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), robust LDA, robust QDA and Support Vector Machines (SVM) with linear and radial basis kernels. Misclassification rates are used to evaluate the models and Receiver Operator Characteristics help to visualise their performance. The results show that robust LDA works best for this kind of data, which contain outliers. Majority voting is applied to analyse the performance on parts of the images and to assign exactly one label to the whole image instead of each pixel. Data preprocessing and the statistical analysis are done with the software Matlab and R, respectively.

Acknowledgements

Studying is a long and winding road with many ups and downs - I want to thank my parents for their full financial and moral support during the last years.

Special thanks to Prof. Peter Filzmoser, who was an excellent supervisor for this work. Due to his consent to discuss urgent matters per email or phone, it was possible for me to write my thesis at a research center and to collect experiences when working with "real" measurement data instead of artificial datasets we often used during lectures. Each talk with him about the task I had was inspiring and brought new ideas to my mind. His motivating manner made it easy to formulate goals and reach them.

I also want to thank the whole team at the CTR AG for six instructive months full of new experiences. Special thanks to my supervisor Dr. Raimund Leitner, who always was open for my questions and a valuable tip concerning working and writing.

Contents

Preface

Data acquisition and its statistical analysis have been performed during a 6-month employment at the CTR AG, which is is the largest non-university reasearch institute in Carinthia. The main research field of the CTR AG are intelligent sensors, for more information see `www.ctr.at`.
Spectral Imaging, which is a completely new topic for a statistics student, and a challenging problem made work interesting and instructive every day. It was an entirely new experience to work with datasets which are generated on site, which was sometimes difficult as the recording device was further developed during the analysis. To choose and check the suitability of a method was often time consuming, and many results and plots cannot be possibly included in this work as space is limited. Problems due to the high dimensionality of the data futher complicated the data evaluation.
As the author is not allowed to publish details about the treatment, the formulation of the results in Chapter 3 seem a bit colourless, but that was the price to write a work on the pulse of research.

This thesis has been written using the software package LaTeX, where a good introduction is provided by Oetiker et al. (2011). The main literature used for writing Chapter 2 are the books Hastie et al. (2009) and Venables and Ripley (2002). The graphs have all been created by the author herself using the graphic package Ti*k*Z, see Tantau (2007). For the statistical analysis and the plots, the commercial mathematical software Matlab, see Appendix A.1, and the free software R, see Appendix A.2 and R Development Core Team (2011), have been used. For the graphs presented in this thesis which have been made with R, the book Wickham (2009) is a helpful reading.

Chapter 1

Introduction

> The field of Statistics is constantly challenged by the problems that science and industry brings to its door.
>
> Hastie et al. (2009)

Wherever data is collected, methods are needed to analyse them. Thereby, the way from data aquisiton to prediction results is often long and includes detours. Image data analysis strongly depends on the system that acquires the data. Classical RGB images only use visible light, but often the electromagentic spectrum is divided in more than the three bands red, green and blue in order to get more information about an object.
Spectral imaging uses different wavelengths to penetrate tissue in varying depths. While hyperspectral imaging uses a series of close wavelengths which produce a rather "continuous" image, *multispectral imaging* uses less wavelengths and the result can be described as "discrete". The aim of this thesis is to find a method for classifying multispectral images of objects from two classes, which will be denoted as "untreated" and "treated". As the project in the course of which the data anlysis has been performed is still running, the objects from which the data comes and the treatment which they underwent cannot be further described. Classification should work for a hand-held device, which means the budget for data recording is limited. This causes a reduced quality of the camera and limits the amount of wavelenghts used for data acquisition. In order to enable the direct use of the device by future customers, the computation time of the classification method should be short.

Chapter 2 provides the statistical theory about the following supervised classification methods: Linear discriminant Analysis (LDA), robust LDA, Fisher's linear discriminant, Quadratic Discriminant Analysis (QDA), robust QDA and Support

Vector Machines (SVM). Additionally, misclassification rates and Receiver Operator Characteristics (ROC) are presented as methods for interpreting the classification results. A short introduction to Cross Validation (CV) completes the second chapter.
Although a statistician is mainly interested in analysing data, in the case of spectral imaging the data acquisition is of some importance which should not be ignored. For this work it is important to have some idea how the data has been collected and how preprocessing works. Therefore the first two parts of *Chapter 3* include a short description of the data acquisition and the preprocessing methods used for the analysis. In the third part of this chapter the results of the methods described in Chapter 2 are visualised and discussed. Additionally, the optimal method for our requirements is selected and further evaluated using two approaches of majority voting. The fourth part of this chapter includes some prospects for further analyses.
Chapter 4 contains the conclusion of the statistical analysis.

Chapter 2

Statistical Theory of Selected Supervised Learning Techniques

For humans and animals, identifying objects is part of everyday live: we identify another person by its voice or its appearance, a blind person will often identify objects by touching them, a dog identifies by smelling and a bat can navigate with the help of ultrasound waves. These examples all use biological senses, which distinguishes us from machines, as they use algorithms for identification. In statistics, the identification of groups in data is one application of what is known as *pattern recognition*.
In order to discriminate between different populations, we assign a label to each observation of a data set. *Statistical Learning* is one of the modern techniques used to analyse often big amounts of data and to implement these assignments correctly. In general, income data (observed or *independent* variables, also called *features*) is used to predict outcome data (unobserved or *dependent* variables). The outcome variables can either be *quantitative*, e.g. expected age or weight, or *qualitative* (also called categorical or discrete), e.g. healthy/ill. There are several ways to learn:

- **Supervised learning**: We have a so-called "training" data set of objects (e.g. patients) where income and outcome values are known. This data is used to develop a prediction model and new, unknown income data (called "test" data) should then lead to an acceptable outcome prediction based on the model. The performance of the model is measured by certain evaluation criteria.
  Examples for supervised learning methods are *Regression* for quantitative outcome and *Classification* for categorical outcome.

- **Unsupervised learning**: The training data does not contain outcome data - only the unlabeled income data is used to detect patterns and structures

in the data. As we do not have labeled training data we cannot evaluate potential models.
Examples for unsupervised learning are *Clustering* and techniques for dimensionality reduction like *Principal Component Analysis* (PCA) and *Singular Value Decomposition* (SVD).

- **Reinforcement learning**: Learning by feedback: learn optimal behaviour, thus how to map situations to actions, in order to achieve maximal reward. The learner has to find out which actions are best by trying them out. Often not only immediate, but also delayed or subsequent reward is included in this trial-and-error learning technique.
  Contrary to supervised and unsupervised learning, reinforcement learning is defined by characterising a learning problem instead of learning methods. Any method which is suitable to solving the problem is called a reinforcement learning method.

For better understanding, Figure 2.1 shows a rough outline of some of the techniques used in Pattern Recognition. As computing facilities and methods have multiplied in the past, it only shows a very limited choice of methods available.

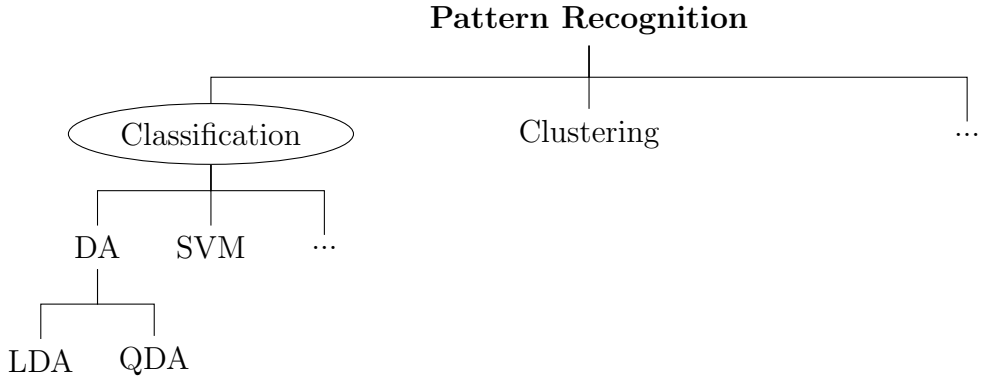


Figure 2.1: A schematic representation of the structure of pattern recognition. DA stands for Discriminant Analysis, LDA for Linear Discriminant Analysis, QDA for Quadratic Discriminant Analysis and SVM for Support Vector Machine.

In this thesis we focus on supervised learning, namely the classification problem, and have a closer look on methods like *Discriminant Analysis* (DA) and *Support Vector Machines* (SVM).

Classification

The aim of statistical classification is to assign new data to predefined classes correctly. Methods to define these classes are for example Discriminant Analysis or Support Vector Machines. Obervations within a class should have as much common characteristics as possible, whereas observations belonging to different classes should have few similarities.
The sample space is divided into K regions by assigning a class label to each training observation, and new data is classified according to their location in the sample space. The groups, in the following denoted by $\mathcal{G}$, are usually represented either by text, e.g. $\mathcal{G} = \{healthy, ill, dead\}$ or by integers, e.g. $\mathcal{G} = \{1, 2, 3\}$. In the case of two classes the groups are often represented by a binary digit like $\{0, 1\}$ or $\{-1, 1\}$. Data is represented by n pairs $(\mathbf{x}_i, g_i)$ with $\boldsymbol{x}_i \in \mathbb{R}^p$ and $g_i \in \mathcal{G}$ for $i = 1, \ldots, n$.
The borders of the regions can be linear or nonlinear. One often-used method of classification is discriminant analysis with the two special cases *Linear Discriminat Analysis* (LDA) and *Quadratic Discriminant Analysis* (QDA).

2.1 Discriminant Analysis

The objective of Discriminant Analysis is to specify one or more different but similar object groups through observed variables (features). This is done by finding those variables which represent differences between the groups and give them high weights, whereas features which do not help to separate the groups are downweighted.
The first step is to determine the differences of objects known to belong to exactly one of $K \geq 2$ similar groups graphically or algebraically in order to find a *discriminant function* which separates the groups by dividing the sample space into K disjoint regions. In the simplest case, the discriminant function is linear, which means that the groups can be graphically separated by a hyperplane (e.g. a straight line in the 2-dimensional case and a plane in the 3-dimensional case). A more complex form of partitioning is done with a quadratic discriminant function.

General assumptions

Let $\boldsymbol{X}$ be a p-dimensional random vector and $\boldsymbol{x}$ its realisation. In general, a predictor $G(\cdot)$ with values $G(\boldsymbol{x}) \in \mathcal{G}$ for new observations $\boldsymbol{x}$ is wanted (let $|\mathcal{G}| < \infty$, i.e. the set of possible values of $G(\cdot)$ is finite). We could establish *discriminant functions* $\delta_k(\boldsymbol{x})$, $k \in K$, and classify new objects $\boldsymbol{x}$ to the group k with the largest value of $\delta_k(\boldsymbol{x})$. Another approach is to compute *class posteriors* $\mathbb{P}(G = k|\boldsymbol{X} = \boldsymbol{x})$

for optimal classification. If either $\delta_k(\boldsymbol{x})$ or $\mathbb{P}(G = k|\boldsymbol{X} = \boldsymbol{x})$ are linear in $\boldsymbol{x}$, then the decision boundaries will be linear. This assumption can be softened: even when a monotone transformation of $\delta_k(\boldsymbol{x})$ or $\mathbb{P}(G = k|\boldsymbol{X} = \boldsymbol{x})$ is linear in $\boldsymbol{x}$ we get linear decision boundaries (e.g. logit transformation).
Let us develop the theory of discriminant analysis for two groups (case $K = 2$) in general (the distinction between linear and quadratic discriminant analysis comes later).

2.1.1 The two-class case: K=2

Assume we have two groups g_1 and g_2 and p random variables $\boldsymbol{X} = (X_1, \ldots, X_p)^\top$ which are used for the classification to one of the two groups. The entity of all objects of the first class are called $\boldsymbol{x}$-population of g_1, and those of the second class are called $\boldsymbol{x}$-population of g_2. The two populations can then be characterised by their probability distributions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$. Sample data is represented by a data matrix $\mathbf{X} \in \mathbb{R}^{(n\times p)}$ with

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix} \tag{2.1}$$

where n is the number of observations and p is the dimension in later sections of this thesis. Note that the unobserved random vector $\boldsymbol{X}$ is written italic, whereas the data matrix $\mathbf{X}$ is straight.

Let Ω denote the sample space, then each element $\boldsymbol{x} \in \Omega$ has to belong either to g_1 or to g_2. Let furthermore S_1 denote the subspace of observations to which we assign objects from g_1 and S_2 the subspace to which we assign the remaining objects, namely those which we assign to g_2, then $\Omega = S_1 \dot\cup S_2$. The point above the cup symbol denotes that it is a disjoint union. When classifying new objects it can happen that an element from group g_1 is wrongly assigned to g_2. For known probability functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$, the *probability of wrong assignment*, in the following denoted as $\mathbb{P}(2|1)$, can be computed as conditional probability:

$$\mathbb{P}(2|1) = \mathbb{P}(\boldsymbol{X} \in S_2|g_1) = \int_{S_2} f_1(\boldsymbol{x})d\boldsymbol{x} \tag{2.2}$$

This integral is the volume of the density function $f_1(\boldsymbol{x})$ (in the case $p = 1$ it is an area) over the region S_2.
The other case is also possible: an object belonging to g_2 can be wrongly assigned to g_1. The corresponding probability is

$$\mathbb{P}(1|2) = \mathbb{P}(\boldsymbol{X} \in S_1|g_2) = \int_{S_1} f_2(\boldsymbol{x})d\boldsymbol{x}. \tag{2.3}$$

The two possible probabilities of wrong assignment are displayed in Figure 2.2.

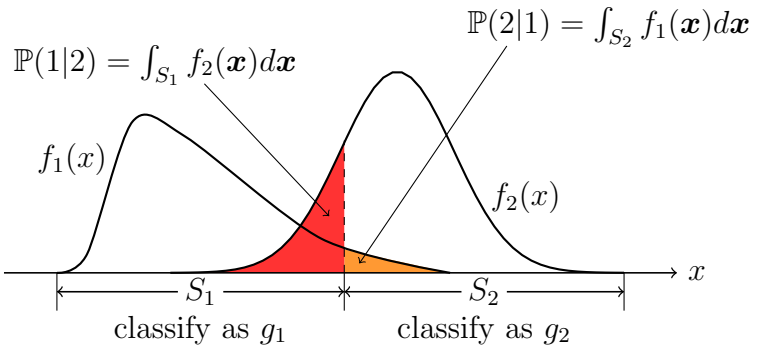


Figure 2.2: Probabilities of misclassification.

Let π_i be the a-priori probability that an object belongs to group g_i, $i \in \{1, 2\}$:

$$\pi_1 = \mathbb{P}(g_1), \pi_2 = \mathbb{P}(g_2) \quad \text{and} \quad \pi_1 + \pi_2 = 1.$$

In general the conditional probability of an event A given an event B is defined as

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{2.4}$$

Using equation (2.4) we can compute the probabilities of correct classification

$$\mathbb{P}(\text{classify an object correctly as } g_1) = \mathbb{P}(\boldsymbol{X} \in S_1|g_1)\mathbb{P}(g_1) = \mathbb{P}(1|1)\pi_1$$
$$\mathbb{P}(\text{classify an object correctly as } g_2) = \mathbb{P}(\boldsymbol{X} \in S_2|g_2)\mathbb{P}(g_2) = \mathbb{P}(2|2)\pi_2$$

and the probabilities of wrong classification

$$\mathbb{P}(\text{classify an object wrongly as } g_1) = \mathbb{P}(\boldsymbol{X} \in S_1|g_2)\mathbb{P}(g_2) = \mathbb{P}(1|2)\pi_2$$
$$\mathbb{P}(\text{classify an object wrongly as } g_2) = \mathbb{P}(\boldsymbol{X} \in S_2|g_1)\mathbb{P}(g_1) = \mathbb{P}(2|1)\pi_1.$$

Misclassification is often directly connected with costs $c \geq 0$ (denoted by $c(2|1)$ when an element from group g_1 is wrongly assigned to g_2, and $c(1|2)$ when an element from group g_2 is wrongly assigned to g_1), correct classification results in costs $c(1|1) = c(2|2) = 0$. In the case $c(2|1) \neq c(1|2)$ we have asymmetric error weights, which means one type of misclassification creates higher costs. The *expected costs of misclassification* (ECM) can then be computed as

$$\begin{aligned} ECM &= c(2|1)\mathbb{P}(2|1)\pi_1 + c(1|2)\mathbb{P}(1|2)\pi_2 \\ &\overset{(2.2),(2.3)}{=} c(2|1)\left(\int_{S_2} f_1(\boldsymbol{x})d\boldsymbol{x}\right)\pi_1 + c(1|2)\left(\int_{S_1} f_2(\boldsymbol{x})d\boldsymbol{x}\right)\pi_2. \end{aligned}$$

The aim of a classification rule is to minimize the ECM. $f_1(\cdot)$ is a density function, which means the integral taken over the whole sample space is one, and as $\Omega = S_1 \dot{\cup} S_2$ by definition, we get

$$1 = \int_\Omega f_1(\boldsymbol{x})d\boldsymbol{x} = \int_{S_1} f_1(\boldsymbol{x})d\boldsymbol{x} + \int_{S_2} f_1(\boldsymbol{x})d\boldsymbol{x}.$$

This leads to

$$\begin{aligned} ECM &= c(2|1)\left(1 - \int_{S_1} f_1(\boldsymbol{x})d\boldsymbol{x}\right)\pi_1 + c(1|2)\left(\int_{S_1} f_2(\boldsymbol{x})d\boldsymbol{x}\right)\pi_2 \\ &= \int_{S_1} \underbrace{\left[f_2(\boldsymbol{x})\pi_2 c(1|2) - f_1(\boldsymbol{x})\pi_1 c(2|1)\right]}_{=:g(\boldsymbol{x})} d\boldsymbol{x} + c(2|1)\pi_1. \end{aligned}$$

The quantities p_1, p_2, $c(1|2)$ and $c(2|1)$ have nonnegative values, $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}) \geq 0\ \forall \boldsymbol{x}$. A possible classification rule which minimizes the ECM is "$\boldsymbol{x}$ belongs to S_1 when $g(\boldsymbol{x}) \leq 0$":

$$\begin{aligned} \boldsymbol{x} \in S_1 &\iff g(\boldsymbol{x}) \leq 0 \\ &\iff f_2(\boldsymbol{x})\pi_2 c(1|2) - f_1(\boldsymbol{x})\pi_1 c(2|1) \leq 0 \\ &\iff f_1(\boldsymbol{x})\pi_1 c(2|1) \geq f_2(\boldsymbol{x})\pi_2 c(1|2) \\ &\iff \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{\pi_2}{\pi_1}\right) \end{aligned} \tag{2.5}$$

The subspace S_2 is then defined by

$$\boldsymbol{x} \in S_2 \iff \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{\pi_2}{\pi_1}\right). \tag{2.6}$$

Three special cases can occur:

(i) $\pi_1 = \pi_2$:

$$\boldsymbol{x} \in S_1 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq \frac{c(1|2)}{c(2|1)}, \qquad \boldsymbol{x} \in S_2 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < \frac{c(1|2)}{c(2|1)}$$

(ii) $c(1|2) = c(2|1)$:

$$\boldsymbol{x} \in S_1 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq \frac{\pi_2}{\pi_1}, \qquad \boldsymbol{x} \in S_2 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < \frac{\pi_2}{\pi_1}$$

(iii) $\pi_1 = \pi_2$ and $c(1|2) = c(2|1)$:

$$\boldsymbol{x} \in S_1 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq 1, \qquad \boldsymbol{x} \in S_2 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < 1$$

Normality assumptions

From now on it is assumed that the populations of g_1 and g_2 are multivariate normally distributed with means $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1$, $\boldsymbol{\Sigma}_2$: $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $i = 1, 2$. Therefore the density functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ have the following form:

$$f_i(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{p}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right] \qquad i = 1, 2$$

When these assumptions do not hold, it can happen that the following classification rules fail severely.

2.1.2 Linear Discriminant Analysis (LDA)

Assuming the classes have a common covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$, the joint density function of $\boldsymbol{X} = (X_1, \ldots, X_p)^\top$ for the populations g_1 and g_2 is given by

$$f_i(\boldsymbol{x}) = \underbrace{\frac{1}{(2\pi)^{\frac{p}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}}_{=const} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right] \qquad i = 1, 2.$$

Given $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}$ are known, according to (2.5) and (2.6) the region where the ECM is maximised can be described by

$$\begin{aligned}
\boldsymbol{x} \in S_1 &\Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{\pi_2}{\pi_1}\right) \\
&\Leftrightarrow \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right] \geq \\
&\geq \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{\pi_2}{\pi_1}\right)
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{x} \in S_2 &\Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{\pi_2}{\pi_1}\right) \\
&\Leftrightarrow \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right] < \\
&< \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{\pi_2}{\pi_1}\right)
\end{aligned}$$

as the constant term can be cancelled and the exponent in the denominator is subtracted from the exponent in the nominator. Since all quantities are nonnegative the logarithm does not change the result:

$$\begin{aligned} \boldsymbol{x} \in S_1 \Leftrightarrow & -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)+ \\ & +\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) \geq \ln\left(\frac{c(1|2)}{c(2|1)}\frac{\pi_2}{\pi_1}\right) \end{aligned} \tag{2.7}$$

The left side of inequality (2.7) can be further simplified:

$$\begin{aligned} & -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)+\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) = \\ = & -\cancel{\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}}+\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+ \\ & +\cancel{\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}}-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2-\frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}+\frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 \\ = & \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+\frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 \end{aligned}$$

The last equation holds because all summands are scalars: we perform matrix-vector multiplications with the dimensions $(1\times p)\cdot(p\times p)\cdot(p\times 1)=(1\times 1)$. For a scalar $a\in\mathbb{R}$ it is true that $a=a^\top$.
Expanding with the term $\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2$ results in

$$\begin{aligned} & (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1-\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+\frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 = \\ = & (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)+\frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2) = \\ = & (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2) \end{aligned}$$

Therefore we can rewrite (2.7) as

$$\begin{aligned} & \boldsymbol{x}\in S_1 \Leftrightarrow \\ & (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2) \geq \ln\left(\frac{c(1|2)}{c(2|1)}\frac{\pi_2}{\pi_1}\right). \end{aligned} \tag{2.8}$$

The left side of the classification rule (2.8) is linear in $\boldsymbol{x}$, which implies that the decision boundary between the two classes is a hyperplane in p dimensions. Assuming equal costs $c(1|2)=(2|1)$, an equivalent description of (2.8) are the so-called *linear discriminant functions*

$$\delta_k(\boldsymbol{x})=\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k+\ln(\pi_k) \qquad \text{for } k=1,2$$

which lead to the decision $G(\boldsymbol{x})=\arg\max_k \delta_k(\boldsymbol{x})$.

Estimating mean and covariance

In reality $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}$ are often unknown and have to be estimated. Assuming we have a realisation of the random variable $\boldsymbol{X} = (X_1, \ldots, X_p)^\top$ with n_1 observations from g_1 and n_2 obervations from g_2, which means $n = n_1 + n_2$, the corresponding data matrices are denoted by

$$\mathbf{X}_1 = \begin{pmatrix} \boldsymbol{x}_{11}^\top \\ \boldsymbol{x}_{12}^\top \\ \vdots \\ \boldsymbol{x}_{1n_1}^\top \end{pmatrix} \in \mathbb{R}^{(n_1 \times p)} \quad \text{and} \quad \mathbf{X}_2 = \begin{pmatrix} \boldsymbol{x}_{21}^\top \\ \boldsymbol{x}_{22}^\top \\ \vdots \\ \boldsymbol{x}_{2n_2}^\top \end{pmatrix} \in \mathbb{R}^{(n_2 \times p)}$$

The sample means and sample covariance matrices are given by

$$\begin{aligned} \bar{\boldsymbol{x}}_1 &= \frac{1}{n_1} \sum_{j=1}^{n_1} \boldsymbol{x}_{1j}, \quad \boldsymbol{S}_1 = \frac{1}{n_1 - 1} \sum_{j=1}^{n_1} (\boldsymbol{x}_{1j} - \bar{\boldsymbol{x}}_1)(\boldsymbol{x}_{1j} - \bar{\boldsymbol{x}}_1)^\top \\ \bar{\boldsymbol{x}}_2 &= \frac{1}{n_2} \sum_{j=1}^{n_2} \boldsymbol{x}_{2j}, \quad \boldsymbol{S}_2 = \frac{1}{n_2 - 1} \sum_{j=1}^{n_2} (\boldsymbol{x}_{2j} - \bar{\boldsymbol{x}}_2)(\boldsymbol{x}_{2j} - \bar{\boldsymbol{x}}_2)^\top \end{aligned} \tag{2.9}$$

with $\bar{\boldsymbol{x}}_i \in \mathbb{R}^{p \times 1}$ and $\boldsymbol{S}_i \in \mathbb{R}^{p \times p}$ for $i = 1, 2$. Since one assumption was that the populations g_1 and g_2 have the same covariance matrix $\boldsymbol{\Sigma}$, the sample covariance matrices are combined to the *pooled covariance matrix*

$$\boldsymbol{S}_{pooled} = \left(\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1)} \right) \boldsymbol{S}_1 + \left(\frac{n_2 - 1}{(n_1 - 1) + (n_2 - 1)} \right) \boldsymbol{S}_2, \tag{2.10}$$

which is an unbiased estimator of $\boldsymbol{\Sigma}$. Rule (2.8) can now be written as

$$\begin{aligned} &\boldsymbol{x} \in S_1 \Leftrightarrow \\ &(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} \boldsymbol{x} - \frac{1}{2} (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} (\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2) \geq \ln \left(\frac{c(1|2)}{c(2|1)} \frac{\pi_2}{\pi_1} \right). \end{aligned} \tag{2.11}$$

Additionally, the prior probabilities π_1 and π_2 can be estimated by the ratios $\frac{n_1}{n}$ and $\frac{n_2}{n}$. An important special case occurs when the costs of misclassification and the prior probabilities are equal:

$$\left. \begin{matrix} c(1|2) = c(2|1) \\ \pi_1 = \pi_2 \end{matrix} \right\} \Rightarrow \frac{c(1|2)}{c(2|1)} \frac{\pi_2}{\pi_1} = \ln(1) = 0$$

Rule (2.11) can then be simplified by defining

$$\hat{y} := \underbrace{(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1}}_{=:\hat{\boldsymbol{a}}^\top} \boldsymbol{x} \tag{2.12}$$

$$\begin{aligned} \hat{m} :=& \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2) \\ =& \frac{1}{2}(\bar{y}_1 + \bar{y}_2) \end{aligned}$$

with

$$\bar{y}_1 = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} \boldsymbol{x}_1 = \hat{\boldsymbol{a}}^\top \bar{\boldsymbol{x}}_1$$

and

$$\bar{y}_2 = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} \boldsymbol{x}_2 = \hat{\boldsymbol{a}}^\top \bar{\boldsymbol{x}}_2$$

and formulating the classification rule as

$$\boldsymbol{x}_0 \in S_1 \Leftrightarrow \hat{y}_0 = \hat{\boldsymbol{a}}^\top \boldsymbol{x}_0 \geq \hat{m}.$$

The quantity y is one-dimensional and is a linear combination of observations from both groups, g_1 and g_2.

Figure 2.3 shows what happens when we use LDA although the group variances are not equal. The dotted curves represent the class distributions under the assumption $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$, the solid curves represent the class distributions for unequal variances. According to LDA, the point $\hat{m}$ marks the decision boundary. We can see that for the solid curves, the probabilities of misclassification differ significantly for the two groups: More observations from g_2 will be wrongly classified in this setting.

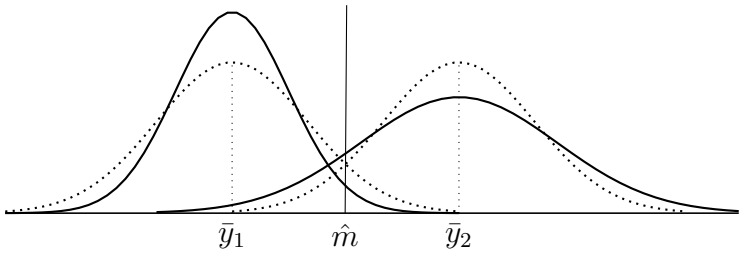


Figure 2.3: Probabilities of misclassification for equal and unequal variances.

2.1.3 Quadratic Discriminant Analysis (QDA)

In the case of unequal covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, the ratio of the density functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ in (2.5) and (2.6) is more complicated. Under the normality assumption it is again easier to examine the logarithm of the ratio:

$$\begin{aligned}\ln\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) &= \ln\left(\frac{|\boldsymbol{\Sigma}_2|^{\frac{1}{2}}}{|\boldsymbol{\Sigma}_1|^{\frac{1}{2}}}\right) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\\ &= \frac{1}{2}\ln\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right) - \frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\Sigma}_2^{-1})\boldsymbol{x} + (\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1})\boldsymbol{x} +\\ &\quad - \frac{1}{2}\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\end{aligned}$$

Therefore we get the following classification rule:

$$\begin{aligned}&\boldsymbol{x} \in S_1 \Leftrightarrow\\ &-\frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\Sigma}_2^{-1})\boldsymbol{x} + (\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1})\boldsymbol{x} - C \geq \ln\left(\frac{c(1|2)}{c(2|1)}\frac{\pi_2}{\pi_1}\right)\\ &\boldsymbol{x} \in S_2 \Leftrightarrow\\ &-\frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\Sigma}_2^{-1})\boldsymbol{x} + (\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1})\boldsymbol{x} - C < \ln\left(\frac{c(1|2)}{c(2|1)}\frac{\pi_2}{\pi_1}\right)\end{aligned} \tag{2.13}$$

with

$$C = \frac{1}{2}\ln\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right) + \frac{1}{2}(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2)$$

The constant term C only depends on mean and covariance of the distributions. In the case $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$, (2.13) can be reduced to rule (2.8). The above classificaton rule is quadratic in $\boldsymbol{x}$.
Assuming equal costs $c(1|2) = (2|1)$, the *quadratic discriminant functions* are

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\ln|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \ln(\pi_k) \qquad \text{for } k = 1, 2$$

and lead to the decision $G(\boldsymbol{x}) = \arg\max_k \delta_k(\boldsymbol{x})$. As the quantities $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are generally unknown they can be again estimated by the corresponding sample means and covariances $\bar{\boldsymbol{x}}_1$, $\bar{\boldsymbol{x}}_2$, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$.

2.1.4 Robust Discriminant Analysis

As $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are directly used in the computation of (2.8) and (2.13), the estimation of these two quantities is crucial.
The classical estimators (2.9) are not robust, which means they are not resistant

to outliers. The so-called *breakdown point* ε_n^* is the smallest amount of data contamination which can cause an arbitrary value of an estimator. In Donoho and Huber (1983) it is defined as follows:

Definition. *Let T be an estimator and let $x_1, \ldots, x_n$ be a sample. Replace m data points $x_{1_1}, \ldots, x_{1_m}$ by arbitary values $y_1, \ldots, y_m$ and denote the new data with $z_1, \ldots, z_n$. Then the (gross-error)* breakdown point *(for finite samples) of T is defined by*

$$\varepsilon_n^*(T; x_1, \ldots, x_n) = \min \left\{ \frac{m}{n}; \max_{i_1,\ldots,i_m} \sup_{y_1,\ldots,y_m} |T(z_1, \ldots, z_n)| = \infty \right\}$$

According to Hampel (1971), the asymptotic version of the definition of the *breakdown point for infinite samples* ε^* is in general provided by

$$\lim_{n\to\infty} \varepsilon_n^* = \varepsilon^*.$$

The arithmetic mean has the breakdown point $\varepsilon^* = 0\%$, which means that already one outlier can distort the estimator arbitrarily. A more robust location estimation is the $\alpha\%$-trimmed mean, which is the arithmetic mean of the central $(100-2\alpha)\%$ of the sorted sample, with $\varepsilon^* = \alpha\%$. The median has the maximal possible breakdown point $\varepsilon^* = 50\%$. The maximal value is 50% because in the case of a bigger value the majority of outliers could be seen as "good" data points.

In the case of robust discriminant analysis we want to find robust estimators $\boldsymbol{T} : \mathbb{R}^{(n\times p)} \to \mathbb{R}^p$ and $\boldsymbol{C} : \mathbb{R}^{(n\times p)} \to \mathbb{R}^{(p\times p)}$ of the multivariate location $\boldsymbol{\mu}$ and scale $\boldsymbol{\Sigma}$ in (2.8) and (2.13). $\boldsymbol{T}$ and $\boldsymbol{C}$ are functions of the data matrix $\mathbf{X}$. Two popular methods are the *Minimum Volume Ellipsoid* (MVE) and the *Minimum Covariance Determinant* (MCD) estimators which both feature a maximal breakdown point. According to Rousseeuw (1985) they are defined as follows:

- **MVE**: determine the ellipsoid with minimal volume which includes at least half of the data points of $\mathbf{X}$; then $\boldsymbol{T}(\mathbf{X})$ is the centre of that ellipsoid and $\boldsymbol{C}(\mathbf{X})$ is given by the same ellipsoid, multiplied with a factor for consistency at normal distributions

- **MCD**: determine the ellipsoid which includes at least h data points of $\mathbf{X}$ and whose empirical covariance matrix has the smallest determinant; then $\boldsymbol{T}(\mathbf{X})$ is the centre of that ellipsoid, and $\boldsymbol{C}(\mathbf{X})$ is given by the same ellipsoid, multiplied with a factor for consistency at normal distributions

Besides the high breakdown point the MVE estimator offers another desirable property: it is *affine equivariant*, which means that it is independent of the underlying

measurement scale and coordinate system. We are therefore allowed to transform the data, e.g. standardise them, and the MVE estimator also correctly transforms the result. Let $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ be a non-singular matrix and $\boldsymbol{b} \in \mathbb{R}^p$, then the following equations should hold for $\boldsymbol{T}$ and $\boldsymbol{C}$:

$$\begin{aligned} \boldsymbol{T}(\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{b}, \ldots, \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{b}) &= \boldsymbol{A} \cdot \boldsymbol{T}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) + \boldsymbol{b} \\ \boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{b}, \ldots, \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{b}) &= \boldsymbol{A} \cdot \boldsymbol{C}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \cdot \boldsymbol{A}^\top \end{aligned} \tag{2.14}$$

This property is especially important for distributions with elliptic symmetry, which means the density has the form

$$f_{\boldsymbol{\mu},\Sigma}(\boldsymbol{x}) = \frac{1}{\sqrt{\det \Sigma}}\, g\left(\sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}\right),$$

e.g. for the multivariate normal distribution.
For the MVE estimator the equations (2.14) hold due to the fact that

$$V(\boldsymbol{A}\mathbf{X} + \boldsymbol{b}) = |\det \boldsymbol{A}| \cdot V(\mathbf{X})$$

where $V(x)$ denotes the volume of x.

For the breakdown point of the MVE estimator we get

$$\varepsilon_n^*(\boldsymbol{T}, \mathbf{X}) = \frac{\left[\frac{n}{2}\right] - p + 1}{n} \to 50\%.$$

If the parameter h for the MCD estimator is set to $h \approx \frac{n}{2}$ the maximal breakdown point is achieved. For increasing h the breakdown point decreases to $\frac{n-h}{n}$. A good compromise between high breakdown point and acceptable efficiency is $h = \frac{3}{4}n$.

2.1.5 Fisher's Linear Discriminant

Starting from a different idea, the British statistician Sir Ronald Aylmer Fisher developed a linear discriminant function similar to (2.12), see Fisher (1938). He tried to transform multivariate to univariate observations with the aim, that the two transformed groups are separated as much as possible. The transformation $y = \boldsymbol{a}^\top \boldsymbol{x}$ is a fixed linear combination of the input variable $\boldsymbol{x}$ and results in univariate quantities $y_{11}, y_{12}, \ldots, y_{1n_1}$ for the first and $y_{21}, y_{22}, \ldots, y_{2n_2}$ for the second group. The separation of the two populations is made by maximising the distance between the arithmetic means of the univariate values. Using units of the standard deviation as measure for this distance, the following criterion can be fomulated:

$$\max_{\boldsymbol{a}} \frac{|\bar{y}_1 - \bar{y}_2|}{s_y} \tag{2.15}$$

The vector $\boldsymbol{a}$ with $||\boldsymbol{a}|| = 1$ is the required direction of projection and s_y stands for the root of the pooled variance of the univariate y-values defined by

$$s_y^2 = \frac{\sum_{i=1}^{n_1}(y_{1i} - \bar{y}_1)^2 + \sum_{i=1}^{n_2}(y_{2i} - \bar{y}_2)^2}{n_1 + n_2 - 2}.$$

By squaring the main condition in (2.15), the direction is not changed. Equivalent formulations of the target function are

$$\frac{(\bar{y}_1 - \bar{y}_2)^2}{s_y^2} = \frac{(\boldsymbol{a}^\top \bar{\boldsymbol{x}}_1 - \boldsymbol{a}^\top \bar{\boldsymbol{x}}_2)^2}{\boldsymbol{a}^\top \boldsymbol{S}_{pooled} \boldsymbol{a}} \tag{2.16}$$

with $\boldsymbol{S}_{pooled}$ defined as (2.10). For solving (2.15) we use the following inequality:

Extended Cauchy-Schwarz inequality
Let $\boldsymbol{b}, \boldsymbol{d} \in \mathbb{R}^{(p\times 1)}$ and $\boldsymbol{B}$ be a p-dimensional positive definite matrix, then

$$(\boldsymbol{b}^\top \boldsymbol{d})^2 \leq (\boldsymbol{b}^\top \boldsymbol{B} \boldsymbol{b})(\boldsymbol{d}^\top \boldsymbol{B}^{-1} \boldsymbol{d}) \tag{2.17}$$

and equality holds if and only if there exists a constant $c \in \mathbb{R}$ such that $\boldsymbol{b} = c\boldsymbol{B}^{-1}\boldsymbol{d}$ (see e.g. Johnson and Wichern (2007)).

Setting $\boldsymbol{b} = \hat{\boldsymbol{a}}$, $\boldsymbol{d} = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$ and $\boldsymbol{B} = \boldsymbol{S}_{pooled}$, (2.17) becomes

$$\left[\hat{\boldsymbol{a}}^\top (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)\right]^2 \leq (\hat{\boldsymbol{a}}^\top \boldsymbol{S}_{pooled} \hat{\boldsymbol{a}}) \underbrace{(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)}_{=:D^2}$$

which is equivalent to

$$\frac{\left[\hat{\boldsymbol{a}}^\top (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)\right]^2}{\hat{\boldsymbol{a}}^\top \boldsymbol{S}_{pooled} \hat{\boldsymbol{a}}} \leq D^2. \tag{2.18}$$

We have found an upper boundary for (2.16). As stated before, the equality in equation (2.18) holds if an only if there exists a constant c such that $\hat{\boldsymbol{a}} = c\boldsymbol{S}_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$. The constant can be defined as 1 and therefore the maximal value of (2.16), which is D^2, is achieved for $\hat{\boldsymbol{a}} = \boldsymbol{S}_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$. The linear combination which solves (2.15) is given by

$$\hat{y} = \hat{\boldsymbol{a}}^\top \boldsymbol{x} = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} \boldsymbol{x}. \tag{2.19}$$

Finally, the following classification rule can be formulated:

$$\begin{aligned} \boldsymbol{x}_0 \in S_1 &\Leftrightarrow \hat{y}_0 \geq \hat{m} \\ &\Leftrightarrow (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} \boldsymbol{x}_0 \geq \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1} (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2) \end{aligned} \tag{2.20}$$

We can see that Fisher's linear discriminant is a special case of rule (2.8): if the expected costs for misclassification and the a-priori probabilities are equal, which means that if $c(2|1) = c(1|2)$ and $\pi_1 = \pi_2$, rule (2.8) and rule (2.20) are the same. Using the fact that

$$\begin{aligned}\frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^\top \boldsymbol{S}_{pooled}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2) &= \frac{1}{2}(\hat{\boldsymbol{a}}^\top \bar{\boldsymbol{x}}_1 + \hat{\boldsymbol{a}}^\top \bar{\boldsymbol{x}}_2) \\ &= \frac{1}{2}(\bar{y}_1 + \bar{y}_2),\end{aligned}$$

rule (2.20) can be written as

$$\boldsymbol{x}_0 \in S_1 \Leftrightarrow \hat{y}_0 \geq \frac{1}{2}(\bar{y}_1 + \bar{y}_2).$$

In Figure 2.4, rule (2.20) is visualised for the case $p = 2$.

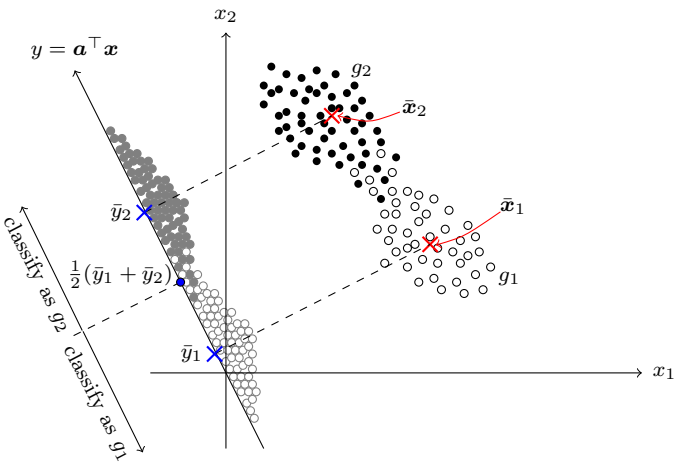


Figure 2.4: Schematic visualisation of Fisher's linear discriminant.

In contrast to rule (2.8), the normality assumption of the populations is not needed for Fisher's classification rule. However, the assumption that the classes have a common covariance matrix is still implicitly made, as the pooled covariance matrix is used in (2.15).

2.2 Support Vector Machine (SVM)

Let again be $K = 2$, which means there are two groups. When the data clouds of the two groups overlap, the classes are non-separable and linear decision boundaries, among others, perform poorly. Support Vector Machines transform the

feature space into a higher-dimensional space and construct linear decision boundaries there. This Section closely follows the first part of Chapter 12 in Hastie et al. (2009).

2.2.1 Linear hyperplanes

Assuming that training data is available, our training data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ has the following form:

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{x}_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

The class membership is denoted by the vector $\boldsymbol{g} \in \mathbb{R}^n$ with $g_i \in \{-1, 1\}$ for $i = 1 \dots, n$. We can summarise the training information to n pairs

$$(\boldsymbol{x}_1, g_1), (\boldsymbol{x}_2, g_2), \dots, (\boldsymbol{x}_n, g_n).$$

Assuming $\boldsymbol{\beta}$ is a unit vector ($||\boldsymbol{\beta}|| = 1$), a hyperplane can be characterised by the affine set

$$\{\boldsymbol{x} \in \Omega : \underbrace{\boldsymbol{x}^\top \boldsymbol{\beta} + \beta_0}_{=: f(\boldsymbol{x})} = 0\} \tag{2.21}$$

with the corresponding classification rule

$$G(\boldsymbol{x}) = \text{sgn}\left[f(\boldsymbol{x})\right] \tag{2.22}$$

(sgn stands for the signum function). The function value $f(\boldsymbol{x}_0)$ denotes the signed distance of the observation $\boldsymbol{x}_0$ to the hyperplane $f(\boldsymbol{x}) = 0$ (in the case $||\boldsymbol{\beta}|| \neq 0$ the signed distance would be $\frac{1}{||\boldsymbol{\beta}||} f(\boldsymbol{x})$). If an observation $\boldsymbol{x}_i$ is correctly classified, then $g_i = \text{sgn}\left[f(\boldsymbol{x}_i)\right]$ and therefore $g_i f(\boldsymbol{x}_i) > 0$.

The separable case

In the separable case we can always find a function $f : \mathbb{R}^p \to \mathbb{R}$ defined in (2.21) with $g_i f(\boldsymbol{x}_i) > 0 \; \forall i \in \{1, \dots, n\}$, which means the observations can be completely separated by a hyperplane. The distance M is defined as the minimal distance of a point to the hyperplane taken over all observations:

$$M = \min_{i=1,\dots,n} g_i f(\boldsymbol{x_i})$$

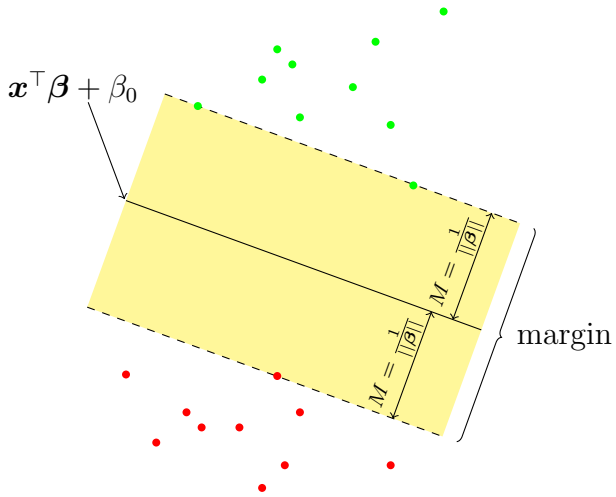


Figure 2.5: The separable case.

An example for the two-dimensional case is shown in Figure 2.5. The yellow band on both sides of the hyperplane in Figure 2.5, which is exactly $2M$ units wide, is called the *margin*. The aim is to find the biggest margin between the training data of the two classes by solving the optimisation problem

$$\begin{aligned} &\max_{\substack{\boldsymbol{\beta},\beta_0 \\ ||\boldsymbol{\beta}||=1}} M \\ \text{s.t.} \quad &g_i(\boldsymbol{x}_i^\top\boldsymbol{\beta} + \beta_0) \geq M \quad i = 1, \ldots, n. \end{aligned} \tag{2.23}$$

This approach provides a unique solution for the problem of finding a separating hyperplane between the classes. The side conditions ensure that the distance of each data point to the decision boundary is at least M and the main condition seeks the maximal value of M over the parameters $\boldsymbol{\beta}$ and β_0, which define the hyperplane.
We still have the requirement that $\boldsymbol{\beta}$ has to be normed, but we can avoid it: assuming we drop the constraint $||\boldsymbol{\beta}|| = 1$, we replace the side conditions in (2.23) by

$$\frac{1}{||\boldsymbol{\beta}||} g_i(\boldsymbol{x}_i\boldsymbol{\beta} + \beta_0) \geq M \quad i = 1, \ldots, n, \tag{2.24}$$

which leads to new coefficients $\tilde{\boldsymbol{\beta}}$ and $\tilde{\beta}_0$ with

$$\tilde{\boldsymbol{\beta}} = \frac{\boldsymbol{\beta}}{||\boldsymbol{\beta}||}, \quad \tilde{\beta}_0 = \frac{\beta_0}{||\boldsymbol{\beta}||}$$

and $||\tilde{\boldsymbol{\beta}}|| = 1$. Multiplying (2.24) with $||\boldsymbol{\beta}||$ leads to the new side conditions

$$g_i(\boldsymbol{x}_i\boldsymbol{\beta} + \beta_0) \geq ||\boldsymbol{\beta}||M \quad i = 1, \ldots, n. \tag{2.25}$$

Let $\boldsymbol{\beta}$ and β_0 satisfy (2.25), then any positively scaled multiple of them also satisfy (2.25). Setting $||\boldsymbol{\beta}|| = \frac{1}{M}$ and using the fact that

$$\min_{a \in \mathbb{R}^+} \frac{1}{a} = \max_{a \in \mathbb{R}^+} a$$

leads to a more familiar formulation of the support vector criterion for separated data:

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \beta_0} ||\boldsymbol{\beta}|| \\ \text{s.t.} \quad &g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 \quad i = 1, \ldots, n \end{aligned} \tag{2.26}$$

Problem (2.26) is a convex optimisation problem with a quadratic criterion and linear inequality constraints. When using the Lagrange method to solve it, one has to minimize the *Lagrange primal function* defined as

$$L_p = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{n} \alpha_i [g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) - 1] \tag{2.27}$$

with respect to $\boldsymbol{\beta}$ and β_0. The *Lagrange multipliers* α_i have to be nonnegative, $\alpha_i \geq 0$. As norm functions only take values in $\mathbb{R}_+$ we can square the target function and add the costant $\frac{1}{2}$ without changing the resulting minimum. Assuming $||.||$ stands for the Euclidean norm, which means $||\boldsymbol{\beta}||^2 = \boldsymbol{\beta}^\top \boldsymbol{\beta}$, these modifications make the derivation easier:

$$\begin{aligned} \frac{\partial L_p}{\partial \boldsymbol{\beta}} &= \boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i g_i \boldsymbol{x}_i \\ \frac{\partial L_p}{\partial \beta_0} &= \sum_{i=1}^{n} \alpha_i g_i \end{aligned}$$

Setting these derivatives to zero and substituting them in (2.27) results in the *Lagrange dual function*

$$\begin{aligned} L_d &= \frac{1}{2}(\sum_{i=1}^{n} \alpha_i g_i \boldsymbol{x}_i)^\top (\sum_{j=1}^{n} \alpha_j g_j \boldsymbol{x}_j) - (\sum_{i=1}^{n} \alpha_i g_i \boldsymbol{x}_i)^\top (\sum_{j=1}^{n} \alpha_j g_j \boldsymbol{x}_j) - \beta_0 \underbrace{\sum_{i=1}^{n} \alpha_i g_i}_{=0} + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j \boldsymbol{x}_i^\top \boldsymbol{x}_j, \end{aligned} \tag{2.28}$$

which gives a lower bound on the objective function (2.26). The solution of problem (2.26) is then obtained by solving the following simpler convex optimisation

problem:

$$\begin{aligned} &\max_{\alpha_i} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \right] \\ &\text{s.t.} \quad \alpha_i \geq 0 \text{ for } i = 1, \ldots, n \end{aligned} \tag{2.29}$$

In order to be optimal, the solution of problem (2.29) also has to satisfy the so-called *Karush-Kuhn-Tucker conditions*

$$\sum_{i=1}^{n} \alpha_i g_i \boldsymbol{x}_i = \boldsymbol{\beta} \tag{2.30}$$

$$\sum_{i=1}^{n} \alpha_i g_i = 0 \tag{2.31}$$

$$\alpha_i \left[g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) - 1 \right] = 0 \quad i = 1, \ldots, n. \tag{2.32}$$

From the side condition of problem (2.29) and condition (2.32) we can see that the following implications are true for all $i = 1, \ldots, n$:

- $\alpha_i > 0 \Rightarrow g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) = 1$: the observation $\boldsymbol{x}_i$ lies on one of the boundaries of the margin; these points are called *support vectors*
- $g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) > 1 \Rightarrow \alpha_i = 0$: the observation $\boldsymbol{x}_i$ does not lie on one of the boundaries, but outside of the margin

The case that an observation lies within the margin cannot occur in the separable case. Having a closer look at condition (2.30) we can see that the solution vector $\boldsymbol{\beta}$ from problem (2.26) is a linear combination of the support vectors: if α_i is equal to zero, the i^{th} summand in (2.30) is zero and therefore only the summands of the support vectors do not vanish. Finally, the intercept β_0 can be computed by solving equation (2.32) for any of the support vectors. The separating hyperplane in Figure 2.5 has three support vectors.

The non-separable case

In the non-separable case the two classes overlap and we have to take into account that we cannot find a hyperplane where all points are on the correct side. There exist two types of misclassification:

- an observation $\boldsymbol{x}_i$ with $g_i = 1$ is misclassified when $f(\boldsymbol{x}_i) < 0$
- an observation $\boldsymbol{x}_j$ with $g_j = -1$ is misclassified when $f(\boldsymbol{x}_j) > 0$

In order to include unavoidable misclassified points in our optimisation problem, a so-called *slack variable* $\xi_i \geq 0$ is introduced for each side condition. The value of ξ_i is the violation of the corresponding side condition: $\xi_i > 0$ means the point $\boldsymbol{x}_i$ is on the wrong side of its margin by the amount of $M\xi_i$, $g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) < M$. Points $\boldsymbol{x}_j$ on the correct side of the margin have slack variables $\xi_j = 0$. An example is given in Figure 2.6.

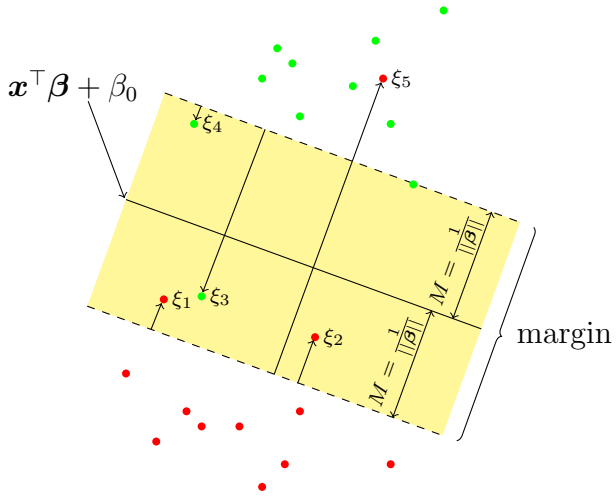


Figure 2.6: The non-separable case.

The new side conditions are $g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq M(1 - \xi_i)$, which measure the overlap of obervations on the wrong side of the margin in relative distance. As the sum of violations should be as small as possible, $\sum_{i=1}^n \xi_i \leq const$ is added to the optimisation problem. When $\xi_i > 1$ the observation $\boldsymbol{x}_i$ is misclassified, therefore the restriction $\sum_{i=1}^n \xi_i \leq C$ means the total amount of misclassified training observations has to be smaller than the constant C. Then (2.26) can be written as

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} ||\boldsymbol{\beta}|| \\ \text{s.t.} \quad &\begin{cases} g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i \text{ for } i = 1, \ldots, n \\ \xi_i \geq 0 \text{ for } i = 1, \ldots, n \\ \sum_{i=1}^n \xi_i \leq const. \end{cases} \end{aligned} \tag{2.33}$$

Points clearly outside the margins do not play an important role for determining $\boldsymbol{\beta}$ and β_0 and thus can be ignored for shaping the class boundary. In linear discriminant analysis, by contrast, all points have influence on the decision rule through the mean vectors and covariance matrices. This property of the SVM can be useful

in the presence of outliers in the data which do not lie near the decision boundary. Problem (2.33) is also a convex optimisation problem and can be again solved by using Lagrange multipliers. As in the separable case we replace $||\boldsymbol{\beta}||$ by $\frac{1}{2}||\boldsymbol{\beta}||^2$ for easier derivation. The side condition $\sum_{i=1}^{n} \xi_i \leq const$ is included in the main condition by adding the term $C\sum_{i=1}^{n} \xi_i$, where the C is the so-called *cost parameter*. In the separable case C is set to ∞. Problem (2.33) can then be written as

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} \left[\frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n}\xi_i\right] \\ \text{s.t.} \quad &\begin{cases} g_i(\boldsymbol{x}_i^\top\boldsymbol{\beta}+\beta_0) \geq 1-\xi_i \text{ for } i=1,\ldots,n \\ \xi_i \geq 0 \text{ for } i=1,\ldots,n. \end{cases} \end{aligned} \tag{2.34}$$

The corresponding *Lagrange primal function*, which has to be minimised with respect to $\boldsymbol{\beta}, \beta_0$ and λ_i, is

$$L_p = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[g_i(\boldsymbol{x}_i^\top\boldsymbol{\beta}+\beta_0) - (1-\xi_i)] - \sum_{i=1}^{n}\lambda_i\xi_i \tag{2.35}$$

and the required derivatives of L_p are

$$\begin{aligned} \frac{\partial L_p}{\partial \boldsymbol{\beta}} &= \boldsymbol{\beta} - \sum_{i=1}^{n}\alpha_i g_i \boldsymbol{x}_i \\ \frac{\partial L_p}{\partial \beta_0} &= \sum_{i=1}^{n}\alpha_i g_i \\ \frac{\partial L_p}{\partial \xi_i} &= C - \alpha_i - \lambda_i \quad \forall i = 1,\ldots,n. \end{aligned}$$

Additionally α_i, λ_i and ξ_i have to be nonnegative. Setting these derivatives to zero and substituting them in (2.35) we obtain the *Lagrange dual function*

$$\begin{aligned} L_d =& \frac{1}{2}(\sum_{i=1}^{n}\alpha_i g_i \boldsymbol{x}_i)^\top(\sum_{j=1}^{n}\alpha_j g_j \boldsymbol{x}_j) + \cancel{\sum_{j=1}^{n}(\alpha_i+\lambda_i)\xi_i} - (\sum_{i=1}^{n}\alpha_i g_i \boldsymbol{x}_i)^\top(\sum_{j=1}^{n}\alpha_j g_j \boldsymbol{x}_j) + \\ &- \beta_0 \underbrace{\sum_{i=1}^{n}\alpha_i g_i}_{=0} + \sum_{i=1}^{n}\alpha_i - \cancel{\sum_{i=1}^{n}\alpha_i\xi_i} - \cancel{\sum_{i=1}^{n}\lambda_i\xi_i} \\ =& \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j g_i g_j \boldsymbol{x}_i^\top \boldsymbol{x}_j. \end{aligned} \tag{2.36}$$

The solution of problem (2.33) is then obtained by solving the following simpler convex optimisation problem:

$$\begin{aligned} &\max_{\alpha_i} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \right] \\ &\text{s.t.} \quad 0 \leq \alpha_i \leq C \text{ for } i = 1, \ldots, n \end{aligned} \tag{2.37}$$

In order to be optimal, the solution of problem (2.37) also has to satisfy the *Karush-Kuhn-Tucker conditions*

$$\sum_{i=1}^{n} \alpha_i g_i \boldsymbol{x}_i = \boldsymbol{\beta} \tag{2.38}$$

$$\sum_{i=1}^{n} \alpha_i g_i = 0 \tag{2.39}$$

$$C = \alpha_i + \lambda_i \tag{2.40}$$

$$\lambda_i \xi_i = 0 \tag{2.41}$$

$$\alpha_i \left[g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \right] = 0 \tag{2.42}$$

$$g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \geq 0. \tag{2.43}$$

Conditions (2.40)–(2.43) have to hold for $i = 1, \ldots, n$. As in the separable case we can see from constraint (2.38) that the solution of $\boldsymbol{\beta}$ is a linear combination of those $\boldsymbol{x}_i$ for which $\alpha_i > 0$, the other summands are zero. These observations $\boldsymbol{x}_i$ with positive α_i are again called *support vectors* as $\boldsymbol{\beta}$ is only constructed out of them. According to condition (2.42), if $\alpha_i > 0$ then $g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) = (1 - \xi_i)$ which leads to two kinds of support vectors:

- $\xi_i = 0$: the observation $\boldsymbol{x}_i$ lies on one of the two boundaries of the margin; due to conditions (2.40) and (2.41) these vectors are characterised by $0 < \alpha_i < C$
- $\xi_i > 0$: the observation $\boldsymbol{x}_i$ does not lie on one of the two boundaries of the margin; as condition (2.41) implies $\xi_i > 0 \Rightarrow \lambda_i = 0$, these vectors are characterised by $\alpha_i = C$

Any of the margin points with $\alpha_i > 0$ and $\xi_i = 0$ can be used to determine the intercept β_0 by solving the equation $g_i(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \beta_0) = 1$. In order to obtain numerical stability, the average over all of the solutions can be computed. By changing the cost parameter C the amount of support vectors and the width of the margin changes and therefore C is a so-called *tuning parameter*.

2.2.2 Moving beyond linearity

Often linear hyperplanes are just a convenient approximation of a much better separation of the classes. Moreover, linear models are easy to calculate and do not easily overfit. A possible compromise between a linear model and a nonlinear decision boundary can be achieved by using transformations of the original data $\boldsymbol{x} = (x_1, \ldots, x_p)$ as input.
Let $h_m(\boldsymbol{x})$ be the m^{th} transformation of $\boldsymbol{x}$ with $h_m : \mathbb{R}^p \to \mathbb{R},\ m = 1, \ldots, M$. A *linear basis expansion* of $\boldsymbol{x}$ is then defined as

$$H(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m h_m(\boldsymbol{x})$$

Let $p < M$ and $h : \mathbb{R}^p \to \mathbb{R}^M$ be the transformation in a higher-dimensional feature space, then our new input features are $h(\boldsymbol{x_i}) = (h_1(\boldsymbol{x_i}), \ldots, h_M(\boldsymbol{x}_i))$ instead of $\boldsymbol{x}_i$ for $i = 1, \ldots, n$. We re-define the linear function $f(\cdot)$ in (2.21) to

$$f(\boldsymbol{x}) := h(\boldsymbol{x})^\top \boldsymbol{\beta} + \beta_0, \tag{2.44}$$

with $\boldsymbol{\beta} \in \mathbb{R}^M$ and $\beta_0 \in \mathbb{R}$. As the basis function h_m are fixed, the model is linear in the new variables $h(\boldsymbol{x})$. This fact causes that the fitting is computed as before, although we actually work with a larger feature space. The function (2.44) is nonlinear in $\boldsymbol{x}$ which results in a nonlinear classifier defined by

$$\widehat{G}(\boldsymbol{x}) = \operatorname{sgn}\left[\hat{f}(\boldsymbol{x})\right].$$

with $\hat{f}(\boldsymbol{x}) = h(\boldsymbol{x})^\top \hat{\boldsymbol{\beta}} + \hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}, \hat{\beta}_0$ being the estimated coefficients. In the case of Support Vector Machines typical basis expansions are polynomials and splines.

The optimisation problem (2.35) and its solution can be represented in a way that only involves the new input features as inner products:
In the following we work with the transformed feature vectors $h(\boldsymbol{x}_i)$ instead of $\boldsymbol{x}_i$. Using the notation $\langle \cdot, \cdot \rangle$ for the inner product, the Langrangian dual function (2.36) can be written as

$$L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j) \rangle \tag{2.45}$$

and using constraint (2.38) the solution function $f(\boldsymbol{x})$ can be written as

$$\begin{aligned} f(\boldsymbol{x}) &= h(\boldsymbol{x})^\top \boldsymbol{\beta} + \beta_0 \\ &= h(\boldsymbol{x})^\top \left(\sum_{i=1}^{n} \alpha_i g_i h(\boldsymbol{x}_i) \right) + \beta_0 \\ &= \sum_{i=1}^{n} \alpha_i g_i \langle h(\boldsymbol{x}), h(\boldsymbol{x}_i) \rangle + \beta_0. \end{aligned} \tag{2.46}$$

The intercept β_0 is again determined by solving the equation $g_i f(\boldsymbol{x}_i) = 1$ for any $\boldsymbol{x}_i$ with $0 < \alpha_i < C$. As we can see, (2.45) and (2.46) involve $h(\boldsymbol{x})$ only through inner products and therefore we do not need to specify the transformation $h(\cdot)$. It is enough to know a special symmetric positive (semi-) definite function $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, the so-called *kernel function*, with

$$K(\boldsymbol{u}, \boldsymbol{v}) = \langle h(\boldsymbol{u}), h(\boldsymbol{v}) \rangle.$$

K computes inner products in the transformed feature space. The following three choices for K are implemented in the R-function `svm()` from the package `e1071`:

- **Linear kernel**:
$$K(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$$
- (d^{th}-degree) **Polynomial kernel**:
$$K(\boldsymbol{u}, \boldsymbol{v}) = (c_0 + \gamma \langle \boldsymbol{u}, \boldsymbol{v} \rangle)^d \quad \text{for a constant } c_0, \gamma > 0$$
- **Radial basis kernel** (also called *RBF* (from Radial Basis Function) or *Gaussian* kernel):
$$K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma ||\boldsymbol{u} - \boldsymbol{v}||^2) \quad \text{for } \gamma > 0$$
- **Sigmoid kernel** (also called *neural network* or *hyperbolic tangent* kernel):
$$K(\boldsymbol{u}, \boldsymbol{v}) = \tanh(\gamma \langle \boldsymbol{u}, \boldsymbol{v} \rangle + c_0) \quad \text{for a constant } c_0, \gamma > 0$$

After selecting a kernel, choosing the right parameters is often a difficult task. Methods like k-fold cross validation can be used to search for them in a set of possible values. For details see Section 2.3.4.

In the nonlinear case, the cost parameter C plays an even more important role than in the linear case: A large value of C penalises observations on the wrong side of the margin heavily and therefore only a few ξ_i, if any, will be positive. This results in a small margin and a sinuous and overfit decision boundary in the original feature space. A small value of C causes a wider margin and a smoother decision boundary, as observations on the wrong side are not penalised as heavily.

2.3 Error rates

In a classification problem, normally a training and a test data set is used: the model is fit to the training data and can be evaluated with the test data, from which the correct classes are known. With this approach we can only obtain information about the performance of the model for the available data. However, an important question is how the method will perform for similar, but completely new and unseen data only available in the future. Overfitted models will lead to low error rates for the given test data but arbitrarily high error rates for new data. The aim is to find a model which returns acceptable error rates for both the test data and completely new data. Of course the term "acceptable" thereby depends, amongst other things, heavily on the problem and given facilities. Often it is a very subjective opinion of one or more indivduals involved in the task of finding an adequate classification method. In the following some common error rates are presented, and for the sake of completeness we also discuss cross validation, which is used to improve the estimated prediction error.

2.3.1 Misclassification rate (mcr)

Let N be the total number of classified obervations. Then the misclassification rate is the ratio of wrongly classified observations:

$$mcr = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\Big\{g_i \neq \widehat{G}(\boldsymbol{x_i})\Big\}$$

One can display the result in a contingency table (e.g. in R with the command `table()`) which is also called *confusion matrix*:

Predicted classes \ True classes	*positive*	*negative*
POSITIVE	t_p true positive	f_p false positive
NEGATIVE	f_n false negative	t_n true negative

Figure 2.7: Confusion matrix

For easier differentiation the true class memberships are denoted with lower case letters, namely *positive* and *negative*, and the estimated classes are denoted with upper case letters, namely *POSITIVE* and *NEGATIVE*. The notation positive and negative comes from medicine where a person is classified as "positive" when

a test result, e.g. for cancer, is positive. For this example there exist two correct classifications:

- *true positive*: a patient with cancer is correctly classified as ill
- *true negative*: a patient without cancer is correctly classified as healthy

There are two ways to make an error:

- *false positive*: a patient without cancer is wrongly classified as ill
- *false negative*: a patient with cancer is wrongly classified as healthy

With the notation of Figure 2.7 the misclassification rate can be computed as

$$mcr = \frac{f_p + f_n}{N} = \frac{f_p + f_n}{t_p + t_n + f_p + f_n}.$$

For comparing several misclassification rates of different methods, they can for example be summarised and displayed with the help of boxplots.
In the case of different numbers of observations in the classes of the training set one has to be careful as misclassification rates do not include this difference. An example makes this clear: Suppose 90% of the observations belong to the first and 10% to the second group. One possible classification rule is "assign each observation to the first group". The corresponding misclassification rate is 10%, which is a small value, however, all observations of the second group are classified wrongly. Therefore the interpretation of the misclassification rate in the case of unequal group sizes has to be made carefully.

2.3.2 Receiver Operator Characteristics (ROC)

We can extract even more information from Figure 2.7: using again the example in medicine we could be interested in how many sick patients are actually classified as ill. The higher this so-called *true positive rate* or *hit rate*, the more reliable is a positive test result. In other words: when a test has a high true positive rate, most diseases are detected. The number of wrongly as ill classified patients in those who are healthy is called *false positive rate* or *false alarm rate*. With the notation of Figure 2.7, they are calculated as follows:

- *sensitivity* or *true positive rate* (TPR)

$$TPR = \frac{t_p}{t_p + f_n}$$

- *1-specificity* or *false positive rate* (FPR)

$$FPR = \frac{f_p}{f_p + t_n}$$

The ratio 1-TPR gives us the proportion of undetected cancer patients. We could now ask ourselves: "What is worse, to be ill and the doctor does not diagnose it or to be healthy and get a wrong diagnose?". Both events cause negative effects or costs: In the worst case a wrongly negative test leads to the death of the patient, or the disease of the patient is detected later, when it has progressed and the treatment is eventually more difficult. In the case of false alarm, the wrongly positive test result leads to more examinations, e.g. a biopsy in the case of cancer, and therefore higher costs.
ROC graphs are used to visualize the performance of classifiers by plotting the FPR on the x-axis and the TPR rate on the y-axis. Additionally they are used to set the so-called *operating point*, which is the threshold for classifying an object as "positive". Subsection 2.3.2 closely follows the first six sections of Fawcett (2006), which is a good introduction to ROC theory.

Points in ROC space

Classifiers can either have a discrete (e.g. class label) or a continuous output (e.g. probability of class membership). In the case of discrete output the classification of a test set results in one confusion matrix and therefore one point in the ROC-graph. An example:

	positive	*negative*
POSITIVE	1979	209
NEGATIVE	142	1912

Table 2.1: Confusion matrix for $N = 4242$ observations.

The 45° diagonal line in Figure 2.8 stands for random classification, which means we do not have information about the classes: if a classifier allocates new observations randomly to one of the two classes with the probability $q = 0.5$, the point of this classifier in ROC space will be $(0.5, 0.5)$. In general, the resulting point will be (q, q) for the probability $q \in [0, 1]$. Each point below the diagonal signifies a classifier which performs worse than random allocation. Such a classifier has information about the classes, but it uses it wrongly. The performance can be

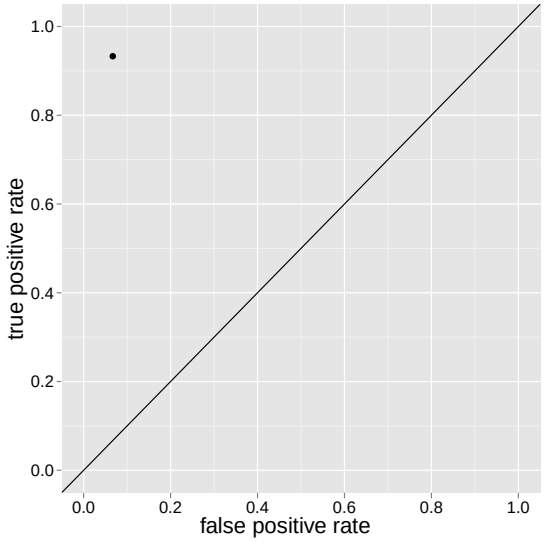


Figure 2.8: ROC point from Table 2.1

increased by negating the class assignments which results in mirroring the ROC point along the diagonal. If a point lies above the diagonal the classifier uses the class information correctly.

From now on we will concentrate on points in the upper triangle. We can further distinguish between *conservative* and *liberal* classifiers: A conservative classifier only assigns an observation to the class "positive" when strong proof is given. Therefore it will have low false positive and false negative rates: when a doctor declares a test on cancer as positive only when he is really sure, just a small amount of healthy patients will be declared as ill, but at the same time a lot of ill patients will slip through the net and do not find out that they have cancer. On the other hand, a liberal classifier classifies an observation as "positive" only with the slightest sign that it could belong to it. Therefore it will have high false positive and false negative rates: the doctor diagnoses cancer on the slightest suspicion, and so a lot of ill patients are wrongly classified as ill, but also the disease of a lot of sick patients is correctly detected and lives can be saved. For better understanding we add a conservative point in red and a liberal plot in green to Figure 2.8. The result can be seen in Figure 2.9.
The ideal case, which means all positive cases are correctly classified as positive and no negative case is wrongly classified as positive, results in the point $(0, 1)$. Two extreme points are $(0, 0)$, which means no observation is classified as positive, and $(1, 1)$, which means all points are classified as positive. A point $A = (a_1, a_2)$ is said to be better than a point $B = (b_1, b_2)$ when it lies north-west of it in the

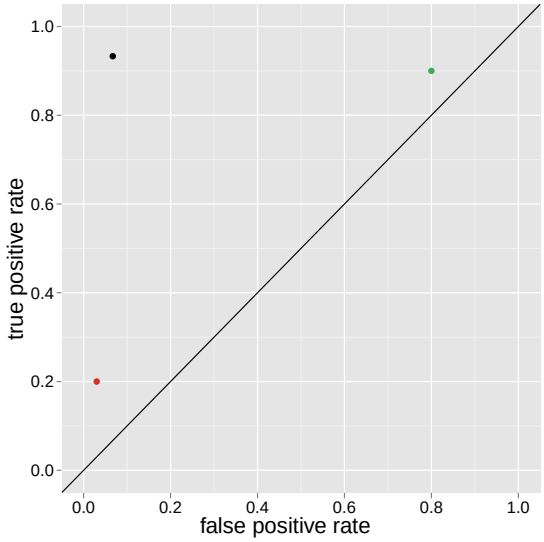


Figure 2.9: ROC graph with a conservative point in red and a liberal point in green.

ROC graph, which means that $a_1 < b_1$ (lower FPR) and $a_2 > b_2$ (higher TPR).

Creating curves in the ROC space

Often it is more informative to create curves instead of points in ROC space. When a classification method retuns a-posteriori probabilities for class membership, also called *scores*, one can determine a threshold τ for the assignment: e.g. for $\tau = 0.8$ those observations with a score bigger than 0.8 are classified as positive, all other observations as negative. Changing the threshold results in a different point in ROC space and by varying τ from $-\infty$ to ∞ we get a curve. As we only take values from one column of the confusion matrix to compute FPR and TPR, respectively, the ROC curve is independent of the underlying class distributions. That means the change of the proportion of negative to positive instancs does not have an impact on the ROC curve. By taking advantage of the monotonicity of threshold classifiers, which means that an instance which is classified as positive for a threshold τ_o will be also classified as positive for every threshold $\tau < \tau_0$, a ROC curve can be constructed as follows:

1. define the following variables:

L ... the set of test data points
N_p ... the number of positive observations in L
N_n ... the number of negative observations in L
f_p ... false positive classified objects, start value = 0
t_p ... true positive classified objects, start value = 0
$\hat{f}(\boldsymbol{x})$ ... the a-posteriori probability that $\boldsymbol{x} \in L$ is positive
R ... a matrix with the false negative rate in the first and the false positive rate in the second column

2. sort the a-posteriori probabilities returned by the classification method and the corresponding class memberships by decreasing size

3. check whether the present observation is assigned to the same class as the previous observation (for $i = 1$ it is automatically NO); if NO, add the point $\left(\frac{f_p}{N_n}, \frac{t_p}{N_p}\right)$ to the matrix R

4. check each pair of the vectors in 2.: when the instance i is classified as positive, increase t_p by 1, when the instance is classified as negative, increase f_p by 1,

5. repeat point 3. and 4. for all $i = 1, \ldots, n$

6. plot f_p versus t_p, i.e. the first column of R versus the second column of R

Point 3. is needed in the case of equally scored instances: let $(0.7, 0.7, 0.7, 0.7, 0.7)$ be a sequence of scores and the corresponding classes are two positive and three negative. As the scores are equal the order of the observations is not fixed - the two extreme cases are $(positive, positive, negative, negative, negative)$ and $(negative, negative, negative, positive, positive)$ which lead to L-shapes forming a rectangle. Any other order leads to a path in this rectangle and as we want the expected performance of a classifier we choose the average value of these paths, which is the diagonal. In Figure 2.10 an example ROC curve is shown.

Setting the operating point

In general, wrong decisions create costs. The optimal operating point represents the best trade-off between the cost of failing to detect a positive observation and the cost of false alarm. Assuming equal costs and equal a-priori possibilities of

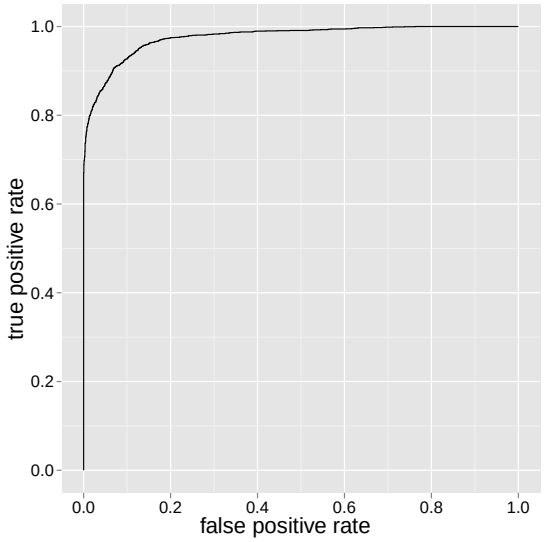


Figure 2.10: An example of a ROC curve.

class membership, normally the operating point is set to 0.5 without further computation. One can also choose the operating point by using the ROC curve: as the point $(0, 1)$ is the ideal case – high true positive rate, low false positive rate – for example the threshold resulting in the point on the ROC curve nearest to the upper left corner can be chosen as optimal cut-off point. One can also choose approximately the value where the slope of the ROC curve gets flatter, as this visualises only small gain of true positive cases and more true negative cases.

2.3.3 k-fold Cross-Validation (CV)

In order to compute the *expected prediction error* of a test data set we use a so-called *loss function* L, which measures the error between the true class g and the estimated class $\hat{G}(\boldsymbol{x})$. Examples are

- *quadratic loss*:
$$L\left(g, \hat{G}(\boldsymbol{x})\right) = \left(g - \hat{G}(\boldsymbol{x})\right)^2$$
- *absolute loss*:
$$L\left(g, \hat{G}(\boldsymbol{x})\right) = |g - \hat{G}(\boldsymbol{x})|$$

The *expected prediction error* is then defined as

$$Err = \mathbb{E}\left[L\left(g, \hat{G}(\boldsymbol{x})\right)\right]. \tag{2.47}$$

For a sample $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, (2.47) can be estimated by

$$\widehat{Err} = \frac{1}{n} \sum_{i=1}^{n} L\left(g_i, \hat{G}(\boldsymbol{x}_i)\right).$$

Often there is not enough data available to use some of it as test data. When fitting a model to a data set and testing it with the same data, the resulting prediction error is in general too optimistic, which means it is smaller than the true value. We also say the error estimate is *biased*. A popular method to obtain a more realistic esimate of the prediction error is to randomly divide the data in k more or less equally sized parts, use one part as test data and the remaining $k-1$ parts as training data. From the data set which is schematically illustrated in Figure 2.11, the third part is used as test data and the rest as training data.

1	2	3	4	...	k
train	train	test	train	...	train

Figure 2.11: An example of dividing the data in training and test data sets.

Let $\hat{G}^{(-j)}(\cdot)$ stand for the classification rule based on the data minus the j^{th} part (in our example $j = 3$), then $\hat{G}^{(-j)}(\boldsymbol{x}_i)$ is the predicted class membership of the observation $\boldsymbol{x}_i$ from the j^{th} part of the dataset. As each observation is exactly once in the test data set we get n predicted values $\hat{g}_i$ for $i = 1, \ldots, n$.
Let $\phi : \{1, \ldots, n\} \to \{1, \ldots, k\}$ be an indexing functions which indicates to which partition $j \in \{1, \ldots, k\}$ the observation $\boldsymbol{x}_i\, (i \in \{1, \ldots, n\})$ has been assigned randomly. The *cross-validation estimate* of the prediction error is then defined as

$$\widehat{Err}_{CV} = \frac{1}{n} \sum_{i=1}^{n} L\left(g_i, \hat{G}^{(-\phi(i))}(\boldsymbol{x}_i)\right),$$

The choice of k is an important task. In the case $k = n$ each partition includes just one point - this is called *leave-1-out-cross-validation*. The model is fit to $n-1$ data points and is evaluated with the left-out observation. High computational effort and high variance due to similar training data sets are often disadvantages of this method. In general 5-fold and 10-fold cross-validation are common choices which provide quite different training data sets. Big data sets are often divided according to the ratio 2:1, which means $\frac{2}{3}$ of the data are used for training and $\frac{1}{3}$ for testing. Repeating this procedure for example 100 times provides estimates of the prediction error.

Chapter 3

Statistical Analysis of Multispectral Data

3.1 Multispectral data collection

A pixel is the smallest controllable unit of a digital image. Often a pixel is a square, but it can also be a dot, a line or a similar object. A picture is built up of a certain amount of adjoining pixels, and their values can be displayed by a matrix. The more pixels an image has, the finer grain it has. When the pixels of an image can be seen we talk about a "pixelated" image. An example of a pixel image is presented in Figure 3.1: on the right side the pixel image of the three-dimensional matrix, which is displayed on the left side, can be seen. This image has been generated in Matlab (for more information see Appendix A.1) with the function `imagesc()`, which matches the values of the matrix to a colour scale.

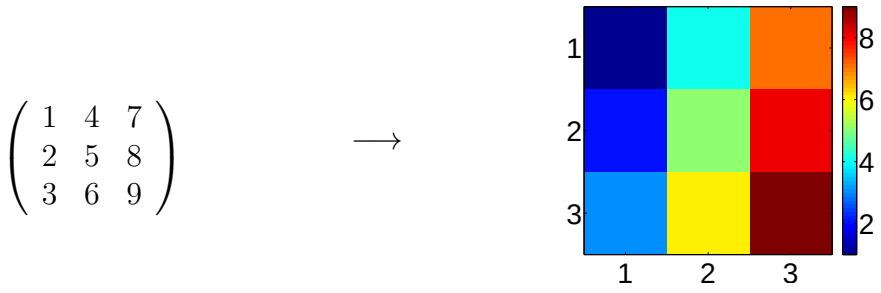


Figure 3.1: An example of a digital image.

Spectral Imaging incorporates *spectroscopy*, which is the analysis of the interaction between radiated energy and an object, and *digital imaging processing*. In a classi-

cal RGB picture, each pixel is represented by a three-dimensional vector: the first entry stands for the red, the second for the green and the third for the blue colour channel. In a multispectral image, each pixel is represented by a p-dimensional vector: each entry stands for the reflectance value at a certain wavelength of light. Often multispectral images are represented as a "data cube" like in Figure 3.2. Additionally, the spectrum of each pixel can be modeled in two dimensions by plotting the wavelength of the energy on the x-axis and its reflectance value on the y-axis. An example of a spectrum can also be seen in Figure 3.2.

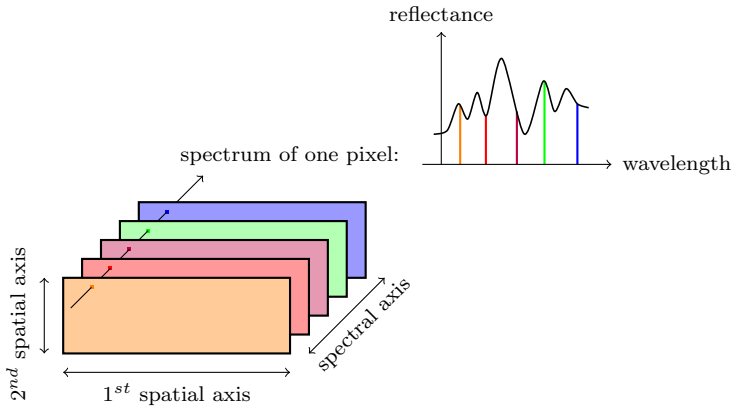


Figure 3.2: Schematic representation of a multispectral image.

The data for the statistical analysis on which this thesis is based has been collected using the so-called *wavelength scanning principle*, which means a single image has been recorded for each wavelength. As the sample is kept fixed while taking the images, this method is also known as *staring imager*. For more information see for example Leitner et al. (2003).
In our case, the sample is illuminated with LEDs of specific wavelengths, which have been previously selected by Principal Component Analysis (PCA). The recorded images are then combined in the computer to a data cube with two spatial and one spectral dimension, see Figure 3.2. For each wavelength, the recorded image is a data matrix with m_1 lines and m_2 rows. After the preprocessing, which will be discussed later, the data can be reshaped to a $(m_1 \cdot m_2) \times p$ matrix by saving each matrix as a $m_1 \cdot m_2$ long vector, reading out the data columnwise. For our example in Figure 3.1, this would result in the column vector $(1, 2, 3, 4, 5, 6, 7, 8, 9)$. Later we will have nine columns, one for each wavelength. This data format is needed for the statistical analysis, especially for the supervised learning techniques: the columns represent the features, the rows (which are nothing else than the

spectra) represent the observations. One additional column containing the group membership of each spectrum will be added to the training data.

3.1.1 Description of the data

The measurement equipment has been constructed by employees of the CTR AG. It has been improved and further elaborated during the work for this thesis, from time to time evaluated by generating a new data set and analysing it. The dataset acquired in September 2011 has been used for the results presented in Section 3.3. The multispectral imaging data has nine variables, which are the nine wavelengths used in data acquisition. We are not interested in the value of the wavlengths (e.g. 520 nm), and therefore they are denoted by wavelength no. i or `wl`i for $i = 1 \ldots, n$, respectively. Each time an image has been taken at a certain wavelength, not only one but three measurements have been made and checked for differences. Analysis has shown that there are no significant differences between the measurements, and therefore only the first measurement is discussed in the remainder of this chapter. Each of the nine images has the dimension 71×401 pixels. Data originates from six objects from the same species, to which we will refer by the capital letters A to F (A stands for the first object, B for the second object and so on). For each object, data has been recorded on four different positions, which we number from 1 to 4. E2 therefore stands for object E, second position. As the most significant results were obtained from the second position, we will concentrate on it in the following presentation and discussion of the results.

3.2 Preprocessing of multispectral image data

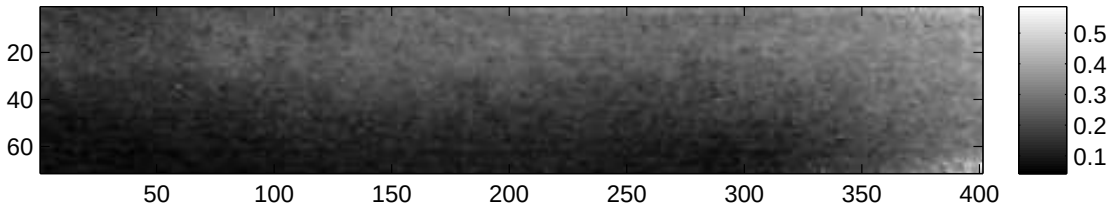
Data preparation is an important task when working with image data. The quality of the data can be improved significantly, which results in more accurate statistical models and lower error rates when predicting class memberships. In this section, the application of several standard image processing techniques is described and illustrated by pixelplots, again generated in Matlab. The aim of using these plots is to make clear which format the data have and how the preprocessing improves their quality. Therefore we do not display all wavelengths but concentrate on one object, one position, two wavelengths and data from the class "untreated". The whole preprocessing was done in Matlab, the statistical analysis, which will be discussed later, has been performed with R, see Appendix A.2.

3.2.1 Converting a colour image to a grayscale image

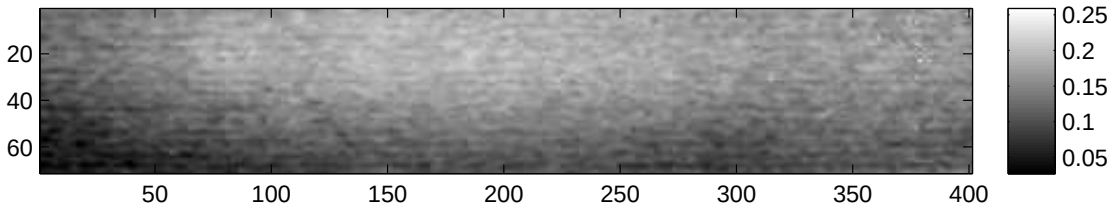
The digital images used for the analysis have been taken with a colour camera, but we only need a monochrome image. Several methods exist to convert a colour image to a grayscale image. One way is to eliminate the hue and saturation information but keep the luminance. That means instead of a 3-dimensional vector, each pixel has only one intensity value. In Matlab, the function `rgb2gray()` converts RGB values to grayscale values by computing the following weighted sum of the red (R), green (G) and blue (B) components:

$$0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

The resulting image only consists of shades of gray with the two extremes white (strongest intensity) and black (weakest intensity). Two examples are given in Figure 3.3.



(a) wavelength no. 2



(b) wavelength no. 7

Figure 3.3: Pixelplots of wavelengths no. 2 and 7 from object E at position 2, group *untreated*.

The two wavelengths have been selected due to the fact that their scatterplot (wavelength 2 on the x-axis, wavelength 7 on the y-axis) is informative, which is not the case for all wavelength pairs. We will have a look at that scatterplot later.

3.2.2 2D Gaussian lowpass filter

As our images include unwanted structural details, a filter is used to remove them. Filtering removes unwanted components by suppressing some aspect of the input signal. In our case, the input signals are the reflectance measurements made by the camera, where the filter is applied to the data of each wavelength seperately. A Gaussian filter transforms the input signal by convolution with the density of a normal (or *Gaussian*) distribution.

The filtering procedure with a Gaussian kernel for two-dimensional data is the following: The density function of the 2-dimensional normal distribution with the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$

$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

is defined as

$$\begin{aligned} f(\boldsymbol{x}) &= \frac{1}{2\pi|\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}} \\ &= \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x_1^2+x_2^2}{\sigma^2}} \qquad \forall\ \boldsymbol{x} = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}. \end{aligned} \tag{3.1}$$

In Figure 3.4, function (3.1) is displayed for the case $\sigma^2 = 1$.

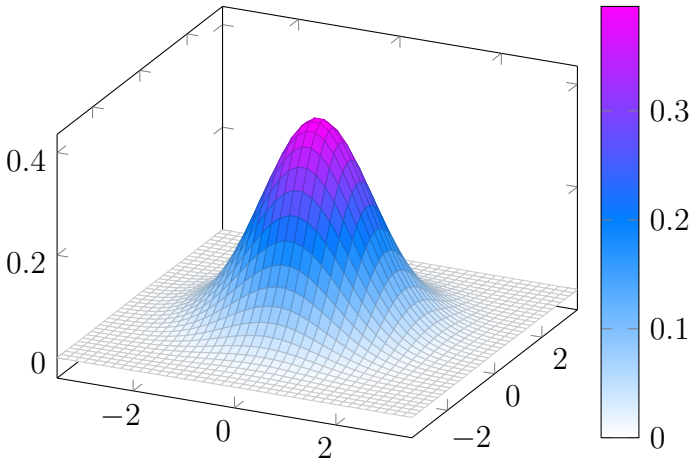


Figure 3.4: The two-dimensional density function of the normal distribution with $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}_2$.

The contour lines of the distribution function are concentric circles with the centre

$(0,0)$. As a pixel image can be visualised by a matrix and therefore consists of discrete points, we also want to approximate the density of the bivariate normal distribution by discrete values. Although the domain of the distribution function is $\mathbb{R} \times \mathbb{R}$, it is nearly zero outside the circle with $\boldsymbol{\mu}$ as centre and 3σ as radius. Therefore we can use discrete values on the square with side length $h \approx 6\sigma$ and centre $(0,0)$ as approximation. The real side length should be an odd number in order to include the pixels around the centre symmetrically. After choosing the window size h and the standard deviation σ, a $h \times h$ *convolution matrix* is calculated as discrete approximation of the density function. Table 3.1 shows a convolution matrix for Figure 3.4 with $h = 5$ and $\sigma = 1$.

$$\begin{pmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{pmatrix}$$

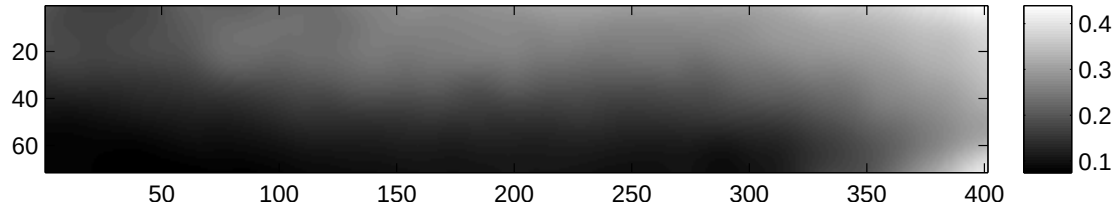
Table 3.1: Convolution matrix for $h = 5$ and $\sigma = 1$.

The new value of the pixel is the weighted average of the old value and its $h \times h$ wide neighbourhood. The values of the convolution matrix are the weights: the weight of the pixel itself is in the centre of the matrix, coloured purple in Table 3.1. The further away we get from the centre, the smaller is the weight of the corresponding pixel. In Table 3.1, this fact is visualised by the fading blue colouring of the cells. Figure 3.5 shows the pixelplots of our object after a Gaussian filter with $h = 37$ and $\sigma = 6$ has been applied. We can clearly see that the patterns which were visible before have been removed. The Gaussian filter is a so-called *blurring technique*, which means that detailed structures are removed.

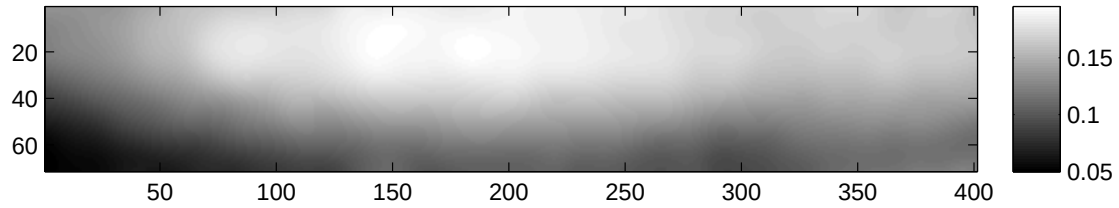
3.2.3 Two-point calibration

In order to calibrate the measuring system, a two-point calibration, which includes *dark current correction* and *system gain calibration*, is applied. Therefore a dark current image and a white diffuse reflectance standard are recorded. Let $\boldsymbol{X}_\lambda$ be the raw image, $\boldsymbol{B}_\lambda$ the dark current image and $\boldsymbol{W}_\lambda$ the white standard image at an arbitrary but fixed wavelength λ, with $\boldsymbol{X}_\lambda, \boldsymbol{B}_\lambda, \boldsymbol{W}_\lambda \in \mathbb{R}^{m_1 \times m_2}$. Let us use the notation $\boldsymbol{X} = (x_{ij})_{j=1,\dots,m_2}^{i=1,\dots,m_1}$ for each matrix. Then the final reflectance image $\boldsymbol{R}_\lambda$ is computed elementwise by

$$r_{ij\lambda} = \frac{x_{ij\lambda} - b_{ij\lambda}}{w_{ij\lambda} - b_{ij\lambda}} \quad \text{for} \begin{cases} i = 1, \dots, m_1 \\ j = 1, \dots, m_2 \end{cases} .$$



(a) wavelength no. 2



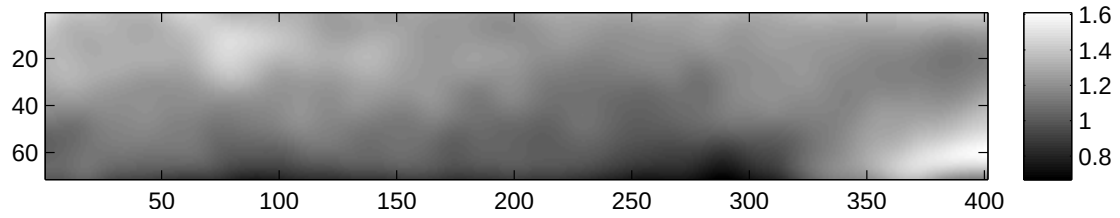
(b) wavelength no. 7

Figure 3.5: Pixelplots of wavelengths no. 2 and 7 from object E at position 2, group *untreated*, after applying a Gaussian filter with $h = 37$ and $\sigma = 6$.

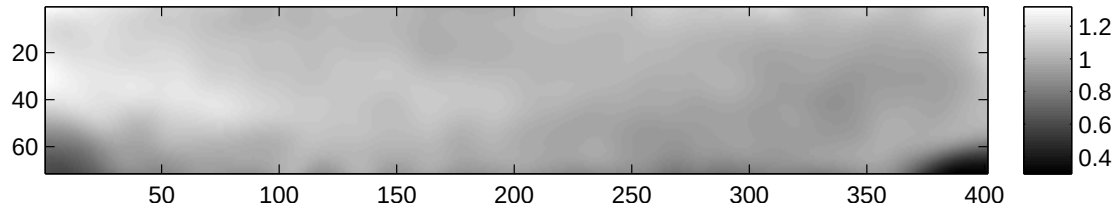
Analysis showed that B_λ is approximately zero (below detector limit and constant) in our case, therefore the raw data were only divided by the refelctance standard. Figure 3.6 shows the pixelplots of our object after the Gaussian filtering and the division by a reflectance standard. In Figure 3.5, differences in brightness can be seen: the lower half of the pictures is darker than the upper half. These differences do not occur in Figure 3.6.

3.2.4 Subset selection

Having a look at Figure 3.6, one can see that still lighter and darker spots are visible at the edges of the images. For our classification, we want to exclude these spots so that they cannot influence the results negatively. We also want to reduce the amount of data for a more efficient data analysis, as certain techniques have very long computational times for big data sets. Therefore a subset is taken by choosing a smaller rectangle out of the given images, see Figure 3.7. It seems that there are still very light and dark spots, but when having a look at the scale at the right we can see that the range of value of the subset is much smaller. The dimension of one image is now 21×101.



(a) wavelength no. 2



(b) wavelength no. 7

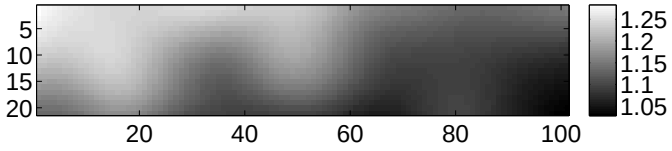
Figure 3.6: Pixelplots of wavelengths no. 2 and 7 from object E at position 2, group *untreated*, after applying a Gaussian filter with $h = 37$ and $\sigma = 6$ and dividing by a reflectance standard.

3.2.5 Image data in long format

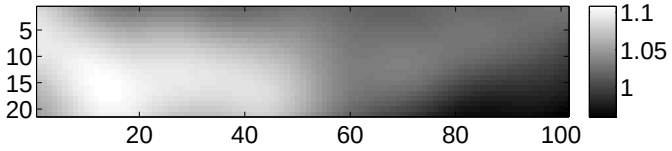
For data analysis we need the image data in the format $\mathbb{R}^{n \times p}$, with $n = m_1 \cdot m_2$ being the number of pixels per image and p the number of wavelengths used for data acquisition, which is the same as the number of images. The following code is an example how the data matrix `dat2` of the second data set, proband E at position 2, looks like in R:

```
> head(dat2[["E"]][["2"]])
   wl1  wl2  wl3  wl4  wl5  wl6  wl7  wl8  wl9     truth pixel
1 1.02 1.54 1.45 1.48 1.78 1.28 1.38 1.22 1.71 untreated     1
2 1.02 1.52 1.42 1.47 1.74 1.26 1.35 1.21 1.64 untreated     2
3 1.01 1.50 1.39 1.46 1.72 1.24 1.33 1.20 1.57 untreated     3
4 1.00 1.48 1.36 1.46 1.69 1.22 1.30 1.19 1.51 untreated     4
5 0.99 1.46 1.34 1.45 1.66 1.20 1.28 1.18 1.45 untreated     5
6 0.99 1.44 1.32 1.45 1.64 1.18 1.26 1.17 1.40 untreated     6
```

The first nine variables (`wl1-wl9`) are the input variables, the column `truth` is the output variable and the column `pixel` denotes the position where the pixel is situated in the image: the pixels in an image are numbered columnwise from 1 to n. The following code shows that exactly one half of the obervations belongs to



(a) wavelength no. 2



(b) wavelength no. 7

Figure 3.7: Subset of the pixelplots of wavelengths no. 2 and 7 from object E at position 2, group *untreated*, after applying a Gaussian filter with $h = 37$ and $\sigma = 6$ and dividing by a reflectance standard.

the class "untreated" and the other half belongs to the class "treated":

```
> E2<-dat2[["E"]][["2"]]
> dim(E2)
[1] 56942    11
> dim(E2[E2$truth=="untreated",])
[1] 28471    11
> dim(E2[E2$truth=="treated",])
[1] 28471    11
> 28471*2
[1] 56942
```

From now on the data analysis has been performed using the statistical software R.

3.2.6 Scatterplot of the image data

An informative representation of data are scatterplots. In the two-dimensional case, the values of one variable are plotted on the x-axis and those of a second variable are plotted on the y-axis. In the three-dimensional case, additionally the values of a third variable are plotted on the z-axis. In our case we have nine dimensions, but plotting them all at the same time is impossible. One possible alternative is to have a look at two variables at a time, which results in 36 scatterplots. They can be displayed as a so-called *scatterplot matrix*, but including

it in this thesis would not be very informative as space is limited. Therefore we only have a look at one scatterplot and keep in mind that it is only one of many. For the statistical analysis which has been carried out for this thesis, of course all pairs have been examined thoroughly.
As we want to have a look at the data of different objects with different value ranges, we have to find a method which makes the six data clouds more comparable and, as a result, improves classification. By columnwise centering of the data, which means that each observation is centered by the mean of the corresponding wavelength, this effect is achieved. Let $\boldsymbol{X} = (x_{ij})_{j=1,\dots,p}^{i=1,\dots,n}$ be the data matrix, then the columnwise centered matrix $\widetilde{\boldsymbol{X}} = (\tilde{x}_{ij})_{j=1,\dots,p}^{i=1,\dots,n}$ is elementwise computed by

$$\tilde{x}_{ij} = x_{ij} - \frac{1}{n}\sum_{i=1}^{n} x_{ij}.$$

Figure 3.8 shows the scatterplot of the wavelengths number 2 and 7, which have been already used for the pixelplots.

The red coloured points belong to the class "untreated", the blue coloured ones to the class "treated". The data looks quite separable according to this wavelengths, except for object F, which is denoted by the darkest shadings of red and green in Figure 3.8. It seems to have "switched" classes, which means the data cloud of object F which belongs to the class "untreated" lies within the data cloud of the objects A-E which belong to the class "treated", and vice versa.
Although two-dimensional plots do not allow for definite conclusions, the examination of the scatterplots suggests to try a linear classifier. A linear model is desired, as it implies short computation time and therefore good applicability for our hand-held device.

Creating Figure 3.8 with R

The following code shows the $n \times (p+1)$-dimensional data frame `df` which has been used for Figure 3.8:

```
> head(df)
   wl1  wl2  wl3  wl4  wl5  wl6  wl7  wl8  wl9         col
1 0.95 1.18 1.23 1.14 1.44 1.13 0.85 0.95 1.32 A untreated
2 0.95 1.17 1.23 1.14 1.45 1.13 0.86 0.95 1.33 A untreated
3 0.95 1.16 1.23 1.14 1.45 1.13 0.86 0.95 1.34 A untreated
4 0.95 1.16 1.23 1.14 1.46 1.13 0.86 0.94 1.34 A untreated
5 0.95 1.15 1.23 1.14 1.46 1.13 0.86 0.94 1.35 A untreated
6 0.94 1.14 1.23 1.14 1.46 1.12 0.87 0.94 1.35 A untreated
```

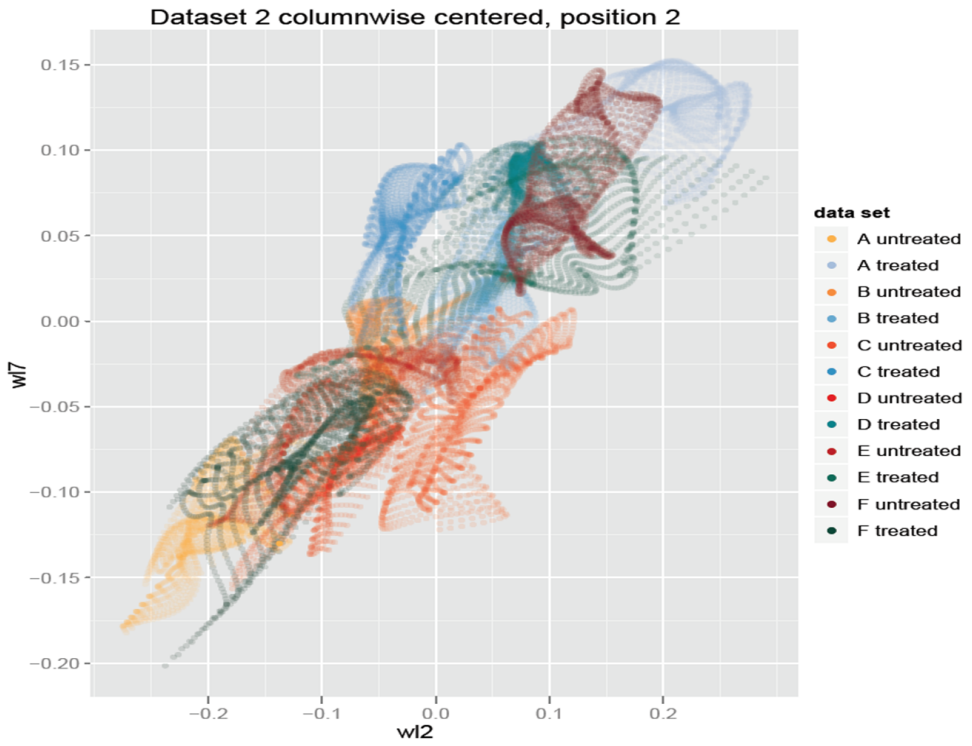


Figure 3.8: Scatterplot of columnwise centered data, position 2, wavlength 2 vs. wavelength 7.

The data frame has 10 columns: The first nine are numerical variables and stand for the wavelengths, the last one is a factor variable indicating the colours of the data clouds. In this case the combination of the object name and the class has been used, which results in 12 colours. For creating Figure 3.8, the R package `ggplot2` has been used, see Appendix A.2. The following code shows how the plot can be created:

```
> ggplot(df,aes(wl2,wl7,colour=col))+
opts(title=paste("Dataset 2 columnwise centered, position 2"))+
geom_point(aes(shape=20),alpha=.1)
```

The colours can be set by the option `scale_colour_manual()`.

3.3 Results

In the following subsections, the results of the classification methods described in Chapter 2 are presented. Like already done in Subsection 3.2.6, some R code is included after the plots in order to give an idea how they can be reproduced. For more information about the R package `ggplot2`, which has been used to produce the plots, see Wickham (2009) and A.2. The data frames for the plots are also shown, always denoted as `df`, so that the reader knows which variables are needed and which format (e.g. `numeric`, `character`) they have.
In general the classification procedure was the following: the data of five objects has been used as training data in order to get a training data set which is more independent of the individual objects. Therefore the data frames have been combined columnwise using the R function `rbind()`. During the analysis, also three or four data frames were used as training data, but the results were better for five training objects. The test data is always one proband which is not in the training set.

3.3.1 Linear Discriminant Analysis (LDA)

We want to begin with a simple model and try out more complex ones later. As all images have the same size, the amount of observations from the groups "untreated" and "treated" are equal in each training data set. Therefore the prior probabilities of the two groups are equal: $\pi_1 = \pi_2 = 0.5$. Additionally, we assume that the expected costs for misclassification are also equal: $c(2|1) = c(1|2)$. This assumption is made in order to simplify the classification and could be dropped in future analyses. In this case the classification rules of LDA and Fisher's linear discriminant are the same, as we have seen in Subsection 2.1.5. The method of Fisher has a big advantage: it provides visual insight into the separability of the data.
In order to find out whether a linear discriminant model is realistic, we have a look at so-called *Fisher plots* as a result of Fisher discriminant analysis, see Fisher (1938). That means the nine-dimensional data is projected on one dimension by looking for the optimal separation, see Section 2.1.5. A good separation of the groups in the Fisher plots suggests that misclassification rates will be low for this method. Figure 3.9 shows six Fisher plots, each for five training objects and one test object. Each plot has been made as follows: the Fisher projection direction has been calculated for five training data objects, and afterwards the data of the remaining object has been projected according to this direction. The results are displayed in one figure, where the test objects are given on the right side of each Fisher plot. One can see that the separation is good except for object F.

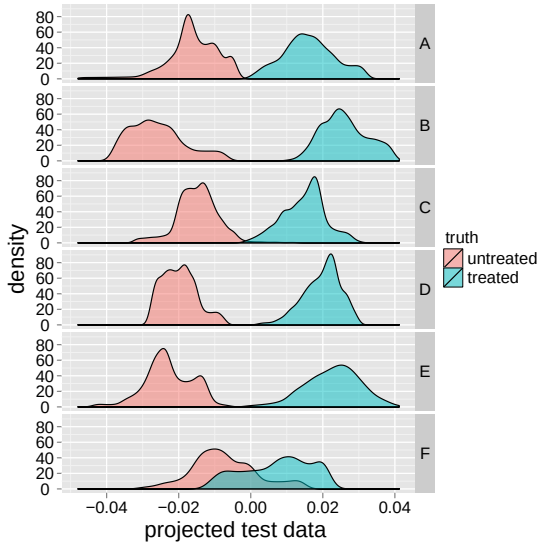


Figure 3.9: Fisherplots for the setting five training objects, one test object, at position 2.

Encouraged by these plots, we calculate a linear model using the function `lda()` from the package `MASS`, see Appendix A.2. Again the data of five objects has been used as training data and one object has been used for testing the model. The function `predict()` delivers the predicted class memberships of the test object, which are evaluated using misclassification rates, see Subsection 2.3.1. The result is displayed in Table 3.2.

	A	B	C	D	E	F
A,B,C,D,E						0.2178
A,B,C,D,F					0.0017	
A,B,C,E,F				0.0000		
A,B,D,E,F			0.0087			
A,C,D,E,F		0.0000				
B,C,D,E,F	0.0009					

Table 3.2: LDA misclassification rates for 5 training data sets and one test data set at position 2.

The objects which are used as training data are displayed in the rows, those which are used as test data in the columns. Those fields which are left blank stand for

settings which have not been computed, as the prediction error would be too optimistic when an object is in the training and in the test data set. Except for object F, which is an outlier as we have already seen in Figure 3.8, the misclassification rates are acceptable. Some values are even zero, which implies not only that the data is linearly separable, but also that the data of five objects include a lot of information about the sixth object. This fact can be used in future applications of the linear model: we do not need data of the object which has to be classified, but can use training data of a fixed set of objects.

3.3.2 Robust LDA (Linda)

Although the results of LDA are already good, we want to find a model which reduces the influence of outliers on the other objects. One possibility is to use robust methods, e.g. `Linda()`, which is an R function from the package `rrcov`, see Appendix A.2. As explained in Chapter 2, robust LDA methods use estimators of the mean and the covariance with a high breakdown point. In the case of `Linda()`, this is done by using the MCD estimator with the default value $h = 0.5$, see Subsection 2.1.4. As an effect, outliers in the data are excluded when fitting the model and therefore do not have a big effect on the result. The misclassification rates of robust LDA can be ssen in Table 3.3.

	A	B	C	D	E	F
A,B,C,D,E						0.2129
A,B,C,D,F					0.0000	
A,B,C,E,F				0.0620		
A,B,D,E,F			0.0052			
A,C,D,E,F		0.0000				
B,C,D,E,F	0.0000					

Table 3.3: Linda misclassification rates for 5 training data sets and one test data set at position 2.

`Linda()` detects the observations from object F as outliers and exludes them when modeling the decision boundary. While the error rates of the outliers may increase, which does not bother us as we want a good prediction for the "non-outlier" data, the misclassification rates of the remaining objects decrease. Another positive aspect of discriminant methods like LDA and robust LDA is, that even when the amount of data is increased by adding new training objects, computing time does not significantly increase. This is due to the fact that the dimension of the

covariance matrix only depends on the dimension of the data and therefore always stays $p \times p$, even when n, the amount of data, is high. As the estimation of the covariance matrix is the most computationally intensive part, computing time will more or less stay the same.

3.3.3 Quadratic Discriminant Analysis (QDA)

Although it seems that the data is linearly separable, we also try quadratic discriminant analysis. As it is difficult to detect every detail of the relationships between nine variables simply by examining scatterplots, there may exist some unrecognised nonlinear structures which can be better described by a quadratic model. Using the R function `qda()`, again from the package `MASS`, we therefore calculate a quadratic model with five training objects and predict the class memberships of the remaining objects. The resulting misclassification rates are displayed in Table 3.4.

	A	B	C	D	E	F
A,B,C,D,E						0.1862
A,B,C,D,F					0.0014	
A,B,C,E,F				0.0005		
A,B,D,E,F			0.0210			
A,C,D,E,F		0.0000				
B,C,D,E,F	0.0005					

Table 3.4: QDA misclassification rates for 5 training data sets and one test data set at position 2.

Quadratic Discriminant Analysis does not bring a large improvement. On the first sight it looks good: the misclassification rates are in general low, and even the value of the setting where F is the test object has decreased. Having a closer look, one can see that the misclassification rates of some settings, which worked good for LDA and robust LDA, slightly increased. This may not seem important, but for other positions, which work in general worse than position 2, the effect is more distinctive. In general, a trade-off between the error rates of outlier objects and non-outlier objects is not desired. We will see later, namely in Subsection 3.3.6, that the overall performance over all four positions is worse than LDA.

3.3.4 Robust QDA

Also for QDA, the robust version is tried out. We use the function `QdaCov()`, again from package `rrcov`, which also uses MCD estimators with $h = 0.5$ as default. The

results can be seen in Figure 3.5.

	A	B	C	D	E	F
A,B,C,D,E						0.2214
A,B,C,D,F					0.0462	
A,B,C,E,F				0.0924		
A,B,D,E,F			0.0328			
A,C,D,E,F		0.0000				
B,C,D,E,F	0.0000					

Table 3.5: QdaCov misclassification rates for 5 training data sets and one test data set at position 2.

Robust QDA performs even worse than the classical QDA, which shows us that the quadratic approach is not suitable for our data. This is good for us, as the linear models are simpler and easier applicable in reality.

3.3.5 Support Vector Machine (SVM)

SVM is a powerful classification method often used in image analysis. As LDA and QDA misclassification results were not convincing at the beginning of the statistical analysis performed for this thesis, SVM has been included. Therefor the function `svm()` from the package `e1071` has been used, see Appendix A.2. After adapting the preprocessing, LDA and QDA results were better than those of SVM and it came out that SVM is not a suitable method for our data as it cannot handle the outlier object. Another big disadvantage of the SVM method is that computation time is high and increases for increasing size of the training data set. Using SVM always includes the decision which kernel should be used. In accordance with the linear separability which can be seen in Figure 3.9, and in order to keep the model simple, the *linear kernel* has been chosen. Additionally the *radial basis kernel* has been used as it is the default kernel and we want to try out whether a more complicated nonlinear approach is useful, which was also one motive to use QDA. Setting the parameters of the radial kernel is a very sensitive exercise and as we have six test settings, we had to choose one and adapt the parameters for it. In the end, object A at position 2 has been used for parameter selection. The function `svmEval()` from the package `chemometrics` is helpful for this task. The following procedure has been used to determine the optimal parameters: C has been fixed at its default value 1, because analysis has shown that changing this parameter does not lead to an error improvement. Then that value has been

chosen from 10 possible γ-values, for which the cross-validation error is smallest. 2/3 of the data has been randomly chosen as training data and the other third has been used as test data. This procedure has been repeated 100 times and the result has been plotted as bar chart. In the end, the paramters $C = 1$ and $\gamma = 0.01$ have been chosen as 0.01 had the highest frequency.

The misclassification errors of SVM with the linear kernel can be seen in Table 3.6.

	A	B	C	D	E	F
A,B,C,D,E						0.7937
A,B,C,D,F					0.0000	
A,B,C,E,F				0.0085		
A,B,D,E,F			0.0879			
A,C,D,E,F		0.0000				
B,C,D,E,F	0.0000					

Table 3.6: SVM (linear kernel) misclassification rates for 5 training data sets and one test data set at position 2.

The result of the outlier object F is very bad, as nearly 80% of its obersvations are classified wrongly. This may be due to overfitting, which means the SVM model does not fit to the data of object F because it adapts too much to the information from the training data. Also the misclassification error of object C is higher than for any other method.

The misclassification errors of SVM with the radial basis kernel can be seen in Table 3.7.

	A	B	C	D	E	F
A,B,C,D,E						0.8347
A,B,C,D,F					0.0000	
A,B,C,E,F				0.0120		
A,B,D,E,F			0.0632			
A,C,D,E,F		0.0000				
B,C,D,E,F	0.0050					

Table 3.7: SVM (radial basis kernel with parameters C=1 and $\gamma = 0.01$) misclassification rates for 5 training data sets and one test data set at position 2.

The misclassification rate of object F is higher than 80%, which shows that the

information provided by the training objects cannot be used to predict the class memberships of the test object for this setting. The other results are moderate, but we will see later that this method does not work for our data at all. Even when the results would be better, the high computation time would make it impossible to use SVM for a hand-held device efficiently.

3.3.6 Selection of the optimal classification method

In order to find the optimal method for the classification of our image data, we want to compare the results of LDA, robust LDA, QDA, robust QDA and SVM, both using the linear and the radial basis kernel, at all four positions. As plotting is a useful way to get an overview, two graphical representations are presented in this subsection.
At first the following list should remind the reader which properties an optimal classifier should have for our problem:

- *Moderate computing time*: the method will be implemented in a hand-held device, therefore computing time should not be too long so that the comsumer does not have to wait.
- *Easily adaptable for new objects*: the project is still running and new data sets will be created, therefore we are looking for a method where we can be quite sure that it will also work for new data, both of the objects already included in the current analysis and of new, unseen objects.
- *Good results for non-outlier objects*: the main attention lies on the correct classification of non-oultier objects. Lower misclassification rates of outlier objects should not result in a trade-off with higher rates of the non-outlier objects.

Boxplot visualisation

The first visualisation method used is the boxplot, which is a graphical implementation of the so-called *five-number summary*, consisting of five quantiles: the minimum, the first quartile, the median, the third quartile and the maximum of a dataset. In Figure 3.10 one can see how a boxplot is structured. The scale on the left side has been chosen as we want to display misclassification rates, which lie between 0 and 1. Of course any other scale is possible. A boxplot is constructed in the following way:

- The **box** is determined by two values: its lower line is the 1^{st} quartile, which is the 0.25-quantile, and its upper line is the 3^{rd} quartile, which is the

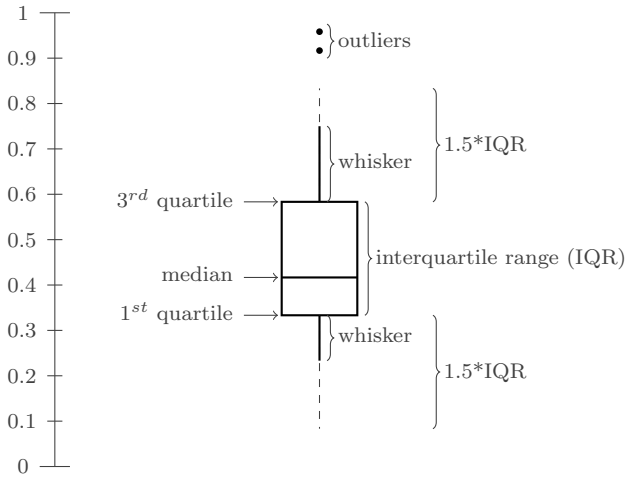


Figure 3.10: Schematic representation of a boxplot.

0.75-quantile. These two lines, which are also called lower and upper *hinge*, can also be determined by alternative methods, which will not be covered in this thesis. However, when using statisical software, one should always consider which method is used in order to interpret the boxplot correctly. The **boxlength**, also known as the *interquartile range* (IQR), is a robust measure of variability. Using the term lower and upper hinge, the boxlength is sometimes referred to as *H-spread*.

- The **median** is a robust estimation of the mean value. It is the 0.5-quantile and therefore situated within the box, visualised by a straight line. The location of the median in the box is a measure of symmetry of the underlying distribution. The boxplot in Figure 3.10 is for example right skewed.

- The **whiskers** are drawn from the end of the box until the last observation which lies within the *inner fence*, which is defined as 1.5*IQR. In Figure 3.10 one can see that the whiskers are shorter than 1.5*IQR, which is drawn as dashed line.

- The **outliers** are alle points which lie outside the inner fence.

Figure 3.11 shows six boxplots of misclassification rates, one for each method. Again the data of five objects has been used as training data and the remaining object was used as test data set, which results in six models. As we have four positions, this results in 24 misclassification rates per boxplot.

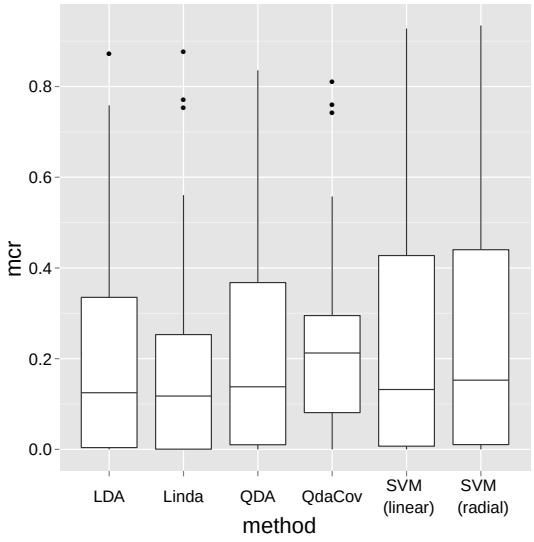


Figure 3.11: Boxplots of all objects and positions.

As we can see, the box lengths of the robust methods are shorter than those of the other methods. That means the variability of these methods is lower, and therefore we can expect that future images will have similar misclassifiation rates, provided that they are no outliers. Also the whiskers are shorter than those of the non robust methods, although all distributions are right skewed. The robust methods detect the outlier objects and exclude them from model fitting, which results in high misclassification rates of the outlier objects and low misclassification rates of the non-outlier objects. The SVM methods do not recognise any outliers and as a result, the overall performance is bad. The median of robust LDA (Linda) is slightly the lowest, but what is even more important is that the box length is short, which indicates that this method will be stable with respect to new observations, which means data from new objects. According to Figure 3.11, robust LDA threfore seems to be the optimal classification method for our problem. As the computation time of the function `Linda()` is short, our requirements are fulfilled.

Creating Figure 3.11 with R

The following code shows the first six lines of the data frame used for this figure:

```
> head(df)
  method          mcr
1    LDA 0.0047147572
```

```
2    LDA 0.0009429514
3    LDA 0.8719943423
4    LDA 0.6117397454
5    LDA 0.0478547855
6    LDA 0.0000000000
```

The column `method` includes the information to which classifier the misclassification rate in the column `mcr` belongs. `method` is a factor variable with the following levels:

```
> levels(df$method))
[1] "LDA"              "Linda"           "QDA"              "QdaCov"
[5] "SVM \n (linear)" "SVM \n (radial)"
```

The character string "`\n`" ensures that the strings for SVM include line breaks. The code to produce the figure itself is the following:

```
> ggplot(df,aes(method,mcr))+geom_boxplot()
```

ROC visualisation

The second visualisation method used are ROC curves, see Subsection 2.3.2. Let us have a look at four ROC plots, each containing six curves, one for each object. The method used for creating them is `Linda()`, since according to Figure 3.11 this is the best method. Figure 3.12 shows the four ROC plots.

The object in the legend are the test objects, the remaining objects have been used as training data. We can see that position 1 and 2 work quite good, whereas position 3 and 4 seem to work worse. The images at the first two positions have been taken at similar locations and the second two have been taken at two similar locations, respectively. That is the reason why the results resemble each other pairwise. When excluding object F as an outlier, the first two positions work very good. In the case of position 3, object B and F use the information of the other five objects wrongly. The first two positions are considered as more important as it is unlikely that the hand-held device is used for positions 3 and 4. On the whole, Figure 3.12 confirms that robust LDA is a suitable classification method for us.

Creating Figure 3.12 with R

The ROC curves have been created using the algorithm described in Subsection 2.3.2. The following code shows the first six lines of the data frame used for plotting at position 2. Object F has been selected as most of the values of the other objects are `NA`. This is because the matrix R in the algorithm has been

defined as a matrix with all entries set to `NA` and due to point 3., only a few points are added to the matrix.

```
> df<-subset(df,object=="F")
> head(df)
      FPR          TPR object
21216   0 0.0000000000      F
21217   0 0.0004714757      F
21218   0 0.0009429514      F
21219   0 0.0014144272      F
21220   0 0.0018859029      F
21221   0 0.0023573786      F
```

The row numbers are so high as object F is the last of the six objects in the data frame. The column `FPR` stands for the false positive rate, the column `TPR` stands for the true positive rate and the column `object` indicates to which object the values belong. This information is needed to set the colour and the linetype of the curves. The code which produces Figure 3.12(b) is the following:

```
> ggplot(plotdat4,aes(x=FPR,y=TPR))+
geom_line(aes(colour=object,linetype=object),size=1.5)+
geom_abline(aes(0,1),size=.8)
```

3.3.7 Majority voting

After choosing `Linda()` as optimal classification method, we want to classify the image as a whole. Until now we classified each pixel and then checked, by using misclassification rates, how many of them have been wrongly classified. When using the hand-held device one should only get the information whether the image which has been taken is "untreated" or "treated". That means we only want to have one classification per image instead of each pixel in the end. Additionally, we are interested in whether certain parts of the picture are classified wrongly more often than others. Therefore the data sets are vertically divided into five parts according to the drawing in Figure 3.13.

Again, the data of five objects has been used as training data and the remaining object has been used for testing, but the test procedure has been slightly modified. Two approaches have been tried out during the analysis:

- **Majority voting**: All pixels of an image of one object have been classified. Afterwards, the class is determined for each part by majority voting: the amount of pixels classified as "untreated" and those classified as "treated"

has been determined, respectively. When more than 50% of the pixels are classified as "untreated", the corresponding part is classified as "untreated", otherwise it is classified as "treated". Figure 3.14 shows the result of this approach for position 2. All parts have been correctly classified for each object, even for the outlier F. In this plot, the length of the bars is the percentage of pixels which determined the predicted class. Having a closer look at 3.14, we can see that in the case of object F, the decision for the right group was not made with the same high percentages as for the other objects. Nevertheless this method even works for the outlier object.

Creating Figure 3.14 with R

The following code shows the first ten lines of the data frame which has been used to create Figure 3.15:

```
> df[1:10,]
   part  object     truth prediction   percent
1     1  test=A untreated       true 1.0000000
2     1  test=B untreated       true 1.0000000
3     1  test=C untreated       true 0.9694118
4     1  test=D untreated       true 0.7623529
5     1  test=E untreated       true 1.0000000
6     1  test=F untreated       true 0.5670588
7     2  test=A untreated       true 1.0000000
8     2  test=B untreated       true 1.0000000
9     2  test=C untreated       true 1.0000000
10    2  test=D untreated       true 1.0000000
```

The data frame has five variables: the column `part` contains the number of the part of the image to which the corresponding observation belongs, the column `object` contains the corresponding test object of the classification, `truth` denotes the true class of the observation, `prediction` denotes whether the predicted class is true or false and `percent` contains the amount of pixels of the corresponding part which have been assigned to the predicted class. The code for creating Figure 3.14 is the following:

```
> ggplot(df)+
    geom_bar(aes(x=factor(prediction),fill=factor(part),
         weight=percent))+
    coord_flip()+facet_grid(object~truth)
```

- **Classification of median per part**: The median of the pixels is computed for each part, which results in five values per image. These values are then classified, resulting in five predicted values. Figure 3.15 shows the result of this approach for position 2. In this case, the result for object F is not good. Only three out of five median values have been classified correctly. For the other objects, also this approach works satisfyingly.

  Creating Figure 3.15 with R

  The data frame has the same variables as before, except for the column `percent`, which is no longer needed as only one pixel is classified per part. The code for creating Figure 3.15 is the following:

```
> ggplot(plotdat)+
     geom_bar(aes(x=factor(prediction),fill=factor(part))+
     coord_flip()+facet_grid(object~truth)
```

Although the results for the two approches are good, we should not forget that position 2 works best. In the case of the other positions, more wrongly classified parts occur.
Having reduced the amount of predicted values to five, we can now classify the whole picture. Therefor again majority voting is applied: when at least three parts are classified as "untreated" the whole image is classified as "untreated", otherwise it is classified as "treated". According to this rule, all images are correctly classified at position 2. Additionally, no serious spatial variation has been found, which means that the applied preprocessing sucessfully removes all disturbing effects which occur during image acquisition.

3.4 Prospects

Some possible modifications could be made for future analysis:

- For the sake of simplicity, no distinction between the type of the objects is made in this thesis, although there exist several degrees of brightness, which could be included in future analysis. Although we have seen that linear decision boundaries should be sufficient, more complicated methods like alternative kernels for SVM or nonlinear decision boundaries could be tried out. As the project is still running and new data sets of new objects are made, it could happen that linear classifiers are not sufficient anymore.

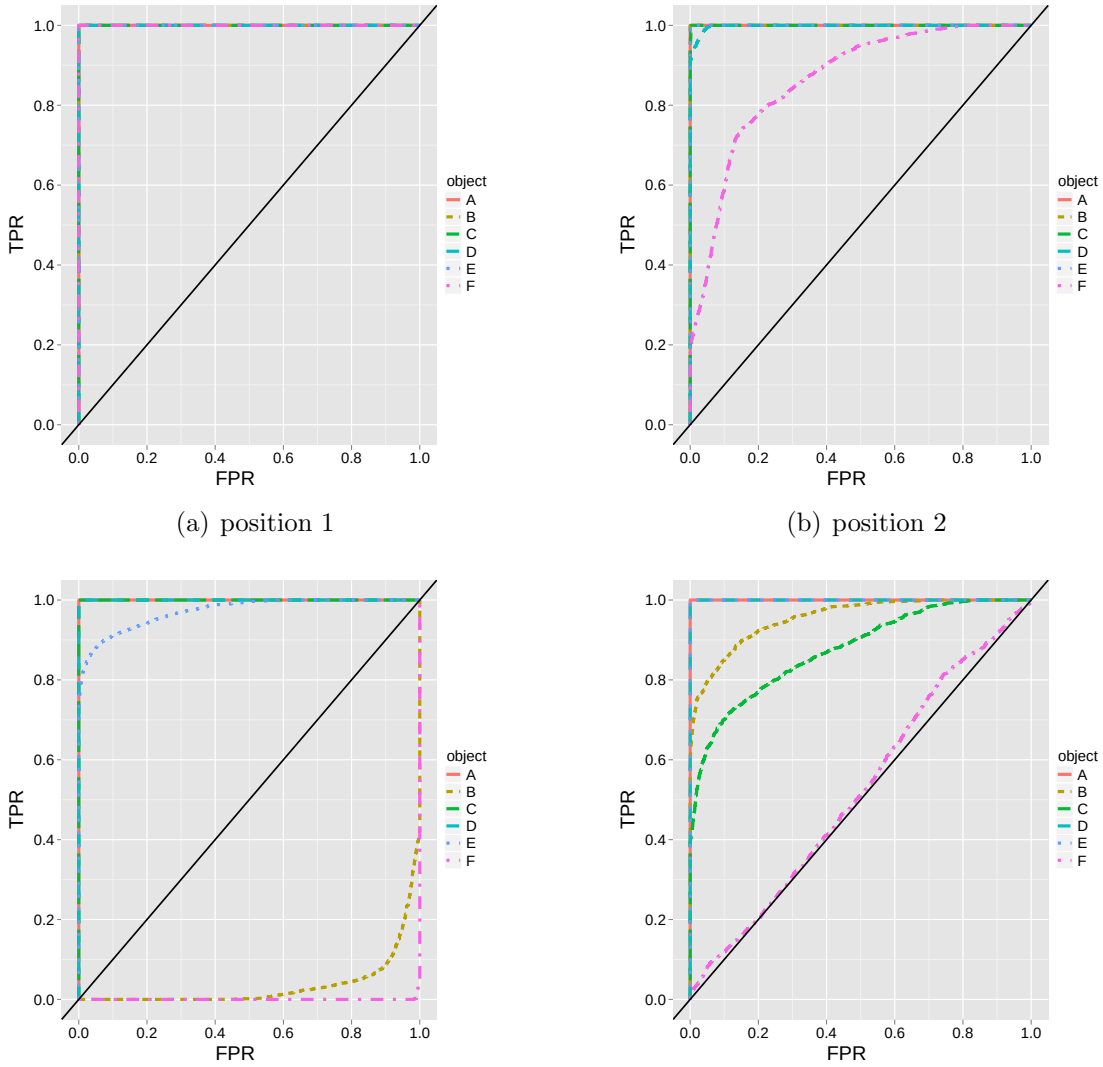
- The results presented in Chapter 3 are for a subset of the actual image. In future, the camera could be improved in order to avoid unwanted incidence of light and use the whole image. As that would imply a siginificant increase of the amount of training data, the methods selected should not be sensitive to the size of the data. Computationally intensive methods like SVM may have to be excluded completely.

- The assumption $c(2|1) = c(1|2)$ could be dropped and replaced by either $c(2|1) > c(1|2)$, which means not detecting an untreated position is worse than not detecting a treated position, or $c(2|1) < c(1|2)$, which means the opposite.

- One could try out alternative error rates.

- During the analysis for this thesis, different kinds of normalisations, like L_1 and L_2, and transformations, like isometric log ratio (ilr), have been tried out in order to improve the classifcation results. Although it was not possible to achieve a considerable improvement with any of these normalisation approaches, they should be kept in mind for future datasets.

- Spatial information could be included.



(a) position 1

(b) position 2

(c) position 3

(d) position 4

Figure 3.12: ROC curves of robust LDA.

1	2	3	4	5

Figure 3.13: Schematic representation of dividing an image into five parts.

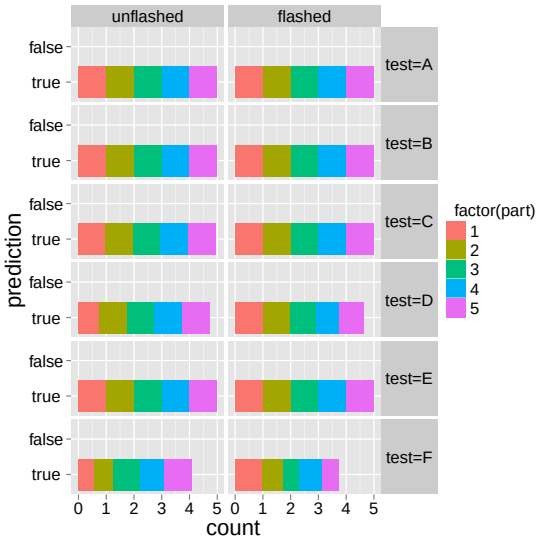


Figure 3.14: Error plots of the majority voting, position 2.

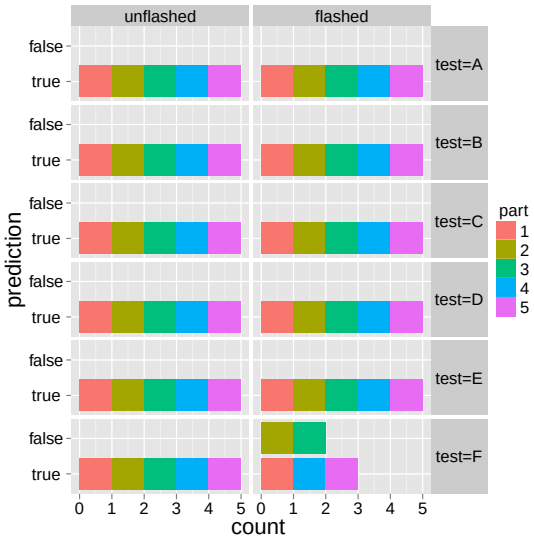


Figure 3.15: Error plots of the classification of the median per part, position 2.

Chapter 4

Conclusion

> The important criterion for a graph is not simply how fast we can see a result; rather it is whether through the use of the graph we can see something that would have been harder to see otherwise or that could not have been seen at all.
>
> William Cleveland
> *The Elements of Graphing Data*

After applying the supervised learning techniques LDA, QDA and SVM to our data, it showed that robust LDA works best. It is also suitable for a hand-held device as computing time is low and new outlier data do not have a big influence on the model.
Let us have a look at the performance of robust LDA for all objects and positions. Therefore we use the results of both methods presented in Subsection 3.3.7, which leads to four possible outcomes: a part is wrongly classified for both methods, it is correctly classified for both methods or it is correctly classified for one and wrongly classified for the other method (two possible scenarios). In order to display the results for all objects in one plot, each part is filled with six colours, each standing for one object. The data are not parted, it is only a method to visualise the results of six objects in one plot in order to save space and and give an overview of the results. Figure 4.1 shows where the results of the objects are positioned. Figure 4.2 shows the results of the classification per part described in Subsection 3.3.7 at all four positions and for all probands.
It consists of four plots, one for each position, including one representation of an image of the class "untreated" and one of the class "treated". Each image is divided in five parts, which are positioned with spaces between them to increase clarity like it is shown in Figure 4.1. Each of the six rectangles of a part, which are

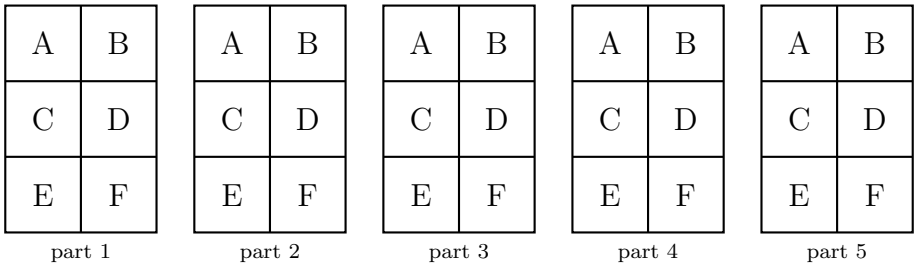
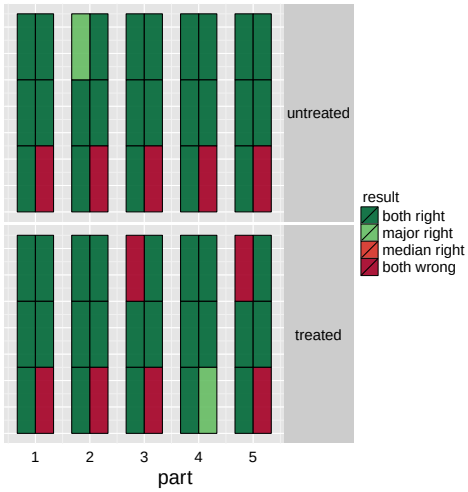
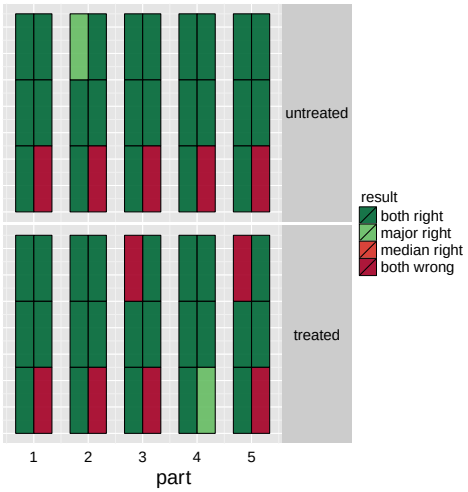


Figure 4.1: Schematic plot of positioning the results for all objects in one plot.

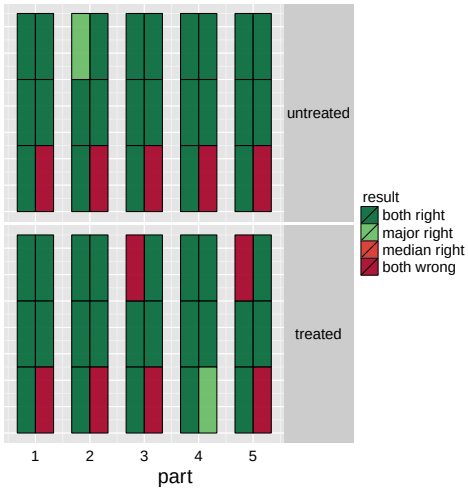
arranged according to Figure 4.1, is filled with one out of the four colours dark red, light red, light green and dark green, depending on the outcome of the two majority votings. The legends on the right side of the subplots explain which colour belongs to which outcome. "major right" thereby refers to the correct majority voting after the classification of all pixels, "median right" refers to the correct classification of the median of the corresponding part. The following description how the colour of one cell is chosen should clarify the information in the plot: Having a look at Figure 3.14 and Figure 3.15, which show the results of the majority voting at position 2, we can combine the result of one test object at one part for one one class, e.g. object A, part one, class "unflashed". For the test object A at part one group "unflashed", the results of both majority voting scenarios are correct, and therefore the corresponding field is dark green. This can be seen in Figure 4.2(b) in the upper half of the plot, where the first rectangle in the upper left corner is dark green.
We can see that for the objects B, C, D and E the majority votings work for all positions and both groups, which means all images of these objects are correctly classified. In general, the performance of robust LDA for the object F is bad at all positions and for all methods. Also object A has wrong results when only the median is classified, but when classifiying all objects at each part and then selecting the class of this part with the help of majority voting, the result is always right.
Results can certainly be improved in future by improving data quality. As already mentioned in Subsection 3.4, new ideas will be tried out and data analysis will be continued with new datasets. The analysis of this multispectral data certainly provides a wide range of interesting approaches and would bear more material than is included in this thesis.



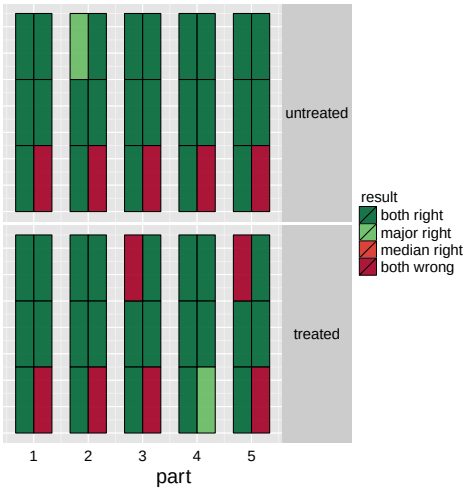
(a) position 1



(b) position 2



(c) position 3



(d) position 4

Figure 4.2: Combined errors of both methods for majority voting from Subsection 3.3.7.

Appendix A

Software

A.1 Matlab

For data acquisition and preprocessing, the mathematical software MATLAB® & Simulink® (Student Version R2009a, The MathWorks™) has been used. Useful toolboxes for the purpose of this thesis have been the *Image Processing Toolbox* which includes the functions `fspecial()` and `imfilter()` for the Gaussian filter described in Subsection 3.2.2. The *Statistics Toolbox* includes statistical functions. Helpful functions for processing and analysing images have been `squeeze()`, with which one can remove singleton dimensions of a dataset, `rgb2gray()`, which converts a colour image to a grayscale image, `fspecial()`, which creates a filter, `imfilter()`, which is used to apply a filter, and `imagesc()`, which is used to display an image.
`http://www.mathworks.de/help/techdoc` is a helpful website for new Matlab users.

A.2 R

The statistical analysis for this thesis has been made using the open source statistical software R, see R Development Core Team (2011). A general explanation of R is given by the following quotation:

> *[R is] an environment within which statistical techniques are implemented. [...] The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.*[1]

[1] `http://www.r-project.org`: What is R?

Packages include extensions of the main R functions regarding special statistical topics. In the following, an alphabetically sorted list of the favourite packages of the author is given, where all of them have been used for this thesis:

- `chemometrics`: Includes functions for data analysis in chemometrics. The function `svmEval` has been used to determine the SVM parameters. The vignette "Multivariate Statistical Analysis using chemometrics" contains useful information about the application of classification methods, especially SVM. For more information see Filzmoser and Varmuza (2011).

- `e1071`: SVM is implemented in several R-packages, e.g. package `e1071` function `svm()` (based on `LIBSVM`). For more information see Dimitriadou et al. (2011).

- `ggplot2`: The R package `ggplot2` is a powerful and multifunctional tool to produce meaningful plots. The book Wickham (2009) is a good introduction, although the reader should already have some experience using R. All of the R plots used in this thesis have been made using `ggplot2`, see the R code included at the end of most of the subsections in Chapter 3.

- `MASS`: One of the standard R packages, including the two functions `lda()` and `qda()`, which are used in this thesis. The book Venables and Ripley (2002) is a helpful support when working with R and gives insight into the programming of the functions implemented in the `MASS` package.

- `plyr`: This package is very useful when working with lists and dataframes, especially when one has as many categories as our data (objects, positions, parts etc.). Data can be quickly restructured and functions can be easily applied to more than one data frame. A good introduction to the methods of this package is provided by Wickham (2011).

- `RColorBrewer`: Colours are an important part of an expressive plot. This package, which provides numerous colour palettes, povides an elegant an quick way to choose appropriate plot colours. For more information see Neuwirth (2011).

- `reshape`: Provides useful functions for restructuring and aggregating data, see Wickham (2007).

- `rrcov`: In R, robust discriminant analysis is for example implemented in the package `rrcov`. The robust version of LDA is called `Linda()`, those of QDA is called `QdaCov()`. The default estimation method is MCD and the parameter h can be set by `alpha`. The article corresponing to the package is Todorov and Filzmoser (2009).

- `robCompositions`: Provides routines for compositional data, see Hron et al. (2010).
- `xtable`: Useful for converting matrices in R to LaTeX tables, see Dahl (2011).

Bibliography

D.B. Dahl. *xtable: Export tables to LaTeX or HTML*, 2011. URL `http://CRAN.R-project.org/package=xtable`. R package version 1.6-0.

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2011. URL `http://CRAN.R-project.org/package=e1071`. R package version 1.6.

D.L. Donoho and P.J. Huber. The notion of breakdown point. In P. Bickel, K. Doksum, and J.L. Hodges Jr., editors, *A Festschrift for Erich Lehmann*, pages 157–184. Wadsworth Pub. Co, Belmont, CA, 1983.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8): 861–874, 2006.

P. Filzmoser and K. Varmuza. *chemometrics: Multivariate Statistical Analysis in Chemometrics*, 2011. URL `http://CRAN.R-project.org/package=chemometrics`. R package version 1.3.7.

R.A. Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8:376–386, 1938.

F.R. Hampel. A general qualitative definition of robustness. *Annals of Mathematical Statistics*, 42:1887–1896, 1971.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2^{nd} edition, 2009. URL `http://www-stat.stanford.edu/~tibs/ElemStatLearn`.

K. Hron, M. Templ, and P. Filzmoser. Imputation of missing values for compositional data using classical and robust methods. *Computational Statistics and Data Analysis*, 54:3095–3107, 2010.

R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, London, 6^{th} edition, 2007.

R. Leitner, H. Mairer, and A. Kercek. Real-time classification of polymers with NIR spectral imaging and blob analysis. *Real-Time Imaging*, 54(4):245–251, 2003.

E. Neuwirth. *RColorBrewer: ColorBrewer palettes*, 2011. URL `http://CRAN.R-project.org/package=RColorBrewer`. R package version 1.0-5.

T. Oetiker, H. Partl, I. Hyna, and E. Schlegl. *The Not So Short Introduction to LaTeX 2ε*, 2011. URL `http://tobi.oetiker.ch/lshort/lshort.pdf`.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL `http://www.R-project.org/`.

P.J. Rousseeuw. Multivariate estimation with high breakdown point. In W. Grossmann, G.Pflug, I.Vincze, and W.Wertz., editors, *Mathematical Statistics and Applications*, volume B, pages 283–297. Akadémiai Kiadó, Budapest, 1985.

T. Tantau. *TikZ & PGF: Manual for Version 2.10*, 2007. URL `http://sourceforge.net/projects/pgf`.

V. Todorov and P. Filzmoser. An object-oriented framework for robust multivariate analysis. *Journal of Statistical Software*, 32(3):1–47, 2009. URL `http://www.jstatsoft.org/v32/i03`.

W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S*. Springer, New York, 4^{th} edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`.

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL `http://www.jstatsoft.org/v21/i12`.

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, New York, 2009. URL `http://had.co.nz/ggplot2/book`.

H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1):1–29, 2011. URL `http://www.jstatsoft.org/v40/i01`.