Gaussian Graphical Models

A new Approach to Discrimination

Master of Science Thesis
Spring 2015
Torben Kløjgaard

AALBORG UNIVERSITY
DENMARK

Department of Mathematical Sciences Aalborg University

Department of Mathematical Sciences
Fredrik Bajers Vej 7G
9220 Aalborg Ø
Telephone 99 40 99 40
Fax 9940 3548
http://www.math.aau.dk

Synopsis:

The aim of this thesis it to develop a classification method inspired by Random Forrest where the building blocks instead of trees are Gaussian Graphical models. The classification method is benchmarked against classical discrimination methods and graphical lasso on two different data sets. The performance of the classification methods is done by considering their Receiver Operating Characteristic (ROC) curves and the area under curve (AUC). The strategy for the classification method is to consider a regression on the neighbors and test the individual parameters for significance. Applying this strategy turn out to be very fast even on data sets with a large amount of variables.

Title:
Gaussian Graphical Models: A new Approach to Discrimination

Project period:
February 16, 2015 - August 3, 2015

Supervisor:
Poul Svante Eriksen

Circulation:
4

Number of pages:
59

Appendix:
1

Completed:
August 3, 2015

Author:

Torben Anders Kløjgaard

Preface

This thesis is written as a part of the master at the Department of Mathematical Sciences at Aalborg University, by Torben Kløjgård during the period from February 16, 2015 to August 3, 2015. The main topic is Gaussian Graphical Models

During the preparation of the project there are used various sources. Bibliographical references are done with [name,year], and these can also be found in the bibliography at the end of the report before the appendix. In the bibliography the sources are stated with author, year, publisher, ISBN-number and UrL for homepages.

Mathematical definitions, figures, tables etc. are enumerated in reference to the chapter i.e. the first definition in chapter 2 has number 2.1, the second has number 2.2 etc. References to definitions, theorems, etc. which start with 'A' refer to the appendix. All statistical codes and functions are implemented with the software R. Moreover the implemented codes and functions are framed and called *R-code* throughout the project. Mathematical proofs are ended by □ and examples are ended by ■.

It is assumed that the reader possesses the mathematical qualifications corresponding to completion of the bachelor education of Mathematical Sciences as minimum. In addition, basic knowledge within probability theory and basic statistical concepts.

A special thanks to my supervisor, Associate Professor Poul Svante Eriksen, Department of Mathematical Sciences at Aalborg Univeristy, who has been very dedicated and a great help during the thesis.

Resumé

Formålet med denne specialeafhandling er at konstruere en klassifikations strategi som er inspireret af Random Forest. Men i stedet for træer benyttes Gaussisk grafiske modeller. Når der udføres model selektion i Gaussisk grafiske modeller er den mest optimale metode at lave likelihood ratio test. Dette bliver dog meget tids krævende når antal af variable i et datasæt vokser. Derfor foreslås en model selektions metode som er baseret på regression på naboerne til en knude i en graf. Denne metode viser sig at været rigtig hurtig og effektiv sammenlignet med likelihood ratio test. Metoden kaldes Headlong Method (HLM) eftersom at den vælger en kant i en grat tilfældigt. For at udvikle denne metode benyttes nogle elementære satistiske emner. Her benyttes vigtig emner som betinget sandsynlighed, graf teori og den multivariate normalfordeling som alle udgør fundamentet for en Gaussisk grafiske model.

For at lave regression på naboerne introduceres generelle lineære modeller $\boldsymbol{Y} = X\boldsymbol{\beta} + \sigma^2 I$, hvor X er en design matrix, $\boldsymbol{\beta}$ er regression parameter og σ er standard normalfordelt. Yderligere introduceres t-test som benyttes til at teste de individuelle parameter for signifikans.

HLM metoden benchmarkes imod Lineær Diskiminant Analyse, kvardartisk diskiminant analyse og grafical lasso på to datasæt. De to datasæt er forskellige på den måde at det ene består af mange observationer og få variable og det andet består af få observationer og mange variable. Ved at betragte benchmark resultaterne udvælges den bedste klassifikation model.

List of Notations

X, Y, Z	Random variables.
$\boldsymbol{X}$	Vector of random variables.
$\mathcal{G} = (V, E)$	A graph with a set of vertexes V and a set of edges E.
v_i—v_j	Undirected edge where $v_i, v_j \in V$.
$(\boldsymbol{X} \perp \boldsymbol{Y})$	Independent random variables.
$(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$	Conditional independent random variables.
$\mathrm{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2.
$\mathrm{N}_p(\boldsymbol{\mu}, \Sigma)$	Multiariate normal distribution of dimension p with mean vector $\boldsymbol{\mu}$ and covariance Σ.
$\mathrm{E}[\cdot]$	Expected value.
$\mathrm{Var}[\cdot]$	Variance.
$\mathrm{Cov}[\cdot, \cdot]$	Covariance.
D	Design matrix.
E	The set of edges in a graph.
V	The set of nodes in a graph.
$f(x)$	Density of X.
$f(y\|x)$	Conditional density of X given Y.

Contents

1 Introduction

The main topic of this thesis is graphical models, which today are commonly used in probability theory, statistics and machine learning. A graphical model is illustrated by a graph where the nodes represent variables and the edges represent dependencies between variables. The graph is used to define the joint and conditional probability distribution between variables. More specifically we concentrate on undirected graphical models also known as Markov random fields. A Markov random field is a vector of random variables having Markov properties described by an undirected graph. If the finite-dimensional random vector follows a multivariate normal or Gaussian distribution we have a Gaussian Markov random field (GMRF).

This thesis considers GMRFs also called gaussian graphical models where we cover both theory and their applications. A key point is, of course, the sparsity of the precision matrix (the inverse covariance matrix) and the structure of its nonzero entrances. It is useful to present a GMRF on a graph where the missing edges correspond to zeros in the precision matrix. This provides a unified way of presenting the conditional independencies through a GMRF.

When we are doing model selection in graphical models, the most optimal method is likelihood ratio test which for example is done in the R-package `gRim`. This method can, however, be very time-consuming when the number of variables is large. The reason for this is that we have to fit a new model, using for example iterative proportional scaling (IPS), every time an edge is added or removed.

But what if we could develop a much more efficient and faster method?

In this thesis we aim at developing a classification strategy inspired by Random Forests which we will call the Headlong Method (HLM). Instead of trees the building blocks are Gaussian Graphical Models.

The developed HLM is benchmarked against Linear Discrimination Analysis (LDA), Quadratic Discrimination Analysis (QDA) and graphical lasso. The applied method is depending on the number of observations in proportion to the number of variables in a statistical sample.

Since our task is to describe how the data are classified, we use supervised learning. In supervised learning we know the classes which we use to calibrate a model such that given the real valued variables we can predict the class of an observation.

Throughout the report we use two data sets: One with a large number of observations and few variables and another with few observations and a large number of variables. Both data sets define a supervised classification problem where each observation is composed by some real valued variables and a class variable. The two data sets are used as benchmark for the developed HLM.

The first data set we consider is the `Satellite` data which are obtained from the R-package `mlbench`. The following description of the data is based on the reference manual, [Leisch and Dimitriadou, 2010], which describes the `mlbench` package.

The `Satellite` data consist of multi-spectral values of pixels in 3×3 neighbourhoods in a satellite image and the classification of the central pixel in each neighbourhood. Given the multi-spectral values the aim is to predict the classification. The data frame consists of 6435 observations and 37 variables where the first 36 are real valued variables (`x.1, x.2, ... ,x.36`), and the last one is a classification variable (`classes`). The classification variable contains the following six classes:

- red soil
- cotton crop
- grey soil
- damp grey soil
- soil with vegetation stubble
- very damp grey soil

The data have been obtained from the UCI Machine learning Repository at the url `http://archive.ics.uci.edu/ml/datasets/Statlog+%28Landsat+Satellite%29`.

The variables of the `Satellite` data represent digital images (3×3 pixel) of the same scene in four different spectral bands. Two of the bands are in the visible region and almost correspond to green and red in the visible spectrum. The other two are in the infra-red spectrum. Every value of a variable is expressed as an 8-bit binary number where 0 corresponds to black and 255 to white. The spatial resolution of a pixel, which determines how sharp the image looks, is about 80m×80m, and each image contains 2340×3380 of these pixels.

The `Satellite` data set is only a small part of the entire 2340×3380 pixel image, which consists of 80×100 pixels. Considering each row in the data set it contains the pixel values of the four spectral bands of each of the 9 pixels in the 3×3 neighborhood and the classification of the central pixel.

The data is given in random order, and it is not possible to reconstruct the original image from the data set since certain rows of data have been removed.

Looking closer at each row of the data set, the first four entries are spectral values for the top-left pixel followed by the next four spectral values for the top-middle pixel and then those for the top-right pixel and so forth, provided that the pixels are read as left-to-right and top-to-bottom. Hence the four spectral values associated with the central pixel are the variables `x.17`, `x.18`, `x.19` and `x.20`.

The `Satellite` data have a total of 6435 observations which are divided into two subsets: a training set with 4435 observations and a test set with the remaining 2000 observations.

We also use the data set `breastcancer` from the R-package `gRbase`. This data set is derived from a study comparing gene expression profiles in tumors within two groups of patients with breast cancer, namely a group with and another without a mutation in the p53 tumor suppression gene. The data frame consists of 250 observations and 1001 variables. The first 1000 variables are the log-transformed gene expression values, and the last one is a binary classification variable where `case` stand for patients with a p53 mutation and `control` are patients without a p53 mutation. There are 58 cases and 192 controls in the data set. A description of the study mentioned above is given in [Lance D. Miller, 2005].

The `breastcancer` data are also divided into a training set and a test set each with 125 observations.

The organization of the thesis is as follows:

Chapter 2 treats some of the basic subjects which contribute to the construction of the graphical models considered in this project. We start by giving an introduction to conditional independence, graph theory and the multivariate normal distribution, and with all this in place we can define a Gaussian Markov Random Field (GMRF).

In chapter 3 we consider the theory of general linear models which we use to develop the HLM. Additionally, we introduce LDA, QDA and glasso and these methods are used to benchmark against the HLM. We also introduce the concept Receiver Operating Characteristic (ROC) curves which visualizes the performance of a binary classifier.

In chapter 4 we introduce a new approach to model selection in Gaussian graphical models which is efficient and fast. However, this is not the most optimal method which is obtained by likelihood ratio test. Instead of likelihood ratio test we consider a regression on the neighbors where we use the t-test as test for significance.

In chapter 5 the new obtained HLM is benchmarked against the classical discrimination methods LDA and QDA using the `Satellite` data. The method is also benchmarked against the glasso method using the `breastcancer` data. To compare the different methods we estimate the misclassification rate and accuracy using a confusion matrix. When considering a binary classifier we also estimate the ROC curve.

In chapter 6 we recapitulate the different methods and their ability to classify correctly. We also discuss when to prefer a certain method over another using the results achieved in chapter 5.

Background Theory 2

In this chapter we introduce some of the basic concepts for understanding Gaussian Markov Random Fields (GMRF) which is introduced at the end of this chapter. We start by considering conditional independence which is an important concept in the applications of GMRFs later on.

2.1 Conditional independence

This section is inspired by [Rue and Held, 2005, Section 2.1.4].

The notation $f(\cdot)$ and $f(\cdot|\cdot)$ is a generic notation for densities, that is $f(\boldsymbol{x})$ is the density of the random vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)^T$ and $f(\boldsymbol{x}_A|\boldsymbol{x}_{-A})$ is the conditional density of $\boldsymbol{X}_A$, given a realization $\boldsymbol{X}_{-A} = \boldsymbol{x}_{-A}$ where $A \subseteq \{1, 2, \ldots, p\}$ and $-A$ is the complement of A. To compute this conditional density we use that:

$$f(\boldsymbol{x}_A|\boldsymbol{x}_{-A}) = \frac{f(\boldsymbol{x})}{f(\boldsymbol{x}_{-A})},$$

where $f(\boldsymbol{x}_{-A}) > 0$.

Two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent if and only if $f(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x})f(\boldsymbol{y})$ and we notate this by $\boldsymbol{X} \perp \boldsymbol{Y}$. Additionally we have that two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditionally independent given a random vector $\boldsymbol{Z}$ if and only if $f(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{z}) = f(\boldsymbol{x}|\boldsymbol{z})f(\boldsymbol{x}|\boldsymbol{z})$. This is denoted $\boldsymbol{Y} \perp \boldsymbol{Y}|\boldsymbol{Z}$. Although $\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditionally independent given $\boldsymbol{Z}$, they might be marginally dependent.

An important property of conditional independence is the following factorization criterion. Using this criterion it is easy to verify conditional independence.

Theorem 2.1
Let $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$ be random vectors, then:

$$\boldsymbol{X} \perp \boldsymbol{Y}|\boldsymbol{Z} \quad \Longleftrightarrow \quad f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \phi_1(\boldsymbol{x}, \boldsymbol{z})\phi_2(\boldsymbol{y}, \boldsymbol{z}) \tag{2.1}$$

for some functions ϕ_1 and ϕ_2, and for all $f(\boldsymbol{z}) > 0$.

Proof. By considering $\boldsymbol{X} \perp \boldsymbol{Y}|\boldsymbol{Z}$ we have that:

$$\begin{aligned} f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) &= f(\boldsymbol{x}|\boldsymbol{z})f(\boldsymbol{y}|\boldsymbol{z})f(\boldsymbol{z}) \\ &= f(\boldsymbol{x},\boldsymbol{z})f(\boldsymbol{y}|\boldsymbol{z}) \\ &= \phi_1(\boldsymbol{x},\boldsymbol{z})\phi_2(\boldsymbol{y},\boldsymbol{z}). \end{aligned}$$

This indicates that if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditionally independent given $\boldsymbol{Z}$, they can be written as the prescribed product of two functions.

The other way around we consider the factorization, $\phi_1(\boldsymbol{x},\boldsymbol{z})\phi_2(\boldsymbol{y},\boldsymbol{z})$ which imply that the joint density is given as:

$$f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \phi_1(\boldsymbol{x},\boldsymbol{z})\phi_2(\boldsymbol{y},\boldsymbol{z}).$$

If we integrate $\boldsymbol{x}$ and $\boldsymbol{y}$ out, respectively, then we obtain the densities:

$$\begin{aligned} f(\boldsymbol{x},\boldsymbol{z}) &= \phi_1(\boldsymbol{x},\boldsymbol{z}) \int \phi_2(\boldsymbol{y},\boldsymbol{z})d\boldsymbol{y} \\ f(\boldsymbol{y},\boldsymbol{z}) &= \phi_2(\boldsymbol{y},\boldsymbol{z}) \int \phi_1(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{x}, \end{aligned}$$

and dividing both equations above by $f(\boldsymbol{z})$ yields the conditional densities:

$$f(\boldsymbol{x}|\boldsymbol{z}) = \frac{\phi_1(\boldsymbol{x},\boldsymbol{z})}{f(\boldsymbol{z})} \int \phi_2(\boldsymbol{y},\boldsymbol{z})d\boldsymbol{y} \tag{2.2}$$

$$f(\boldsymbol{y}|\boldsymbol{z}) = \frac{\phi_2(\boldsymbol{y},\boldsymbol{z})}{f(\boldsymbol{z})} \int \phi_1(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{x}. \tag{2.3}$$

If we integrate $\boldsymbol{x}$ out in (2.2) we obtain:

$$\frac{1}{f(\boldsymbol{z})} \int \phi_1(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{x} \int \phi_2(\boldsymbol{y},\boldsymbol{z})d\boldsymbol{y} = 1 \tag{2.4}$$

Hence by using (2.4) the product of (2.2) and (2.3) is given as:

$$\begin{aligned} f(\boldsymbol{x}|\boldsymbol{z})f(\boldsymbol{y}|\boldsymbol{z}) &= \frac{\phi_1(\boldsymbol{x},\boldsymbol{z})\phi_2(\boldsymbol{y},\boldsymbol{z})}{f(\boldsymbol{z})^2} \int \phi_2(\boldsymbol{y},\boldsymbol{z})d\boldsymbol{y} \int \phi_1(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{x} \\ &= \frac{\phi_1(\boldsymbol{x},\boldsymbol{z})\phi_2(\boldsymbol{y},\boldsymbol{z})}{f(\boldsymbol{z})}, \end{aligned}$$

which implies that the joint density can be expressed as:

$$f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = f(\boldsymbol{x}|\boldsymbol{z})f(\boldsymbol{y}|\boldsymbol{z})f(\boldsymbol{z}).$$

This shows that the factorization, $\phi_1(\boldsymbol{x},\boldsymbol{z})\phi_2(\boldsymbol{y},\boldsymbol{z})$, can be written as a product of the densities $f(\boldsymbol{x}|\boldsymbol{z})$ and $f(\boldsymbol{y}|\boldsymbol{z})$, which indicates that $\boldsymbol{X} \perp \boldsymbol{Y}|\boldsymbol{Z}$. □

Aside from conditional independence another very important topic is graphs which are keeping track of the conditional independence.

2.2 Graph Theory

Graphs are useful in a variety of applications, for instance we use graphs to illustrate conditional independence between variables in the vector $\boldsymbol{X}$. In this section we give a short introduction to graph theory where [Koller and Friedman, 2009, Section 2.2] has been used as source of inspiration.

A *graph*, as we use it in this project, is a pair of sets $\mathcal{G} = (V, E)$ where V is a set of nodes also called vertexes, and E is a set of edges $\{v_i, v_j\}$ where $v_i, v_j \in V$ and $i \neq j$. The size of a graph is the number of its edges and is denoted $|E|$. A pair of vertexes, v_i, v_j, in a graph can be connected by an undirected edge, v_i—v_j, or a directed edge, $v_i \to v_j$. If the graph only contains undirected edges, we say that it is an *undirected* graph and if all edges are directed, the graph is said to be *directed*. However, in this project we only concentrate on undirected graphs.

There is an undirected edge between the nodes v_i and v_j if $\{v_i, v_j\} \in E$. Otherwise there is no edge between the two nodes. A graph is called *complete* if $\{v_i, v_j\} \in E$ for all $v_i, v_j \in V$, where $i \neq j$.

Whenever we have v_i—$v_j \in E$, we say that v_i is a *neighbor* to v_j in $\mathcal{G}$ and vice versa. The neighbors of the node v_i is denoted by $\text{Ne}(v_i) = \{v_j \in V : \{v_i, v_j\} \in E\}$. The *degree* of a vertex v_i is the number of edges that connect to it, and the degree of a graph is the maximal degree of a vertex in the graph. Figure 2.1 shows an example of an undirected graph where for example $\text{Ne}(v_1) = v_2$ and $\text{Ne}(v_2) = \{v_2, v_3\}$. Also we see that the degree of the graph is three.

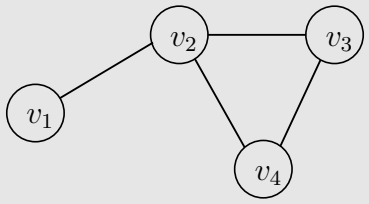

Figure 2.1. An undirected graph.

By using the basic notation of edges, we can define some longer-range connections in the graph. For example we have that a *path* from v_1 to v_m is a sequence of distinct nodes in V for which $(v_i, v_{i+1}) \in E$ where $i = 1, 2, \ldots, m-1$. We say that a subset $C \subset V$ *separates* two disjoint subsets $A \subset V \setminus C$ and $B \subset \setminus C$ if all $a_i \in A$ and $b_i \in B$ are separated by C. That is, every path between A and B passes through C by at last one node.

When talking about graphs we also have the important topic of subgraphs. A graph $\mathcal{G}_0 = (V_0, E_0)$ is said to be a *subgraph* of $\mathcal{G} = (V, E)$ if $V_0 \subseteq V$ and $E_0 \subseteq E$. For example, if we let $\mathcal{G}$ be the graph in figure 2.1, then the graph $\mathcal{G}_0 = (V_0, E_0)$, where $V_0 = \{v_1, v_2, v_3\}$ and $E_0 = \{\{v_1, v_2\}, \{v_2, v_3\}\}$, is a subgraph of $\mathcal{G}$.

2.3 The Multivariate Normal Distribution

This section is based on the two sources [Koller and Friedman, 2009, Section 7.1] and [Lauritzen, 1996, Section C.1].

A multivariate normal distribution or multivariate Gaussian distribution over the random vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$, is characterized by an p-dimensional mean vector $\boldsymbol{\mu}$ and a symmetric $p \times p$ covariance matrix Σ. The density of $\boldsymbol{X}$ is defined as:

$$f(\boldsymbol{x}) = (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big), \tag{2.5}$$

where $|\cdot|$ is the determinant. This we also write as $\boldsymbol{X} \sim \mathrm{N}_p(\boldsymbol{\mu}, \Sigma)$. In order for the equation to induce a well-defined density, that is, it integrate to one, the covariance matrix, Σ, has to be *positive definite*. The matrix, Σ, is said to be positive definite if $\boldsymbol{x}^T\Sigma\boldsymbol{x} > 0$ for any $\boldsymbol{x} \in \mathbb{R}^p$ where $\boldsymbol{x} \neq 0$. Since Σ is also symmetric, then it is called a symmetric and positive definite (SPD) matrix.

Now assume that $\boldsymbol{X}$ is partitioned into two parts so that $\boldsymbol{X} = (\boldsymbol{X}_A^T, \boldsymbol{X}_B^T)^T$, where $\boldsymbol{X}_A \in \mathbb{R}^r$ and $\boldsymbol{X}_B \in \mathbb{R}^s$ with $p = r + s$. The mean vector and covariance matrix can then be partitioned into the following block matrices:

$$\boldsymbol{\mu} = \begin{pmatrix}\boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B\end{pmatrix} \quad \Sigma = \begin{pmatrix}\Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB}\end{pmatrix}$$

such that $\Sigma_{AA} \in \mathbb{R}^{r\times r}$ is the covariance matrix of $\boldsymbol{X}_A$, $\Sigma_{BB} \in \mathbb{R}^{s\times s}$ is the covariance matrix of $\boldsymbol{X}_B$ and $\Sigma_{AB} \in \mathbb{R}^{r\times s}$ is the covariance matrix between $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$. Note that $\Sigma_{BA} = \Sigma_{AB}^T$.

Remark 2.2
If $\boldsymbol{X}$, $\boldsymbol{\mu}$ and Σ are partitioned as above where $\Sigma_{AB} = 0$. Then $\boldsymbol{X}$ has density:

$$\begin{aligned} f(\boldsymbol{x}) &= c \cdot \exp\Big(-\frac{1}{2}(\boldsymbol{x}_A-\boldsymbol{\mu}_A)^T\Sigma_{AA}^{-1}(\boldsymbol{x}_A-\boldsymbol{\mu}_A) - \frac{1}{2}(\boldsymbol{x}_B-\boldsymbol{\mu}_B)^T\Sigma_{BB}^{-1}(\boldsymbol{x}_B-\boldsymbol{\mu}_B)\Big) \\ &= c \cdot f(\boldsymbol{x}_A)f(\boldsymbol{x}_B) \end{aligned}$$

where c is a normalization constant. This implies that $\Sigma_{AB} = 0$ if and only if $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are independent.

The conditional distribution of $\boldsymbol{X}_A$ given $\boldsymbol{X}_B = \boldsymbol{x}_B$ and the marginal distribution of $\boldsymbol{X}_B$ can be found as follows.

Theorem 2.3
Let $\boldsymbol{X}$ be multivariate normal distributed with mean vector $\boldsymbol{\mu}$ and SPD covariance matrix Σ where $\boldsymbol{X}$, $\boldsymbol{\mu}$ and Σ are partitioned as above. If the sub-matrix

Σ_{BB} is positive definite, then the conditional distribution of $\boldsymbol{X}_A$ given $\boldsymbol{X}_B = \boldsymbol{x}_B$ is:

$$\boldsymbol{X}_A|\boldsymbol{X}_B = \boldsymbol{x}_B \sim N_r(\boldsymbol{\mu}_{A|B}, \Sigma_{A|B}), \tag{2.6}$$

where

$$\boldsymbol{\mu}_{A|B} = \boldsymbol{\mu}_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B) \quad \text{and} \quad \Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}.$$

Additionally the marginal distribution over $\boldsymbol{X}_B$ has a multivariate normal distribution given by $N_s(\boldsymbol{\mu}_B, \Sigma_{BB})$.

Proof. We note that when Σ is positive definite, then Σ_{BB} is positive definite and invertible. Hence $\boldsymbol{\mu}_{A|B}$ and $\Sigma_{A|B}$ is well-defined. Consider a transformation from $\boldsymbol{X} = (\boldsymbol{X}_A^T, \boldsymbol{X}_B^T)^T$ to the new variables $\boldsymbol{X}_B$ and $\boldsymbol{X}'_A = \boldsymbol{X}_A - \Sigma_{AB}\Sigma_{BB}^{-1}\boldsymbol{X}_B$ which is obtained by the linear transformation:

$$\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{X}'_A \\ \boldsymbol{X}_B \end{pmatrix} = \begin{pmatrix} I & -\Sigma_{AB}\Sigma_{BB}^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \boldsymbol{X}_A \\ \boldsymbol{X}_B \end{pmatrix} = A\boldsymbol{X} \tag{2.7}$$

According to section A.3 in appendix the linear transformation above is a multivariate normal distribution with mean $A\boldsymbol{\mu}$ and covariance $A^T\Sigma A$. For convenience we let $D = -\Sigma_{AB}\Sigma_{BB}^{-1}$. Thus the covariance of $\boldsymbol{X}'_A$ and $\boldsymbol{X}_B$ is given as:

$$\begin{aligned} \mathrm{Cov}\left[\boldsymbol{X}'_A, \boldsymbol{X}_B\right] &= \mathrm{Cov}\left[\boldsymbol{X}_A + D\boldsymbol{X}_B, \boldsymbol{X}_B\right] \\ &= \mathrm{Cov}\left[\boldsymbol{X}_A, \boldsymbol{X}_B\right] + D\mathrm{Cov}\left[\boldsymbol{X}_B, \boldsymbol{X}_B\right] \\ &= \Sigma_{AB} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BB} \\ &= 0 \end{aligned}$$

This indicates that $\boldsymbol{X}'_A$ and $\boldsymbol{X}_B$ are uncorrelated, and since they are multivariate normal distributed, they are independent according to remark 2.2. Hence we have the following conditional mean:

$$\begin{aligned} \mathrm{E}\left[\boldsymbol{X}_A|\boldsymbol{X}_B = \boldsymbol{x}_B\right] &= \mathrm{E}\left[\boldsymbol{X}'_A - D\boldsymbol{X}_B|\boldsymbol{X}_B = \boldsymbol{x}_B\right] \\ &= \mathrm{E}\left[\boldsymbol{X}'_A|\boldsymbol{X}_B = \boldsymbol{x}_B\right] - \mathrm{E}\left[D\boldsymbol{X}_B|\boldsymbol{X}_B = \boldsymbol{x}_B\right] \\ &= \mathrm{E}\left[\boldsymbol{X}'_A\right] - D\boldsymbol{x}_B \\ &= \boldsymbol{\mu}_A + D\boldsymbol{\mu}_B - D\boldsymbol{x}_B \\ &= \boldsymbol{\mu}_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B) \end{aligned}$$

which proves the mean in (2.6). Considering the covariance matrix note that:

$$\begin{aligned} \mathrm{Var}\left[\boldsymbol{X}_A|\boldsymbol{X}_B = \boldsymbol{x}_B\right] =& \mathrm{Var}\left[\boldsymbol{X}'_A - D\boldsymbol{X}_B|\boldsymbol{X}_B = \boldsymbol{x}_B\right] \\ =& \mathrm{Var}\left[\boldsymbol{X}'_A|\boldsymbol{X}_B = \boldsymbol{x}_B\right] + \mathrm{Var}\left[D\boldsymbol{X}_B|\boldsymbol{X}_B = \boldsymbol{x}_B\right] \\ & - D\mathrm{Cov}\left[\boldsymbol{X}'_A, \boldsymbol{X}_B\right] - \mathrm{Cov}\left[\boldsymbol{X}_B, \boldsymbol{X}'_A\right] D^T \\ =& \mathrm{Var}\left[\boldsymbol{X}'_A|\boldsymbol{X}_B = \boldsymbol{x}_B\right] \\ =& \mathrm{Var}\left[\boldsymbol{X}'_A\right]. \end{aligned}$$

This indicate that the conditional variance is given as:

$$\begin{aligned}
\mathrm{Var}\left[\boldsymbol{X}_A|\boldsymbol{X}_B=\boldsymbol{x}_B\right] &= \mathrm{Var}\left[\boldsymbol{X}'_A\right] \\
&= \mathrm{Var}\left[\boldsymbol{X}_A + D\boldsymbol{X}_B\right] \\
&= \mathrm{Var}\left[\boldsymbol{X}_A\right] + \mathrm{Var}\left[D\boldsymbol{X}_B\right] + D\mathrm{Cov}\left[\boldsymbol{X}_A, \boldsymbol{X}_B\right] + \mathrm{Cov}\left[\boldsymbol{X}_B, \boldsymbol{X}_A\right] D^T \\
&= \Sigma_{AA} + \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BB}\Sigma_{BB}^{-1}\Sigma_{BA} - 2\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA} \\
&= \Sigma_{AA} + \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA} - 2\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA} \\
&= \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}
\end{aligned}$$

which proves the variance in (2.6). Since $\boldsymbol{X}'_A$ and $\boldsymbol{X}_B$ are independent the conditional distribution of $\boldsymbol{X}'_A$ given $\boldsymbol{X}_B = \boldsymbol{x}_B$ can be found as:

$$\begin{aligned}
\boldsymbol{X}'_A|\boldsymbol{X}_B = \boldsymbol{x}_B &\sim \mathrm{N}_r\Big(\boldsymbol{\mu}_A - \Sigma_{AB}\Sigma_{BB}^{-1}\boldsymbol{\mu}_B, \Sigma_{A|B}\Big) \\
\Rightarrow \boldsymbol{X}_A - \Sigma_{AB}\Sigma_{BB}^{-1}\boldsymbol{X}_B|\boldsymbol{X}_B = \boldsymbol{x}_B &\sim \mathrm{N}_r\Big(\boldsymbol{\mu}_A - \Sigma_{AB}\Sigma_{BB}^{-1}\boldsymbol{\mu}_B, \Sigma_{A|B}\Big) \\
\Rightarrow \boldsymbol{X}_A - \Sigma_{AB}\Sigma_{BB}^{-1}\boldsymbol{x}_B|\boldsymbol{X}_B = \boldsymbol{x}_B &\sim \mathrm{N}_r\Big(\boldsymbol{\mu}_A - \Sigma_{AB}\Sigma_{BB}^{-1}\boldsymbol{\mu}_B, \Sigma_{A|B}\Big) \\
\Rightarrow \boldsymbol{X}_A|\boldsymbol{X}_B = \boldsymbol{x}_B &\sim \mathrm{N}_r\Big(\boldsymbol{\mu}_{A|B}, \Sigma_{A|B}\Big).
\end{aligned}$$

Thus the conditional distribution (2.6) is also normal.

From the above results the variance of $\boldsymbol{Y}$ in (2.7) is:

$$\mathrm{Var}\left[\boldsymbol{Y}\right] = \begin{pmatrix} \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA} & 0 \\ 0 & \Sigma_{BB} \end{pmatrix}$$

Hence from remark 2.2 $\boldsymbol{X}_B \sim \mathrm{N}\left(\boldsymbol{\mu}_B, \Sigma_{BB}\right)$. □

It follows from theorem 2.3 that the conditional mean:

$$\mathrm{E}\left[\boldsymbol{X}_A|\boldsymbol{X}_B=\boldsymbol{x}_B\right] = \boldsymbol{\mu}_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B)$$

depends linearly on $\boldsymbol{x}_B$, while the conditional covariance matrix:

$$\mathrm{Var}\left[\boldsymbol{X}_A|\boldsymbol{X}_B=\boldsymbol{x}_B\right] = \Sigma_{\boldsymbol{X}_A|\boldsymbol{X}_B}$$

is not dependent on $\boldsymbol{x}_B$.

Based on the concepts which has been presented so far, we can now turn to the introduction of Gaussian Markov random field.

2.4 Gaussian Markov Random Fields

A Gaussian Markov random field (GMRF) is simply a vector following a multivariate Gaussian distribution. GMRFs is very useful in practice since we are able to compute the things we often need to know in a fast way. This is due to the conditional independence restrictions in the multivariate normal distribution which

can be expressed through zero restrictions on the inverse covariance matrix. This section is inspired by [Rue and Held, 2005, Section 2.2.1].

Since positive definite matrices are invertible, we can derive another parameterization of the multivariate Gaussian distribution where we instead use the SPD inverse covariance matrix $\Sigma^{-1} = J$ called the *precision matrix*. Thus the exponent of (2.5) is:

$$\begin{aligned} -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T J(\boldsymbol{x}-\boldsymbol{\mu}) \\ &= -\frac{1}{2}\left(\boldsymbol{x}^T J\boldsymbol{x} - 2\boldsymbol{x}^T J\boldsymbol{\mu} + \boldsymbol{\mu}^T J\boldsymbol{\mu}\right). \end{aligned}$$

As the last term in this equation is constant, we obtain the density:

$$\begin{aligned} f(\boldsymbol{x}) &\propto \exp\Big(-\frac{1}{2}\boldsymbol{x}^T J\boldsymbol{x} + (J\boldsymbol{\mu})^T\boldsymbol{x}\Big) \\ &= \exp\Big(-\frac{1}{2}\boldsymbol{x}^T J\boldsymbol{x} + (\boldsymbol{h})^T\boldsymbol{x}\Big). \end{aligned} \tag{2.8}$$

This expression of the Gaussian density is called the *information form*, and the vector $\boldsymbol{h} = J\boldsymbol{\mu}$ is called the potential vector.

Now let $\boldsymbol{X} = (X_1, \ldots, X_p)^T$ be a p-dimensional normal distributed vector and let $\mathcal{G} = (V, E)$ an undirected graph where the vertices $V = \{1, \ldots, p\}$ represent the random vector $\boldsymbol{X}$. The set of edges E is constructed such that $\{X_i, X_j\} \notin E$, if and only if X_i and X_j obey the pairwise Markov property, that is:

$$X_i \perp X_j | \boldsymbol{X}_{-i,-j}, \quad i \neq j$$

where $\boldsymbol{X}_{-i,-j} = \boldsymbol{X} \setminus \{X_i, X_j\}$. Here we say that X_i and X_j are independent given the rest of the graph. Additionally, a random vector is said to obey the local Markov property relative to the graph $\mathcal{G}$ if:

$$X_i \perp X_{-\{i,\mathrm{Ne}(i)\}} | X_{\mathrm{Ne}(i)}$$

where $X_{-\{i,\mathrm{Ne}(i)\}} = \boldsymbol{X} \setminus \{X_i, X_{\mathrm{Ne}(i)}\}$. In other words, the local Markov property state that X_i is independent of the rest of the vertices in the graph given its neighbors.

According to theorem 2.4 in [Rue and Held, 2005, Section 2.2.2] the local and pairwise Markov properties are equivalent when $f(\boldsymbol{x})$ is multivariate normal distributed. Hence a GMRF must obey the pairwise Markov property as well as the local Markov property. This means that the pairwise Markov property can be used to define the independence restrictions in the GMRF.

As it turns out in the following theorem, the conditional independence in the multivariate gaussian distribution is not reflected by the covariance matrix, but by the precision matrix.

Theorem 2.4
Let $\boldsymbol{X}$ be a normal distributed random vector with mean $\boldsymbol{\mu}$ and SPD precision matrix J. Then :

$$X_i \perp X_j | \boldsymbol{X}_{-i,-j} \quad \Longleftrightarrow \quad J_{ij} = 0,$$

where $i \neq j$.

Proof. To prove the theorem we let $\boldsymbol{X}$ be partitioned as $(X_i, X_j, \boldsymbol{X}_{-i,-j})$ and use theorem 2.1 on the density $f(x_i, x_j, \boldsymbol{x}_{-i.-j})$. If we fix $i \neq j$ and assume $\boldsymbol{\mu} = \boldsymbol{0}$ without loss of generality, then from (2.8) we get:

$$\begin{aligned} f(x_i, x_j, \boldsymbol{x}_{-i,-j}) &\propto \exp\Big(-\frac{1}{2}\sum_{k,l} x_k J_{kl} x_l\Big) \\ &= \exp\Big(-x_i x_j J_{ij} - \frac{1}{2}\sum_{\{k,l\}\neq\{i,j\}} x_k J_{kl} x_l\Big) \end{aligned}$$

The last term of the above equation does not involve $x_i x_j$, but the first term involves $x_i x_j$ if and only if $J_{ij} \neq 0$. Hence:

$$f(x_i, x_j, \boldsymbol{x}_{-i,-j}) = \phi_1(x_i, \boldsymbol{x}_{-i,-j})\phi_2(x_j, \boldsymbol{x}_{-i,-j})$$

for some functions ϕ_1 and ϕ_2, if and only if $J_{ij} = 0$. The conditional independence $X_i \perp X_j | \boldsymbol{X}_{-i,-j}$ then follows from theorem 2.1. □

By using theorem 2.4 the definition of a GMRF can be stated as:

Definition 2.5
A random vector $\boldsymbol{X} = (X_1, \ldots, X_p)$ is called a GMRF with regard to an undirected graph $\mathcal{G} = (V, E)$ with mean $\boldsymbol{\mu}$ and SPD precision matrix J, if and only if its density is on the form:

$$f(\boldsymbol{x}) = (2\pi)^{-\frac{n}{2}} |J|^{\frac{1}{2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})J(\boldsymbol{x}-\boldsymbol{\mu})^T\Big)$$

and

$$\{i,j\} \notin E \quad \Longrightarrow \quad J_{ij} = 0 \quad \text{for all} \quad i \neq j.$$

Note if J is a completely dense matrix then the graph $\mathcal{G}$ is complete. In this report we will focus on the case where the precision matrix is sparse since GMRFs in such situations are really useful.

Theory 3

3.1 General Linear Model

In this section we introduce general linear models with [Thyregod and Madsen, 2010] as a source of inspiration.

In the case of a normal density the random vector $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_N)^T$ can be written in the matrix form:

$$\boldsymbol{Y} = \boldsymbol{\mu} + \boldsymbol{\varepsilon}.$$

In many applications the Y's are assumed to be independent and have same the variance, that is:

$$\boldsymbol{\varepsilon} \sim \mathrm{N}\left(\mathbf{0}, \sigma^2 I_n\right).$$

This yields the following definition.

Definition 3.1
A *general linear model* (GLM) for $\boldsymbol{Y}$ is a model of the form $\boldsymbol{Y} \sim \mathrm{N}(\boldsymbol{\mu}, \sigma^2 I)$, where an affine hypothesis is formulated for $\boldsymbol{\mu}$. The hypothesis is of the form:

$$\mathcal{H}_0 : \boldsymbol{\mu} - \boldsymbol{\mu}_0 \in \Omega_0, \tag{3.1}$$

where Ω_0 is a linear subspace of $\mathbb{R}^n$ with dimension l, and where $\boldsymbol{\mu}_0$ is a vector of a priori known offset values.

Note that the dimension l of the subspace Ω_0 is the dimension of the model.

Definition 3.2
Assume that a linear subspace is given by $\Omega_0 = \mathrm{span}\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_l\}$, that is, the subspace is spanned by l linear independent vectors, where $l \leq N$. Consider

the general linear model (3.1) for which the hypothesis can be written as:

$$\mathcal{H}_0 : \boldsymbol{\mu} - \boldsymbol{\mu}_0 = D\boldsymbol{\beta}, \quad \boldsymbol{\beta} \in \mathbb{R}^l$$

where D has full rank. The $N \times l$ matrix D of known deterministic coefficients is called the *design matrix*.

The ith row of the design matrix is given by the *model vector*:

$$\boldsymbol{d}_i = (d_{i1}, d_{i2}, \ldots, d_{il}),$$

for the ith observation.

Often the offset value $\boldsymbol{\mu}_0$ in the hypothesis is the zero vector, and the hypothesis then states that $\boldsymbol{\mu}$ lies in a l-dimensional subspace of $\mathbb{R}^N$ spanned by the columns in D. If the offset value $\boldsymbol{\mu}_0$ is different from the zero vector, the hypothesis specifies that $\boldsymbol{\mu}$ lies in a l-dimensional affine space of $\mathbb{R}^N$. In the following we do not include the offset $\boldsymbol{\mu}_0$ since we for the normal distribution model just let $\boldsymbol{Y} := \boldsymbol{Y} - \boldsymbol{\mu}_0$ to obtain $\boldsymbol{\mu}_0 = \mathbf{0}$.

Notice that a design matrix for a GLM is not unique: it is just some matrix whose columns span the linear subspace for the mean vector of the GLM. Sometimes we choose it to be of full rank, typically when establishing some theoretical results.

3.1.1 Estimated parameters in a GLM

Under the following hypothesis:

$$\mathcal{H}_0 : \boldsymbol{\mu} \in \Omega_0$$

the maximum likelihood estimate for $\boldsymbol{\mu}$ is given as the orthogonal projection of $\boldsymbol{y}$ onto the linear subspace Ω_0.

Theorem 3.3
Given the hypothesis:

$$\mathcal{H}_0 : \boldsymbol{\mu}(\boldsymbol{\beta}) = D\boldsymbol{\beta}$$

where D is a $N \times l$ matrix, the maximum likelihood of $\boldsymbol{\beta}$ is found as a solution to the normal equation:

$$D^T\boldsymbol{y} = D^T D\boldsymbol{\beta}$$

Assuming D has full rank, the solution is uniquely given by:

$$\widehat{\boldsymbol{\beta}} = (D^T D)^{-1} D^T \boldsymbol{y} \tag{3.2}$$

The proof of the above theorem is omitted but can be found in [Thyregod and Madsen, 2010, Section 3.4].

Since $\widehat{\boldsymbol{\beta}}$ is a linear combination of the Y's, which is normally distributed, and D is assumed to have full rank, then $\widehat{\boldsymbol{\beta}}$ is a l dimension normal distribution due to lemma A.2 where the mean is given by:

$$\begin{aligned}
\mathrm{E}\left[\widehat{\boldsymbol{\beta}}\right] &= \mathrm{E}\left[(D^T D)^{-1} D^T \boldsymbol{Y}\right] \\
&= (D^T D)^{-1} D^T \mathrm{E}\left[\boldsymbol{Y}\right] \\
&= (D^T D)^{-1} D^T D\boldsymbol{\beta} \\
&= \boldsymbol{\beta},
\end{aligned}$$

and the variance is:

$$\begin{aligned}
\mathrm{Var}\left[\widehat{\boldsymbol{\beta}}\right] &= \mathrm{Var}\left[(D^T D)^{-1} D^T \boldsymbol{Y}\right] \\
&= \left(D^T D\right)^{-1} D^T \mathrm{Var}\left[\boldsymbol{Y}\right] \left(\left(D^T D\right)^{-1} D^T\right)^T \\
&= \sigma^2 \left(D^T D\right)^{-1} D^T \left(D^T\right)^T \left((D^T D)^{-1}\right)^T \\
&= \sigma^2 \left(D^T D\right)^{-1} D^T D \left(D^T D\right)^{-1} \\
&= \sigma^2 \left(D^T D\right)^{-1}.
\end{aligned}$$

Hence the distrtbution over $\widehat{\boldsymbol{\beta}}$ is:

$$\widehat{\boldsymbol{\beta}} \sim \mathrm{N}\left(\boldsymbol{\beta}, \sigma^2 (D^T D)^{-1}\right). \tag{3.3}$$

The projection of $\boldsymbol{y}$ onto the subspace Ω_0 spanned by D gives us the fitted values $\widehat{\boldsymbol{\mu}} = D\widehat{\boldsymbol{\beta}}$, where $\widehat{\boldsymbol{\beta}}$ indicates the local coordinates for the projection.

Definition 3.4
A *projection matrix* H is a matrix satisfying:

$$H = H^T \quad \text{and} \quad H^2 = H,$$

that is, the matrix is idempotent.

Since $\widehat{\boldsymbol{\mu}} = D\widehat{\boldsymbol{\beta}}$, it is easily seen that the matrix:

$$H = D(D^T D)^{-1} D^T$$

is a projection matrix.

As we also are interested in the estimation of the variance, we have the following theorem.

Theorem 3.5
Under the hypothesis:

$$\mathcal{H}_0 : \boldsymbol{\mu}(\boldsymbol{\beta}) = D\boldsymbol{\beta}$$

the bias corrected maximum likelihood estimate for the variance σ^2 is:

$$\widehat{\sigma}^2 = \frac{\left\| \boldsymbol{y} - D\widehat{\boldsymbol{\beta}} \right\|^2}{N - l} = \frac{\left\| (I - H)\boldsymbol{y} \right\|^2}{N - l}. \tag{3.4}$$

Under this hypothesis the distribution of $\widehat{\sigma}^2$ is given as $\widehat{\sigma}^2 \sim \frac{\sigma^2}{N-l}\chi^2_{N-l}$ where χ^2_k is the chi-squared distribution with k degrees of freedom.

We will not cover the proof of the above theorem here, but it can be found in [Thyregod and Madsen, 2010, Section 3.4].

3.1.2 Test of significance

In some cases we want to test the null hypothesis that β_j is equal to some particular value β_j^0. The value of β_j^0 is often taken to be zero which indicates the hypothesis that d_{ij} and y_i are independent. The following theorem states a significance test of individual parameters.

Theorem 3.6
For a hypothesis $\beta_j = \beta_j^0$ which is related to specific values of the parameters is evaluated using the following test quantity:

$$t_j = \frac{\widehat{\beta_j} - \beta_j^0}{\widehat{\sigma}\sqrt{(D^T D)^{-1}_{jj}}} \tag{3.5}$$

The test quantity is compared with the quantiles of the t-distribution $\mathrm{t}(N-l)$. Here the hypothesis is rejected for large values of t_j, that is, for:

$$|t_j| > \mathrm{t}_{1-\alpha/2}(N - l),$$

where α is a level of significance and the p-value is $2P(\mathrm{t}(N-l) \geq |t_j|)$.

Note, that in the special case where $\beta_j^0 = 0$ the test quantity is given as:

$$t_j = \frac{\widehat{\beta_j}}{\widehat{\sigma}\sqrt{(D^T D)^{-1}_{jj}}}. \tag{3.6}$$

The confidence interval for β_j is found as:

$$\widehat{\beta}_j \pm \mathrm{t}_{1-\alpha/2}(N-l)\widehat{\sigma}\sqrt{(D^T D)^{-1}_{jj}}.$$

In order to ensure that the test quantity (3.5) follows a t-distribution, it is required that the two estimates $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$ are independent. To see if they are independent, we calculate the covariance of (3.2) and (3.4). Since $\widehat{\sigma}^2$ is a function $(I-H)\boldsymbol{y}$ the covariance of $\widehat{\sigma}^2$ and $\widehat{\boldsymbol{\beta}}$ is:

$$\begin{aligned}
\mathrm{Cov}\left[\widehat{\boldsymbol{\beta}}, (I-H)\boldsymbol{y}\right] &= (D^T D)^{-1} D^T \mathrm{Cov}\left[\boldsymbol{y}, \boldsymbol{y}\right](I-H) \\
&= (D^T D)^{-1} D^T \sigma^2 I (I-H) \\
&= (D^T D)^{-1} D^T \sigma^2 I - (D^T D)^{-1} D^T D (D^T D)^{-1} D^T \sigma^2 I \\
&= 0
\end{aligned}$$

which according to remark 2.2 indicates that $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}^2$ are independent.

3.2 Discriminant Analysis

In this section we describe *Discriminant Analysis*, which is a statistical analysis to predict a categorical dependent variable Y based on known continuous variables $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)^T$. The qualitative response variable Y can take on K possible distinct and unordered values. Throughout this section we will introduce *linear discriminant analysis* (LDA) and *quadratic discriminant analysis* (QDA), which are inspired by [J. Friedman, 2009, Section 4.3] and [Gareth James, 2013, Section 4.4]. Later on we will be using the two method as benchmark against a new developed classification method.

3.2.1 Linear Discriminant Analysis

Linear Discriminant Analysis is a method for finding the probability that an observation belongs to a specific class k when $\boldsymbol{X} = \boldsymbol{x}$, which is the same as finding $P(Y = k|\boldsymbol{X} = \boldsymbol{x})$. Suppose $f(\boldsymbol{X} = \boldsymbol{x}|Y = k) = f_k(\boldsymbol{x})$ is the density of $\boldsymbol{X}$ when $Y = k$. This density is also called the class-conditional density. Additionally let $\pi_k = P(Y = k)$ be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$.

A natural rule would be to classify according to:

$$\widehat{Y} = \underset{k}{\operatorname{argmax}}\, P(Y = k|\boldsymbol{X} = \boldsymbol{x}) \tag{3.7}$$

This is called the Bayes classifier and predicts the most likely class, given the measurements of the variables $\boldsymbol{X} = \boldsymbol{x}$. Using the expression (A.1) for Bayes theorem we get:

$$\mathrm{P}(Y = k|\boldsymbol{X} = \boldsymbol{x}) = \frac{f_k(\boldsymbol{x})\pi_k}{\sum_{l=1}^{K} f_l(\boldsymbol{x})\pi_l}. \tag{3.8}$$

Assume that each class density is modeled by a p-dimensional normal distribution:

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}}\exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\Big).$$

LDA arises in the special case when all the classes have the same covariance matrix $\Sigma_k = \Sigma$ for all k. Hence (3.8) is equal to:

$$\begin{aligned}\mathrm{P}(Y = k|\boldsymbol{X} = \boldsymbol{x}) =& \frac{\pi_k \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\Big)}{\sum_{l=1}^{K}\pi_l \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_l)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_l)\Big)} \\ =& \frac{\pi_k \exp\Big(-\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_k\Big)}{\sum_{l=1}^{K}\pi_l \exp\Big(-\frac{1}{2}\boldsymbol{\mu}_l^T\Sigma^{-1}\boldsymbol{\mu}_l + \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_l\Big)}.\end{aligned}$$

Since the Bayes classifier, (3.7), compares the above quantity across $k = 1, \ldots, K$ for $\boldsymbol{X} = \boldsymbol{x}$, the denominator is the same for all classes, and we therefor obtain:

$$\widehat{Y} = \underset{k}{\operatorname{argmax}}\, \pi_k \exp\Big(-\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_k\Big) = \underset{k}{\operatorname{argmax}}\, \pi_k h_k(\boldsymbol{x}).$$

As $\log(\cdot)$ is a monotone function, we can consider maximizing $\log(\pi_k h_k(\boldsymbol{x}))$ over $k = 1, \ldots, K$. We can define the rule:

$$\begin{aligned}\widehat{Y} &= \underset{k}{\operatorname{argmax}} \log\left(\pi_k \exp\Big(-\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_k\Big)\right) \\ &= \underset{k}{\operatorname{argmax}} -\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_k + \log\pi_k \\ &= \underset{k}{\operatorname{argmax}}\, \delta_k(\boldsymbol{x}).\end{aligned}$$

We call $\delta_k(\boldsymbol{x})$, $k = 1, \ldots, K$ the discriminant functions and these functions are linear in $\boldsymbol{x}$. Note

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \log\pi_k \tag{3.9}$$

is just an affine function of $\boldsymbol{x}$.

Since we do not know the parameters of the Gaussian distributions, we have to estimate them using our data:

$$\begin{aligned}\widehat{\pi}_k &= \frac{N_k}{N}, \\ \widehat{\boldsymbol{\mu}}_k &= \sum_{i:y_i=k}\frac{\boldsymbol{x}_i}{N_k}, \\ \widehat{\Sigma} &= \frac{1}{(N-K)}\sum_{k=1}^{K}\sum_{i:y_i=k}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^T,\end{aligned} \tag{3.10}$$

where N is the total number of observations, and N_k is the number of observations in the kth class.

The boundary between to classes k and l is the set $\{\boldsymbol{x}|\delta_k(\boldsymbol{x}) = \delta_l(\boldsymbol{x})\}$. This is the same as the amount of $\boldsymbol{x}$ values, which have equal probability of being in class k or in class l. This means that:

$$\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_l) - \frac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_l)^T\Sigma^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_l) + \log\left(\frac{\pi_k}{\pi_l}\right) = 0. \tag{3.11}$$

In the case where we have two classes, $k = 1$ and $l = 2$, we can use equation (3.11) to classify. Using the estimates for Σ, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, π_1 and π_2 from above, and letting:

$$a_0 = \log\left(\frac{N_1}{N_2}\right) - \frac{1}{2}(\widehat{\boldsymbol{\mu}}_1 + \widehat{\boldsymbol{\mu}}_2)^T\widehat{\Sigma}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_2) \quad \text{and}$$
$$(a_1, ..., a_p)^T = \widehat{\Sigma}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_2).$$

Then an observation will be classified as class 1, if $a_0 + \sum_1^p a_j x_j > 0$ and otherwise class 2. This indicates that LDA learns a linear decision boundary.

In the following we consider an example of visualizing the LDA decision boundaries given the `Satellite` data described in chapter 1.

Example 1
In this example we use the `pca()` function obtained from the R-package `MASS` to perform LDA on the `Satellite` data. From the description of the data we know that it consists of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multi-spectral values. In R-code 3.1 the used packages and data is loaded.

```
library(MASS)
library(mlbench)
data(Satellite)
```

Code 3.1. R-packages and data.

Since we want to visualize LDA decision boundaries in the two dimensional space, we do not perform LDA directly on the `Satellite` data, since this is of dimension 36. Hence we first apply Principal Component Analysis (PCA) to the `Satellite` data and then use the first two principal components to perform LDA. In R-code 3.2 we use the function `princomp()` from the R-packed `stats` to perform PCA on the data, and from the estimate we extract the first two principal components.

```
pca <- princomp(Satellite[,-37], cor=T)
pcComp <- pca$scores
pcComp1 <- pcComp[,1] # principal component 1 scores
pcComp2 <- pcComp[,2] # principal component 2 scores
```

Code 3.2. Estimating the values of the first and second principal components.

The number of principal components is equal to the number of variables in the data set. The first principal component has the largest possible variance which indicate

that it explain as much of the variability in the data as possible. The succeeding principal components are orthogonal on the preceding principal components, and they have in turn the largest variance possible. Since we are concentrating on the first two principal components we are interested in how much of the total variability they explain. In R-code 3.3 we calculate the cumulative variance by using the standard deviations which are estimated when applying the `princomp()` function to the `Satellite` data.

```
> sum(pca$sdev[1:2]^2/sum(pca$sdev^2))
[1] 0.8523605
```

Code 3.3. Estimating the cumulative variance of the first two principal components.

From the above R-code the cumulative variance given by the first two principal components is estimated to 85% of the total variance in data. This is a reasonable result for the following analysis.

In R-code 3.4 we use the first two principal components to perform LDA on the whole data set. Here we have chosen the prior probabilities of class membership to be equal for each class.

```
modelLDA <- lda(Satellite[,37] ~ pcComp1+pcComp2, prior=rep(1,6)/6)
> modelLDA
Call:
lda(Satellite[, 37] ~ pcComp1 + pcComp2, prior = rep(1, 6)/6)

Prior probabilities of groups:
   red soil    cotton crop     grey soil    damp grey soil  vegetation stubble
  0.1666667      0.1666667     0.1666667         0.1666667           0.1666667
very damp grey soil
          0.1666667

Group means:
                       pcComp1   pcComp2
red soil            -1.3232849 -1.413214
cotton crop          2.6234318 -8.797711
grey soil           -5.0174861  1.350332
damp grey soil      -0.6959837  2.142695
vegetation stubble   4.5544860  0.777061
very damp grey soil  2.7942530  3.068162

Coefficients of linear discriminants:
               LD1        LD2
pcComp1 -0.2350827 -0.3794722
pcComp2  0.6843989 -0.1406927

Proportion of trace:
   LD1    LD2
0.8542 0.1458
```

Code 3.4. Performing LDA using the first and second principal components.

As we can see above, a call to the function `lda()` returns the prior probability of each class, the class-specific means for each variable, the linear combination coefficients for each linear discriminant (a matrix which transforms observations to discriminant functions) and the amount of the between-group variance that is explained by each linear discriminant. Here we observe that the first linear discriminant explains more than 85% of the between-group variance in the satellite data.

In R-code 3.5 we first use the estimate `modelLDA` from R-code 3.4 to predict, and next we estimate the confusion matrix.

```
predictPCALDA <- predict(modelLDA)$class
cofmatPCALDA <- table(Satellite[,37], predictPCALDA)
> cofmatPCALDA
                  predictPCALDA
                 red  cotton crop grey damp grey veg. stubble very damp grey
  red           1187            1   43         2          300              0
  cotton crop     63          600    0         1           38              1
  grey soil       19            0 1211       119            0              9
  damp grey soil  15            0   92       399           15            105
  veg. stubble   104            4    0        20          479            100
  very damp grey   2            0   11       302           39           1331
```

Code 3.5. Estimating the confusion matrix.

After estimating the confusion matrix we can compute the correct percentage for each class which is seen in R-code 3.6. Here we also estimate the total correct percentage which is estimated to 78.17%.

```
> diag(prop.table(cofmatPCALDA, 1))
   red soil     cotton crop    grey soil    damp grey soil  vegetation stubble
  0.7742988       0.8534852    0.8917526         0.6373802           0.6775106

  very damp grey soil
           0.7652520
> # total proportion correct predicted classes
> sum(diag(prop.table(cofmatPCALDA)))
[1] 0. 0.7816628
```

Code 3.6. Computing the total percent correct.

Figure 3.1 show the LDA decision boundaries for the Satellite data when using the first two principal components. The transparent areas are the predicted classes.

By looking at figure 3.1 clearly some of the red soil observations (the red point) are predicted to be vegetation stubble (the turquoise area). Due to the confusion matrix in R-code 3.5 the amount of red soil observations that are predicted to be vegetation stubble is 300. In general there is a direct connection between figure 3.1 and the confusion matrix in R-code 3.5. ■

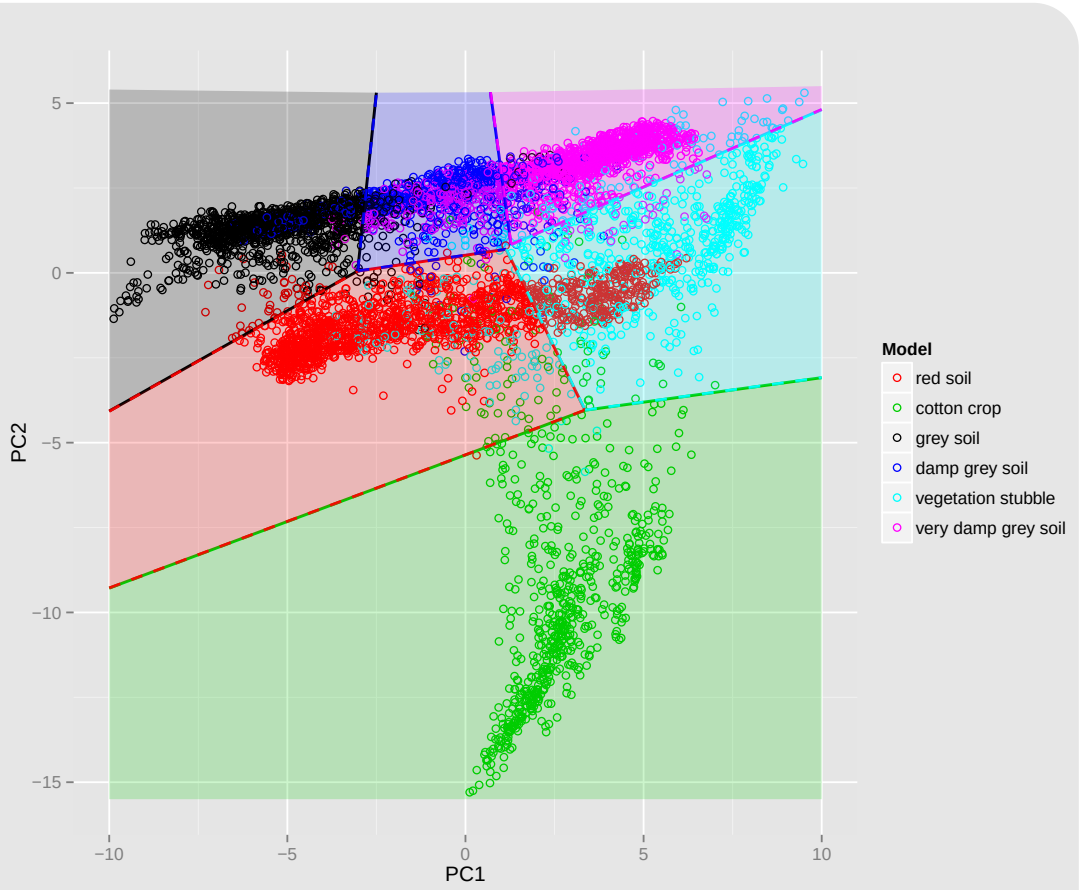

Figure 3.1. LDA decision boundaries for the Satellite data spanned by the first two principal components.

In the following example we perform LDA on the same `Satellite` data as in the example above. But this time we are not interested in visualizing the decision boundaries, and then we do not need to apply PCA on the data. Later on we want to compare LDA with other classification methods.

Example 2

In this example we perform LDA on the `Satellite` data using the same R-packages as in example 1. The aim is still to classify each observation in the data, given the multi-spectral values. As seen in R-code 3.7 we first estimate a random training set containing approximately 2/3 of the observations in the data and a test set containing the last 1/3. Next we perform LDA on the training set using the function `lda()`.

```
set.seed(5)
sampleSize <- sample(nrow(Satellite),4435)
trainSat <- Satellite[sampleSize,]
testSat <- Satellite[-sampleSize,]
ldaSat <- lda(classes~.,trainSat)
```

Code 3.7. Performing LDA on the training set.

In R-code 3.8 we use the test data to predict a LDA classifier. Here we have chosen the default settings for the prior probabilities of class membership which indicates that the class proportions for the training data are used, we also estimate the confusion matrix using the `table()` function.

```
> predictLDA <- predict(ldaSat, testSat)
> confMatrixLDA <- table(testSat[,37], predictLDA$class)
> confMatrixLDA

               red  cotton crop grey damp grey veg. stubble very damp grey
  red          439             0    9         6            3              1
  cotton crop    0           185    1         4           32              0
  grey           1             0  407        19            0              2
  damp grey      0             0   55        58            1             83
  veg. stubble   7             2    0        11          169             45
  very damp grey 0             0   21        41            4            394
```

Code 3.8. Estimating confusion matrix.

We see that the diagonal of the confusion matrix in R-code 3.8 is given by the largest number of every class. Hence for almost every class most observations are predicted correctly, especially the classes red soil, cotton crop and gray soil.

The per cent of correct predicted observations for each class is seen in Rcode 3.9.

```
> diag(prop.table(confMatrixLDA, 1))
   red soil    cotton crop    grey soil    damp grey soil  vegetation stubble
  0.9585153      0.8333333    0.9487179         0.2944162           0.7222222
very damp grey soil
          0.8565217
> ldaSatCorrect <- sum(diag(prop.table(confMatrixLDA)))
> ldaSatCorrect
[1] 0.826
```

Code 3.9. Estimating the accuracy of the prediction.

In R-code 3.9 we see that the total percent of correct observations when using LDA is 82.6%. ■

The data set is divided into two parts; a training set and a test set. The training set is used to fit our model, and the test data is used to test the fitted model. As we run through the test data, we have the probability of classifying right or wrong which yields the variable:

$$Z_i = \begin{cases} 1 & \text{if class ok} \\ 0 & \text{otherwise} \end{cases} .$$

If we have N test cases where $\pi = P(Z_i = 1)$, then

$$Z_+ = \sum_{i=1}^{N} Z_i \sim \mathrm{B}\left(N, \pi\right)$$

where $\mathrm{B}\left(\cdot\right)$ is the binomial distribution. Since the estimate of π is given by $\widehat{\pi} = \frac{Z_+}{N}$ we have the standard error:

$$\mathrm{SE}\left(\widehat{\pi}\right) = \sqrt{\frac{\widehat{\pi}(1-\widehat{\pi})}{N}} \tag{3.12}$$

and thereby the confidence interval $\widehat{\pi} \pm 2\mathrm{SE}\left(\widehat{\pi}\right)$

From R-code 3.9 in example 2 we have that the total percent of correct observations is 82.6%. This percentage is our $\widehat{\pi}$ and then we can use (3.12) to compute the standard error. Since the test set consists of 2000 observations we have:

$$\text{SE}(0.826) = \sqrt{\frac{0.826(1-0.826)}{2000}} = 0.008477,$$

and the confidence interval:

$$0.826 \pm 2\text{SE}(0.826) = 0.826 \pm 0.01695.$$

The standard error tells something about the uncertainty of our estimate. Later on we use the standard error and confidence interval to compare LDA with other classification method.

3.2.2 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is not really that much different from LDA, except that the covariance matrix can be different for each class. Hence we will estimate the covariance matrix Σ_k separately for each class k, $k = 1, 2, ..., K$. That is, it assumes that an observation from the kth class is of the form $\boldsymbol{X}_k \sim N(\boldsymbol{\mu}_k, \Sigma_k)$, where Σ_k is a covariance matrix for the kth class. Thus the log ratio of $\text{P}(Y = k|\boldsymbol{X} = \boldsymbol{x})$ and $\text{P}(Y = l|\boldsymbol{X} = \boldsymbol{x})$ is given by:

$$\begin{aligned}
\log \frac{\text{P}(Y=k|\boldsymbol{X}=\boldsymbol{x})}{\text{P}(Y=l|\boldsymbol{X}=\boldsymbol{x})} &= \log \frac{f_k(\boldsymbol{x})}{f_l(\boldsymbol{x})} + \log \frac{\pi_k}{\pi_l} \\
&= -\frac{1}{2}\Bigg(\log \frac{|\Sigma_k|}{|\Sigma_l|} + (\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_k) \\
&\quad - (\boldsymbol{x}-\boldsymbol{\mu}_l)^T \Sigma_l^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_l) \Bigg) + \log \frac{\pi_k}{\pi_l} \\
&= -\frac{1}{2}\Bigg(\log \frac{|\Sigma_k|}{|\Sigma_l|} + \left(\boldsymbol{x}^T \Sigma_k^{-1} \boldsymbol{x} - 2\boldsymbol{x}^T \Sigma_k^{-1} \boldsymbol{\mu}_k + \boldsymbol{\mu}_k^T \Sigma_k^{-1} \boldsymbol{\mu}_k \right) \\
&\quad - \left(\boldsymbol{x}^T \Sigma_l^{-1} \boldsymbol{x} - 2\boldsymbol{x}^T \Sigma_l^{-1} \boldsymbol{\mu}_l + \boldsymbol{\mu}_l^T \Sigma_l^{-1} \boldsymbol{\mu}_l \right) \Bigg) + \log \frac{\pi_k}{\pi_l}, \\
&= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}\Bigg(\log \frac{|\Sigma_k|}{|\Sigma_l|} + \boldsymbol{x}^T \left(\Sigma_k^{-1} - \Sigma_l^{-1} \right) \boldsymbol{x} \\
&\quad - 2\boldsymbol{x}^T \left(\Sigma_k^{-1} \boldsymbol{\mu}_k - \Sigma_l^{-1} \boldsymbol{\mu}_l \right) + \left(\boldsymbol{\mu}_k^T \Sigma_k^{-1} \boldsymbol{\mu}_k - \boldsymbol{\mu}_l^T \Sigma_l^{-1} \boldsymbol{\mu}_l \right) \Bigg).
\end{aligned} \tag{3.13}$$

Unlike in equation (3.9), the quantity $\boldsymbol{x}$ appears as a quadratic function in equation (3.13). This is where QDA gets its name. By setting equation (3.13) equal to zero we obtain the following equality:

$$\begin{aligned}
&-\frac{1}{2}(\log|\Sigma_k| + (\boldsymbol{x}-\boldsymbol{\mu}_k)^T {\Sigma_k}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_k)) + \log \pi_k \\
&\qquad = -\frac{1}{2}(\log|\Sigma_l| + (\boldsymbol{x}-\boldsymbol{\mu}_l)^T {\Sigma_l}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_l)) + \log \pi_l.
\end{aligned}$$

We then get the quadratic discriminant function:

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T{\Sigma_k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \log\pi_k.$$

The decision boundary between every pair of classes k and l is described by a quadratic equation $\{\boldsymbol{x}|\delta_k(\boldsymbol{x}) = \delta_l(\boldsymbol{x})\}$. The following example illustrates the use of QDA on the `Sattellite` data.

Example 3
This example is based on the same training and test data as in example 2. In R-code 3.10 we first apply QDA on the training data using the function `qda()`, and next we estimate the confusion matrix using the classifier found by LDA. Here the prior probabilities is the class proportion for the training data.

```
qdaSat <- qda(classes~., trainSat)
predictQDA <- predict(qdaSat, testSat)
confMatrixQDA <- table(testSat[,37], predictQDA$class)
confMatrixQDA

                red  cotton crop grey damp grey veg. stubble very damp grey
  red           447             0    7         0            4              0
  cotton crop     0           222    0         0            0              0
  grey            3             4  403         9            4              6
  damp grey       0             5   56        43            3             90
  veg. stubble    4            14    0         1          197             18
  very damp grey  0             8   31        16           18            387
```

Code 3.10. Performing QDA on the satellite data and estimating the confusion matrix.

The diagonal of the confusion matrix in R-code 3.10 has almost the same values as the confusion matrix in example 2. We see that the classes red soil, cotton crop and vegetation stubble are the only ones that are predicted slightly better.

In R-code 3.11 we see the per cent of correctly predicted observations for each class when performing QDA.

```
> diag(prop.table(confMatrixQDA, 1))
   red soil    cotton crop    grey soil     damp grey soil  vegetation stubble
  0.9759825      1.0000000    0.9393939          0.2182741           0.8418803
very damp grey soil
          0.8413043
> # total percent correct
> qdaSatCorrect <- sum(diag(prop.table(confMatrixQDA)))
> qdaSatCorrect
[1] 0.8495
```

Code 3.11. Estimating accuracy of the prediction.

Due to R-code 3.11 the total accuracy is 84.95% when using QDA on the `Satellite` data. Since the total accuracy is 82.6% when using LDA, we find that QDA is slightly better than LDA.

Like as in the LDA case we can use (3.12) to calculate the standard error of QDA. Using that $\widehat{\pi} = 0.8495$ from R-kode 3.11 we have:

$$\text{SE}(0.8495) = \sqrt{\frac{0.8495(1-0.8495)}{2000}} = 0.0079953,$$

and the confidence interval:

$$0.8495 \pm 2\text{SE}(0.8495) = 0.8495 \pm 0.01599.$$

If we compare the two discrimination methods, we see that the total accuracy of QDA lies outside the confidences interval of LDA. This indicates that the estimated QDA classifier in this case is better that the LDA classifier. ■

LDA and QDA are very efficient and straightforward on data sets where the number of observations is greater than the number of variables, that is where $N > p$. However, problems arise when these methods are applied to a data set where $N << p$. One major problem is that the covariance matrix estimated by a sample of observations is singular and can not be inverted. More specific the number of observations for each class in a data set have to be larger than the number of variables when using LDA and QDA. We could use instead the generalized inverse (see definition A.1 in appendix A) but then the estimate would be very unstable due to lack of observations. Another problem is that direct matrix operations are very time consuming when the number of variables is very large. Hence we have to use another method when we want to classify on data sets where $N << p$.

3.3 Graphical lasso

In some data sets the number of observations are smaller than the number of variables, i.e. $N << p$ which means that we cannot use LDA and QDA as classification methods. Instead we introduced a method called *graphical lasso* derived by [Jerome Friedman, 2007] and [Rahul Mazumder, 2012]. This method is implemented in the R-package `glasso`. In this section we give a further description of the `glasso` algorithm.

Consider a sample matrix $X \in \mathbb{R}^{N \times p}$ whose rows $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ are independent observations each given by a p dimensional normal distribution with mean μ and unknown covariance matrix Σ. Our goal is to estimate the unknown covariance matrix Σ by considering the sparse precision matrix. The graphical lasso uses l_1-regularization to estimate the covariance matrix Σ under the assumption that the precision matrix $J = \Sigma^{-1}$ is sparse. We know from theorem 2.4 in section 2.4 that if the (i, j)-entry of J is zero, i.e. $j_{ij} = 0$, then X_i and X_j are conditionally independent given the rest of the variables $\boldsymbol{X}_{-i,-j}$. The `glasso` algorithm derives an estimate for both the spars precision matrix J and the inverse precision matrix $W = J^{-1}$ where W is an estimate of Σ.

Let S be the empirical covariance matrix:

$$S = \frac{1}{N-1} \sum_{i=1}^{N} (\boldsymbol{X}_i - \bar{\boldsymbol{X}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^T,$$

where $\bar{\boldsymbol{X}}$ is the empirical mean. Additionally, let $\|J\|_1$ denote the l_1-norm which is the sum of the absolute value of the entries in J, that is:

$$\|J\|_1 = \sum_{i=1}^{p}\sum_{j=1}^{p}|j_{ij}|.$$

Then l_1-regularized negative log-likelihood can be written as:

$$\min_J f(J) := -\log|J| + \operatorname{tr}(SJ) + \rho\|J\|_1, \tag{3.14}$$

where J is a SPD matrix and ρ is a regularization parameter controlling the amount of l_1 shrinkage.

The optimal solution to (3.14) is obtained by first differentiating with respect to J and setting it equal to zero. The derivative of the first two terms are obtained by using (57) and (104) in [Petersen and Pedersen, 2012], respectively. Hence it follows that:

$$-J^{-1} + S + \rho\Gamma = 0, \tag{3.15}$$

and since the l_1-norm is not differentiable at zero, we define a matrix Γ which is a matrix of component-wise signs of J:

$$\begin{aligned}\gamma_{ij} &= \operatorname{sign}(j_{ij}) \quad \text{if} \quad j_{ij} \neq 0\\ \gamma_{ij} &\in [-1,1] \quad \text{if} \quad j_{ij} = 0.\end{aligned}$$

The global stationary conditions of (3.15) require J_{ii} to be positive which implies that $\gamma_{ii} = 1$, i.e.:

$$w_{ii} = s_{ii} + \rho \tag{3.16}$$

where $i = 1, \ldots, p$ and $W = J^{-1}$ is the estimate of Σ.

The `glasso` algorithm uses a row/column method when solving 3.15. Hence the following partitioning of J is considered:

$$J = \begin{pmatrix} J_{11} & \boldsymbol{j}_{12} \\ \boldsymbol{j}_{21} & j_{22} \end{pmatrix}$$

where J_{11} is a $(p-1)\times(p-1)$ matrix, $\boldsymbol{j}_{12}$ is a $(p-1)\times 1$ vector, and j_{22} is a scalar. Note that $\boldsymbol{j}_{21}^T = \boldsymbol{j}_{12}$ and the matrices W, S and Γ are partitioned the same way. By using the properties of inverse block-partitioned matrices in [Petersen and Pedersen, 2012, Section 9.1.3], then the equality $W = J^{-1}$ can be written as:

$$\begin{aligned}\begin{pmatrix} W_{11} & \boldsymbol{w}_{12} \\ \boldsymbol{w}_{21} & w_{22} \end{pmatrix} &= \begin{pmatrix} J_{11} & \boldsymbol{j}_{12} \\ \boldsymbol{j}_{21} & j_{22} \end{pmatrix}^{-1} \\ &= \begin{pmatrix} \left(J_{11} - \frac{\boldsymbol{j}_{12}\boldsymbol{j}_{21}}{j_{22}}\right)^{-1} & -W_{11}\frac{\boldsymbol{j}_{12}}{j_{22}} \\ -\frac{\boldsymbol{j}_{21}}{j_{22}}W_{11} & \frac{1}{j_{22}} + \frac{\boldsymbol{j}_{21}W_{11}\boldsymbol{j}_{12}}{j_{22}^2} \end{pmatrix}\end{aligned} \tag{3.17}$$

If we consider the pth column of (3.15) where the rest of the columns are fixed, then we obtain the following equation:

$$-\boldsymbol{w}_{12} + \boldsymbol{s}_{12} + \rho\boldsymbol{\gamma}_{12} = \boldsymbol{0} \tag{3.18}$$

From (3.17) we see that $\boldsymbol{w}_{12}$ is given as:

$$\boldsymbol{w}_{12} = -W_{11}\frac{\boldsymbol{j}_{12}}{j_{22}} \tag{3.19}$$

and if we plug this into (3.18) we have:

$$W_{11}\frac{\boldsymbol{j}_{12}}{j_{22}} + \boldsymbol{s}_{12} + \rho\boldsymbol{\gamma}_{12} = \mathbf{0} \tag{3.20}$$

The `glasso` algorithm solves (3.20) for $\boldsymbol{\lambda} = \frac{\boldsymbol{j}_{12}}{j_{22}}$, and we end up with the following equality:

$$W_{11}\boldsymbol{\lambda} + \boldsymbol{s}_{12} + \rho\boldsymbol{\gamma}_{12} = \mathbf{0} \tag{3.21}$$

where $\gamma_{12} \in \text{sign}(\boldsymbol{\lambda})$ since $j_{22} > 0$. By looking at (3.21) we see that this is the stationarity equation for the l_1 regularized quadratic program:

$$\min_{\boldsymbol{\lambda}} \Big\{ \frac{1}{2}\boldsymbol{\lambda}^T W_{11}\boldsymbol{\lambda} + \boldsymbol{\lambda}^T \boldsymbol{s}_{12} + \rho\|\boldsymbol{\lambda}\|_1 \Big\} \tag{3.22}$$

where W_{11} is SPD and assumed to be fixed. This is the same as the lasso regression problem on the last variable given the rest with the exception that the cross-product matrix S_{11} is replaced by its current estimate W_{11}. The problem in (3.22) can be solved efficiently by using elementwise coordinate descent which exploits the sparsity in $\boldsymbol{\lambda}$. Having the estimate $\widehat{\boldsymbol{\lambda}}$ it is easy to obtain $\widehat{\boldsymbol{w}}_{12}$ through (3.19):

$$\widehat{\boldsymbol{w}}_{12} = -W_{11}\widehat{\boldsymbol{\lambda}} \tag{3.23}$$

From the lower right entry of (3.17) $\widehat{j}_{22}$ is obtained by:

$$\frac{1}{\widehat{j}_{22}} = w_{22} + \widehat{\boldsymbol{\lambda}}^T\widehat{\boldsymbol{w}}_{12} \tag{3.24}$$

Finally we need to know the estimate $\widehat{\boldsymbol{j}}_{12}$ which is recovered from $\widehat{\boldsymbol{\lambda}}$ and $\widehat{j}_{22}$ since $\widehat{\boldsymbol{j}}_{12} = \widehat{\boldsymbol{\lambda}}\widehat{j}_{22}$. Notice that when having solved for $\boldsymbol{\lambda}$ and updated $\boldsymbol{j}_{12}$ the `glasso` algorithm can move onto the next row/column. The deriving of $\boldsymbol{j}_{12}$ and j_{22} can be done at the end when the algorithm over all rows/columns has converged.

From the above results we express the `glasso` algorithm as follows.

Algorithm 3.7 (glasso algorithm)

1: **initialize:** $W = S + \boldsymbol{\rho}I$.
2: **repeat**
3: Cycle around the columns performing the following steps.
   a) Rearrange the rows/columns so that the target column is last (implicitly).
   b) Solve the lasso problem (3.22).
   c) Update the row/column (off-diagonal) of the covariance using $\widehat{\boldsymbol{w}}_{12}$ derived in (3.23).

d) Save the estimated $\widehat{\boldsymbol{\lambda}}$ in the matrix Λ.

4: **until** Convergence.

5: For every row/column compute the diagonal $\widehat{j}_{ii}$ using (3.24), and convert the Λ matrix to J.

3.4 ROC curves

When having a two-class prediction problem we could be interested in how well an estimated binary classifier performs. In this section we introduce a *receiver operating characteristic* (ROC) curve which visualizes the performance of a binary classifier. The following description is inspired by [Gareth James, 2013, Section 4.4.3] and [Wikipedia, 2015].

Consider a binary classification problem where the outcome is either positive, Po, or negative, Ne. The result of a classification has four possible outcomes. If the outcome and the predicted value both are Po, then it is called true positive, TP, if the outcome on the other hand is Ne, then it is called false positive FP. Thus the true negative, TN, appears when both the outcome and predicted value are negative, and the false negative, FN, appears when the predicted value is negative and the outcome is positive. The four outcomes can be formulated in a 2×2 confusion matrix as in table 3.1, where Po^* are the numbers of predicted positive, and Ne^* are the numbers of predicted negative.

		Predicted class		
		Positive	Negative	Total
Observed class	Positive	TP	FN	Po
	Negative	FP	TN	Ne
	Total	Po^*	Ne^*	$Po + Ne$

Table 3.1. Possible outcome for a binary classification.

From table 3.1 we obtain the following estimates:

- Sensitivity or true positive rate (TPR): The amount of positive, which is predicted positive.
$$\text{TPR} = \frac{TP}{Po} = \frac{TP}{TP + FN}$$
- Specificity or true negative rate (TNR): The amount of negative, which is predicted negative.
$$\text{TNR} = \frac{TN}{Ne} = \frac{TN}{FP + TN}$$

- False positive rate (FPR): The amount of negative which is predicted false positive.

$$\text{FPR} = \frac{FP}{Ne} = \frac{FP}{FP + TN} = 1 - \frac{TN}{FP + TN} = 1 - \text{Specificity}$$

- Accuracy (ACC): The proportion of correctly classified predictions.

$$\text{ACC} = \frac{TP + TN}{Po + Ne}$$

A ROC curve illustrates the performance of a binary classifier as the parameters of a classification rule is varied. The ROC curve is created by plotting the sensitivity (TPR) against the 1-specificity (FPR) for different cut-off values, which means that the curve is non-decreasing. Hence each point on the ROC curve represents a sensitivity/specificity pair which corresponds to a particular cut-off value. The TPR indicates the amount of correct positive results among all positive observations in a test set. The FPR on the other hand gives the amount of incorrect positive results among all negative observations in a test set. Notice that each prediction or confusion matrix yields a point on the ROC curve.

A perfect classification has a ROC curve which passes through the upper left corner or coordinate $(0, 1)$, representing a sensitivity and specificity of 100%. The diagonal joining the lower left corner and the upper right corner is referred to as the line of no-discrimination. Points landing on this line indicate a prediction that is no better than a completely random guess. Thus points above the diagonal represent a "good" classification, and points under the diagonal represent a "poor" classification. Note that poor prediction can be inverted to obtain a "good" prediction.

The area under the curve, AUC, is the performance of a classifier summarized over all possible cut-off values. A perfect ROC curve will go through the top left corner, so the larger the AUC the better the classifier. The way the ROC curve is constructed implies that the overall area can maximally be one. ROC curves are useful for comparing different classifiers, since they take into account all possible cut-off values.

Since classifications are often based on continuous variables, we can write the probability of belonging to a certain class as a function of a cut-off value C as $P_1(C)$ and the probability of not belonging to the class as $P_0(C)$. Hence the TPR is given as:

$$\text{TPR}(C) = \int_C^\infty P_1(x)dx,$$

and the FPR is given by:

$$\text{FPR}(C) = \int_C^\infty P_0(x)dx.$$

Now using the two integrals above the ROC curve plots $\text{TPR}(C)$ against $\text{FPR}(C)$ where C is a varying parameter. Figure 3.2 shows the two distributions for a varying cut-off value C.

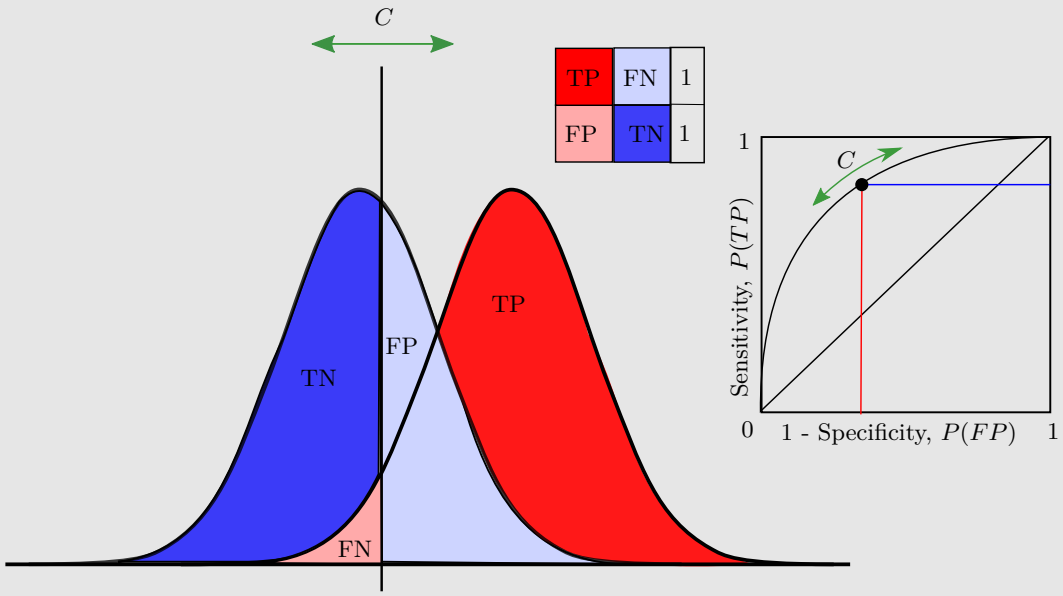

Figure 3.2. Distributions of TPR and FPR for a varying cut-off value C.

By moving the cut-off value in figure 3.2 to the right the amount of FN increases and the amount of FP decreases. The actual shape of the ROC cure is given by how much the two distributions overlap.

When estimating a ROC curve for a binary classifier we use the `roc()` function from the R-package `pROC`. In example 2 we applied LDA on the `Satellite` data, and by using these results we can estimate a ROC curve for each class in the data. For example a ROC curve for the red soil class is constructed by the binary classification where observations either belongs to the red soil class or not. The ROC curve for the red soil class is estimated in R-code 3.12, where we use the posterior probabilities from the estimated classification variable, `predictLDA`, in R-code 3.8.

```
> postLDA <- predictLDA$posterior
> rocLDA <- roc(test[,37]=="red soil", postLDA[,1])
> rocLDA

Call:
roc.default(response = test[, 37] == "red soil", predictor = postLDA[, 1])

Data: postLDA[, 1] in 2470 controls (test[, 37] == "red soil" FALSE) < 748
    cases (test[, 37] == "red soil" TRUE).
Area under the curve: 0.9973
```

Code 3.12. Estimating the ROC curve for the red soil class.

From the call of the estimated ROC curve in R-code 3.12 we observe that the binary classification is given by 2470 observations not being the red soil class (controls) and 748 observations being the red soil class (case). We also observe that the AUC is 0.9973, which is quite good. In the same way as with the red soil class we estimate a ROC curve for each of the remaining classes in the `Satellite` data. Figure 3.3 visualize the ROC curves for each class when LDA is applied on the `Satellite` data.

The AUC for each class is listed in table 3.2, where we have also listed the sensitivity

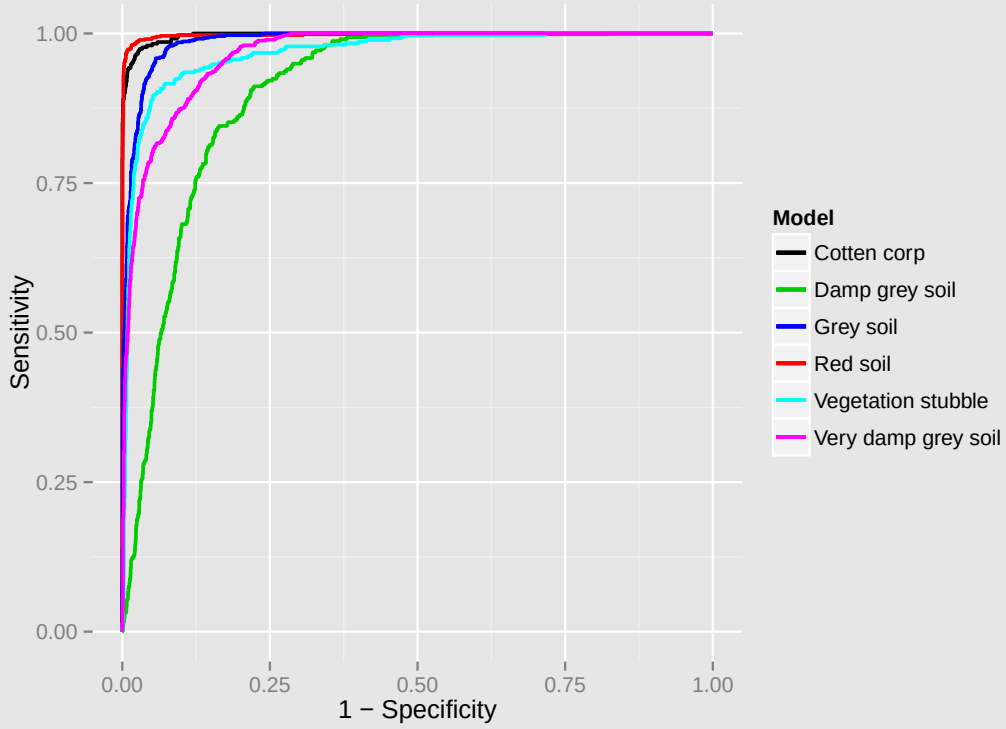

Figure 3.3. ROC curves for LDA based on `Satellite` data.

of each class from R-code 3.9. If we compare the sensitivity with the AUC, we see that the values do not go together, i.e. when AUC is high, the sensitivity is not necessarily also high. For example does cotton corp have an AUC at 0.997 but only a sensitivity at 0.833. This is because, we have to take the prior probabilities into consideration. The prior probabilities when using LDA are given in R-code 3.13 where we observe that the prior probability for cotton corp is 0.109.

```
> ldaSat$prior
    red soil     cotton crop    grey soil   damp grey soil  vegetation stubble
  0.24239008      0.10845547   0.20947012       0.09673055          0.10665163
very damp grey soil
         0.23630214
```

Code 3.13. Prior probabilities when using LDA.

Since cotton corp has low prior probability, it also have low sensitivity which we also observe in table 3.2.

In same way we can estimate the ROC curves for each class where apply QDA to the `Satellite` data. Figure 3.4 illustrate the ROC curves for each class. We observe that the ROC curves for QDA do not chance much compared to the ROC curves for LDA

	AUC	Class sensitivity
red soil	0.9973	0.9498
cotton crop	0.997	0.833
grey soil	0.9868	0.9044
damp grey soil	0.9065	0.6345
vegetation stubble	0.9679	0.7735
very damp grey soil	0.9662	0.7609

Table 3.2. AUC and sensitivity for each class when using LDA.

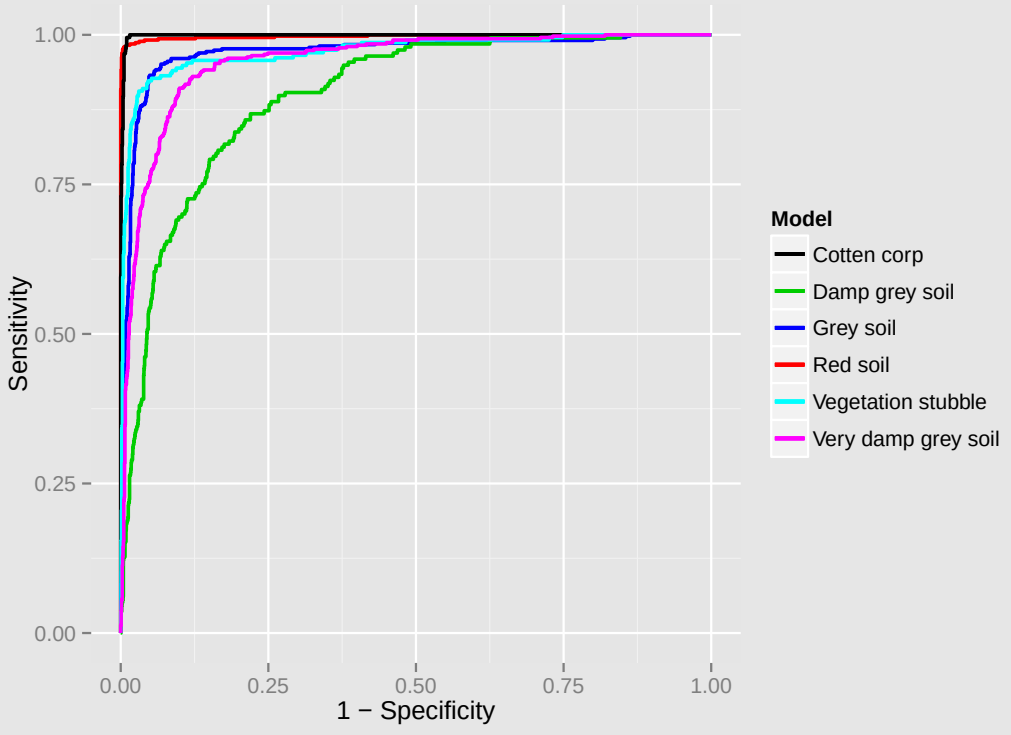

Figure 3.4. ROC curves for QDA based on the `Satellite` data.

In table 3.3 we have listed the AUC and sensitivity for each class when performing QDA on the data. If we compare AUC and sensitivity, we see that the largest gap is for the damp gray soil class which is larger than in the LDA case. Also we have that the values of AUC and sensitivity do not go together.

	AUC	Class sensitivity
red soil	0.9957	0.976
cotton crop	0.9986	1.00
grey soil	0.9749	0.9394
damp grey soil	0.8999	0.2183
vegetation stubble	0.9715	0.8419
very damp grey soil	0.9571	0.8495

Table 3.3. AUC and sensitivity for each class when using QDA.

Model selection in graphical models 4

In this chapter we describe an algorithm for doing efficient and fast model selection in graphical models. The most optimal method is based on likelihood ratio test which for example is done in the R-package `gRim`, but this method is going to be prohibitively slow as the number of variables increases.

Consider a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ of N observations of $\boldsymbol{X}_i \sim \mathrm{N}_p(\boldsymbol{\mu}, \Sigma)$ where $i = 1, \ldots, N$. Let S denote the sample covarians matix:

$$S = \frac{1}{N-1} \sum_{i}^{N} (\boldsymbol{X}_i - \bar{\boldsymbol{X}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^T$$

where $\bar{\boldsymbol{X}}$ is the sample mean. Our model is represented by a GMRF with regard to a graph $\mathcal{G} = (V, E)$ where the vertices $V = \{1, \ldots, p\}$ represent the variables. Initially, this is the empty graph.

Before describing the algorithm we first define the matrix $\mathcal{T} = \{t_{ab}\}_{a,b \in V}$, which contains all t-test quantities for pairwise independence. These quantities are not exactly the t-test since we use the numeric value of the t-score. Initially, this is given by:

$$t_{ab} = \frac{|r_{ab}|\sqrt{N-2}}{\sqrt{1 - r_{ab}^2}}$$

where $r_{ab} = \widehat{\mathrm{corr}}(X_a, X_b)$ is the estimated correlation between variables a and b. The following algorithm is constructed such that it generates random models. Let crit denote a standard normal quantile depending on the level of significance, let Eli denote eligible edges for removing/adding and let $\mathcal{G} = (V, E)$ be the empty graph. Initially, these are the edges with a t-value below crit.

Algorithm 4.1 (Random algorithm)
Input: $\mathcal{G} = (V, E)$ and $\mathcal{T} = \{t_{ab}\}_{a,b \in V}$.
1: **initialize:** Add Eli $= \{e_{ab}\}_{a,b \in V}$ where

$$e_{ab} = e_{ba} = \begin{cases} \text{TRUE} & \min\{t_{ab}, t_{ba}\} > \text{crit} \\ \text{FALSE} & \min\{t_{ab}, t_{ba}\} \leq \text{crit} \end{cases}$$

2: **repeat**
3: choose $\{a, b\}$ randomly such that $e_{ab} = \text{TRUE}$
4: **if** $\{a, b\} \notin E$ **then**
5: add $\{a, b\}$ and update $\mathcal{G}, \mathcal{T}$ and Eli.
6: **end if**
7: **if** $\{a, b\} \in E$ **then**
8: remove $\{a, b\}$ and update $\mathcal{G}, \mathcal{T}$ and Eli.
9: **end if**
10: **until** $e_{ab} = \text{FALSE}$ for all $\{a, b\}$.

Algorithm 4.1 contains various generic components that are in use during every iteration. Updating the graph $\mathcal{G}$ is straightforward since we only have to add or remove an edge. But we have to choose how to update the matrix $\mathcal{T}$ with new t-values. After the matrix $\mathcal{T}$ is updated, we can update the Eli matrix with new Eli values. When updating the Eli values we determine which edges are eligible for adding or removing. The following algorithm shows how to update the Eli matrix when an edge $\{a_0, b_0\}$ is removed or added. In the following algorithm $\emptyset$ is the empty set.

Algorithm 4.2 (Updating Eli)
$\{a_0, b_0\}$ is added or deleted.
Input: Eli $= \{e_{ab}\}_{a,b \in V}$.
1: for each edge $\{a, b\} \in E$ such that $\{a, b\} \cap \{a_0, b_0\} \neq \emptyset$
2: **if** $\min\{t_{ab}, t_{ba}\} < \text{crit}$ **then**
3: $e_{ab} = e_{ba} = \text{TRUE}$.
4: **else**
5: $e_{ab} = e_{ba} = \text{FALSE}$
6: **end if**
7: for each edge $\{a, b\} \notin E$ such that $\{a, b\} \cap \{a_0, b_0\} \neq \emptyset$
8: **if** $\min\{t_{ab}, t_{ba}\} > \text{crit}$ **then**
9: $e_{ab} = e_{ba} = \text{TRUE}$.
10: **else**
11: $e_{ab} = e_{ba} = \text{FALSE}$
12: **end if**

When an edge $\{a_0, b_0\}$ is added or removed we only have to update the rows and columns for a_0 and b_0 in the Eli matrix . In the case of removing edges we consider the edge $\{a, b\} \in E$ and if $\min\{t_{ab}, t_{cba}\} < \text{crit}$ then $e_{ab} = e_{ba} = \text{TRUE}$ and else FALSE. An alternative for removing an edge could be $\max\{t_{ab}, t_{ba}\} < \text{crit}$ where both of the t-tests is lesser than the crit value. Almost the same goes for adding edges where we consider the edge $\{a, c\} \notin E$ and if $\min\{t_{ab}, t_{ba}\} > \text{crit}$ then $e_{ab} = e_{ba} = \text{TRUE}$ and else FALSE. An alternative for adding an edge would be $\max\{t_{ab}, t_{ba}\} > \text{crit}$.

In the case where we are not interested in generating random models, we have the following greedy algorithm. The algorithm is initialised by the empty graph and the same matrix $\mathcal{T}$ as in algorithm 4.1.

Algorithm 4.3 (Greedy algorithm)
Input: $\mathcal{T} = \{t_{ab}\}_{a,b \in V}$.

1: **initialize:** Start by the empty graph $\mathcal{G} = (V, E)$, where $E = \emptyset$.
2: find $\{a, b\} \in E$ such that $t_{a_0b_0} = \min_{(a,b)\in E} t_{ab}$
3: **if** $t_{a_0b_0} < \text{crit}$ **then**
4: remove the edge and update $\mathcal{G}$ and $\mathcal{T}$.
5: go to step 2.
6: **else**
7: find $\{a_0, b_0\} \notin E$ such that:

$$z_{a_0b_0} = \min\{t_{a_0b_0}, t_{b_0a_0}\} = \max_{\{a,b\}\notin E} \min\{t_{ab}, t_{ba}\}$$

8: **if** $z_{a_0b_0} > \text{crit}$ **then**
9: add edge and update $\mathcal{G}$ and $\mathcal{T}$.
10: Go to step 2.
11: **else**
12: **break**
13: **end if**
14: **end if**

This greedy algorithm also contains the components $\mathcal{G}$ and matrix $\mathcal{T}$ which are updated the same way as in algorithm 4.1. Algorithm 4.3 also haves some alternatives for removing/adding edges. An alternative when removing an edge which is considered in step 2 could be $\max\{t_{a_0b_0}, t_{b_0a_0}\} = \min_{\{a,b\}\in E} \max\{t_{ab}, t_{ba}\}$. When adding an edge which is considered in step 7 an alternative could be $t_{a_0b_0} = \max_{\{a,b\}\notin E} t_{ab}$.

The thing we need to explain is how the test statistics in the $\mathcal{T}$ matrix actually are updated during an iteration in one of the algorithms. The traditional way would be to perform likelihood ratio test where we get one value for the test statistic i.e. $t_{ab} = t_{ba}$, and this is the likelihood ratio test statistic for removing an edge. To determine if an edge have to be added is done by performing likelihood ratio for

removing the added edge. The problem when doing likelihood ratio test is that when the number of variables is large it can not be done in "finite" time. This is because we have to fit a new model using for example iterative proportional scaling (IPS), every time an edge is added/removed. If e.g. $p = 100$ we would have to fit $p(p-1)/2 = 499.500$ models.

An alternative to likelihood ratio could be to use AIC or BIC but since these make use of the deviance we are running into the same problem as above, where we have to fit a new model every time an edge is added/removed.

Since we want to avoid fitting a whole new model every time an edge is added/removed, we introduce an alternative to likelihood ratio test. In the following another way of updating the t-values is suggested, namely the t-test statistics which are considered for a given variable and its neighbors. We construct a t-test for removing an edge to a neighbor or adding an edge to a non-neighbor. Here we use (3.6) from section 3.1.2 to update the t-values where we consider a regression on the neighbors. In case of adding an edge the neighbor set is extended by one node.

Now we consider an efficient method when updating the t-values in the $\mathcal{T}$ matrix, where we do not have to use matrix inversion. Instead we take the advantage of using an already known matrix $(D^TD)^{-1}$ in (3.6) from section 3.1.2. When adding an edge we need the $(D^TD)^{-1}$ matrix which has one dimension higher, whereas when removing an edge we use the actual $(D^TD)^{-1}$ matrix. Hence every time an edge a—b is removed/added then:

- t_{ac} have to be updated for $c = 1, 2, \ldots, p$.
- t_{bc} have to be updated for $c = 1, 2, \ldots, p$.

This indicates that we only need to update two rows and not the entire $\mathcal{T}$ matrix whenever an edge is removed/added. The above method of updating the t-values is derived in the following.

Given a sample of size N, where each sample follows a multivariate normal distribution:

$$\boldsymbol{X}_i \sim \mathrm{N}_p(\boldsymbol{\mu}, \Sigma), \quad i = 1, 2, \ldots, N,$$

and the samples are independent. The matrix $J = \Sigma^{-1}$ is specified through the graph $\mathcal{G}$.

consider the ath element of $\boldsymbol{X}_i$, i.e. X_{ia}, and its neighbors $\boldsymbol{X}_{i\mathrm{Ne}(a)}$ where $\mathrm{Ne}(a)$ can be considered as a function that gives us the neighbors of X_{ia}. By using (2.6) we derive the following conditional distribution:

$$X_{ia}|\boldsymbol{X}_{i\mathrm{Ne}(a)} = \boldsymbol{x}_{i\mathrm{Ne}(a)} \sim \mathrm{N}\left(\alpha_a + \boldsymbol{x}_{i\mathrm{Ne}(a)}^T\boldsymbol{\beta}_a, \sigma_a^2\right) \tag{4.1}$$

where $\boldsymbol{\beta}_a = (\beta_{ab})_{b\in\mathrm{Ne}(a)}$ and α_a is an intercept. The conditional distributions are independent for $i = 1, 2, \ldots, N$. If we remove the edge $\{a, b\} \in E$ then $\beta_{ab} = 0$, so we want to test whether this is acceptable.

By centralizing it can be assumed that $\alpha_a = 0$, i.e.:

$$X_{ia} := X_{ia} - \frac{1}{N}\sum_{j=1}^{N} X_{ja}, \quad i = 1, 2, \ldots, N, \quad a = 1, 2, \ldots, p$$

Hence (4.1) is given as:

$$\boldsymbol{X}_a | X_{\mathrm{Ne}(a)} \sim \mathrm{N}_N\Big(X_{\mathrm{Ne}(a)}\boldsymbol{\beta}_a, \sigma_a^2 I\Big) \tag{4.2}$$

According to (3.2) the estimate of $\boldsymbol{\beta}_a$ in the model (4.2) is:

$$\widehat{\boldsymbol{\beta}}_a = \Big(X_{\mathrm{Ne}(a)}^T X_{\mathrm{Ne}(a)}\Big)^{-1} X_{\mathrm{Ne}(a)}^T \boldsymbol{X}_a.$$

By defining the matrix $S := X_V^T X_V$ where V is the set of vertexes in the graph, we can rewrite the estimate $\widehat{\boldsymbol{\beta}}_a$ as:

$$\widehat{\boldsymbol{\beta}}_a = S_{\mathrm{Ne}(a),\mathrm{Ne}(a)}^{-1} S_{\mathrm{Ne}(a),a},$$

and due to (3.3) this estimate has variance:

$$\mathrm{Var}\left[\widehat{\boldsymbol{\beta}}_a\right] = \sigma_a^2 S_{\mathrm{Ne}(a),\mathrm{Ne}(a)}^{-1}$$

Since this equation contains σ_a^2, we have to calculate the estimate of σ_a^2 which according to (3.4) is given by:

$$\begin{aligned}
\widehat{\sigma}_a^2 &= \frac{1}{N-l}\Big\|\boldsymbol{X}_a - X_{\mathrm{Ne}(a)}\Big(X_{\mathrm{Ne}(a)}^T X_{\mathrm{Ne}(a)}\Big)^{-1} X_{\mathrm{Ne}(a)}^T \boldsymbol{X}_a\Big\|^2 \\
&= \frac{1}{N-l}\Big\|\boldsymbol{X}_a - H_a \boldsymbol{X}_a\Big\|^2 \\
&= \frac{1}{N-l}\Big(\boldsymbol{X}_a - H_a\boldsymbol{X}_a\Big)^T\Big(\boldsymbol{X}_a - H_a\boldsymbol{X}_a\Big) \\
&= \frac{1}{N-l}\Big(\boldsymbol{X}_a^T\boldsymbol{X}_a - \boldsymbol{X}_a^T H_a \boldsymbol{X}_a - \boldsymbol{X}_a^T H_a^T \boldsymbol{X}_a + \boldsymbol{X}_a^T H_a^T H_a \boldsymbol{X}_a\Big).
\end{aligned}$$

Then by using definition 3.4 we have that:

$$\begin{aligned}
\widehat{\sigma}_a^2 &= \frac{1}{N-l}\Big(\boldsymbol{X}_a^T\boldsymbol{X}_a - \boldsymbol{X}_a^T H_a \boldsymbol{X}_a\Big) \\
&= \frac{1}{N-l}\Big(S_{a,a} - S_{a,\mathrm{Ne}(a)} S_{\mathrm{Ne}(a),\mathrm{Ne}(a)}^{-1} S_{\mathrm{Ne}(a),a}\Big)
\end{aligned}$$

where l is the number of neighbors plus one. We have to plus one due to the intercept in (4.1) which was omitted by centralizing. We have that

$$\widehat{\boldsymbol{\beta}}_a = \Big(\widehat{\beta}_{ab}\Big)_{b\in\mathrm{Ne}(a)}$$

where $\widehat{\beta}_{ab}$ is a regression coefficient to one of the neighbors of $\boldsymbol{X}_a$. Since we want to test if $\boldsymbol{X}_a$ and X_b are conditionally independent given $\mathrm{Ne}\,(a) \setminus \{b\}$, i.e. if $\beta_{ab} = 0$, we use the test quantity (3.6). Thus we have that:

$$t_{ab} = \frac{\widehat{\beta}_{ab}}{\widehat{\sigma}_a\sqrt{\Big(S_{\mathrm{Ne}(a),\mathrm{Ne}(a)}\Big)_{bb}^{-1}}} \sim t(n-k), \quad b \in \mathrm{Ne}\,(a)\,, \tag{4.3}$$

where $\widehat{\boldsymbol{\beta}}_a$ and $\widehat{\sigma}_a^2$ are independent.

Note that t_{ab} in (4.3) only depends on the correlation matrix of X_V, i.e. subsequently we assume that $S = S_{V,V}$ is a correlation matrix.

For each $a \in V$ we store the matrix $W_a = (S_{\mathrm{Ne}(a),\mathrm{Ne}(a)})^{-1}$ and t-values, t_{ab} where $b \neq a$, which indicate if the edge can be removed or added. In the case of removing an edge $\{a, b\}$ we use (4.3) to test for independence. When adding an edge $\{a, b\}$ to the graph we add the node b to $\mathrm{Ne}(a)$, and do regression on $\mathrm{Ne}(a) \cup \{b\}$ which is given as:

$$\boldsymbol{X}_a = X_{\mathrm{Ne}(a)\cup\{b\}}\tilde{\beta}_a + \varepsilon$$

where $\boldsymbol{X}_a$ is centralized. Thus we need to extend W_a by adding the node b so that we obtain $S^{-1}_{\mathrm{Ne}(a)\cup\{b\},\mathrm{Ne}(a)\cup\{b\}}$ and use this to test if a and b are independent given $\mathrm{Ne}(a)$. This implies that we have to find the matrix:

$$\begin{pmatrix} S_{\mathrm{Ne}(a),\mathrm{Ne}(a)} & S_{\mathrm{Ne}(a),b} \\ S_{b,\mathrm{Ne}(a)} & S_{bb} \end{pmatrix}^{-1} \tag{4.4}$$

where we have added the node b. When we use the correlation matrix, we have $r_{\mathrm{Ne}(a)b} = \widehat{\mathrm{corr}}(\mathrm{Ne}(a), b)$ which is the estimated correlation between the neighbors of a and the node b. Since S is the correlation matrix, then $S_{bb} = 1$, and by defining h and α_{ab} as:

$$\begin{aligned} h &:= W_a r_{\mathrm{Ne}(a)b} \\ \alpha_{ab} &:= \frac{1}{1 - r^T_{\mathrm{Ne}(a)b} W_a r_{\mathrm{Ne}(a)b}} \end{aligned}$$

we can express (4.4) as:

$$\begin{pmatrix} W_a^{-1} & r_{\mathrm{Ne}(a)b} \\ r^T_{\mathrm{Ne}(a)b} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} W_a + \alpha_{ab} h h^T & -\alpha_{ab} h \\ -\alpha_{ab} h^T & \alpha_{ab} \end{pmatrix} \tag{4.5}$$

To prove that 4.5 is a proper inverse we have to show that multiplied with its non-inverse yields the identity matrix. Hence we have the following computation:

$$\begin{aligned} &\begin{pmatrix} W_a^{-1} & r_{\mathrm{Ne}(a)b} \\ r^T_{\mathrm{Ne}(a)b} & 1 \end{pmatrix} \begin{pmatrix} W_a + \alpha_{ab} h h^T & -\alpha_{ab} h \\ -\alpha_{ab} h^T & \alpha_{ab} \end{pmatrix} \\ &= \begin{pmatrix} I + \alpha_{ab}(W_a^{-1} h h^T - r_{\mathrm{Ne}(a)b} h^T) & \alpha_{ab}(-W_a^{-1} h + r_{\mathrm{Ne}(a)b}) \\ r^T_{\mathrm{Ne}(a)b} W_a + \alpha_{ab}(r^T_{\mathrm{Ne}(a)b} h h^T - h^T) & \alpha_{ab}(-r^T_{\mathrm{Ne}(a)b} h + 1) \end{pmatrix} \\ &= \begin{pmatrix} I + \alpha_{ab}(W_a^{-1} W_a r_{\mathrm{Ne}(a)b} h^T - r_{\mathrm{Ne}(a)b} h^T) & \alpha_{ab}(-W_a^{-1} W_a r_{\mathrm{Ne}(a)b} + r_{\mathrm{Ne}(a)b}) \\ r^T_{\mathrm{Ne}(a)b} W_a - h^T \alpha_{ab}(1 - r^T_{\mathrm{Ne}(a)b} W_a r_{\mathrm{Ne}(a)b}) & \alpha_{ab}(1 - r^T_{\mathrm{Ne}(a)b} W_a r_{\mathrm{Ne}(a)b}) \end{pmatrix} \\ &= \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} = I \end{aligned}$$

Since (4.5) is a proper inverse we now have an efficient method of estimation the matrix in (4.4) by using the already known matrix W_a.

I the above we have explained how to update the $\mathcal{T}$ matrix in a fast an efficient way. Hence we now know how to handle all the various generic components when running the two algorithms 4.1 and 4.3. Since we want to use a classification strategy inspired by Random Forests, we will concentrate on the random algorithm 4.1. During every iteration the Eli matrix is updated as described in algorithm 4.2. In this algorithm the edges in the graph are selected randomly, and therefor we call it the headlong method (HLM). Algorithm 4.1 is implemented in the R-cript `HLM.R`, where the function `init()` is performing the initialization, and the function `step()` is performing the repetitions in the algorithm.

To see the efficiency of the HLM we compare it to the `forward()` function from the R-packaged `gRim` which uses likelihood ratio test. The methods are both applied to the `Satellite` data, and for convenience we only concentrate on data for the red soil class. In R-code 4.1 we consider the HLM where we start by applying the `init()` function to the data. This yields an initial model `M0` which we use in the `step()` function. This process goes through 359 iterations i 0.25 seconds.

```
d <- trainSat[trainSat[,37]=="red soil",-37]
m0 <- init(d)
> system.time(
+ m <- step(m0)
+ )
  bruger   system forløbet
    0.25     0.00     0.25
> m$it
[1] 359
```

Code 4.1. Applying the HLM to the red soil class.

In R-code 4.2 we first apply the `cmod` function to the model found by the HLM in R-code 4.1 which estimate a model `fitm`. We then apply the `forward()` function to `fitm` where we only add on edge.

```
> fitm <- cmod(edgeL(m),d)
> system.time(
+ forwm <- forward(fitm,criterion="test",alpha=.01,type="unrestricted",search=
    "headlong", steps=1)
+ )
. FORWARD: type=unrestricted search=headlong, criterion=test, alpha=0.01
. Initial model: is graphical=TRUE is decomposable=FALSE
  p.value    0.0006 Edge added: x.32 x.17
  bruger   system forløbet
    2.33     0.00     2.33
```

Code 4.2. Applying `gRim` on the red soil class.

This process takes 2.33 seconds to do one iteration, which is a long time compared with the HLM that went through 359 iterations. Hence we can conclude that the HLM is much faster.

Applications 5

In this chapter we use the HLM described in chapter 4 to perform model selection and classification. The method is illustrated on the two data sets described in chapter 1, and then benchmarked against LDA, QDA and glasso.

We start by considering the `Satellite` data which contain 6435 observations and 37 variables where the last one is a classification variable of six classes. This data set has been divided into a training set of 4435 observations and a test set of 2000 observations.

Initially, we start by applying the HLM to the training data to estimate a suitable model for each class. The models found by the HLM are estimated by the `cmod()` function which yields a precision matrix for each class. Then the idea is to perform some sort of random forest on the training set by estimating R random models all based on the HLM and taking the average. Hence the conditional density of an observation from the test set given the `red soil` class is given as:

$$\frac{1}{R}\sum_{i=1}^{R}|J_{i1}|^{\frac{1}{2}}\exp\Big(\boldsymbol{h}_{i1}^{T}\boldsymbol{x}-\frac{1}{2}\boldsymbol{x}^{T}J_{i1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{h}_{i1}^{T}\boldsymbol{\mu}_{1}\Big)=f(\boldsymbol{x}|\texttt{red soil}).$$

Here J_{i1} is the precision matrix and $\boldsymbol{h}_{i1}$ is the potential vector of the ith estimated model based on the training data for the `red soil` class. Additional, $\boldsymbol{\mu}_1$ is the mean vector for the variables based on the training data for the `red soil` class. Since there are six classes in the data, we have to estimate a conditional density given each class.

If we assume that the probability of belonging to one of the six classes is equal, then the density of each class is given as:

$$\begin{aligned}f(\texttt{red soil})=f(\texttt{cotten crop})=f(\texttt{grey soil})=f(\texttt{veg. stubble})\\=f(\texttt{damp grey soil})=f(\texttt{very damp grey soil})=\frac{1}{6}\end{aligned}$$

Due to (A.1) this implies that the posterior density of the `red soil` class given an observation from test data is:

$$f(\texttt{red soil}|\boldsymbol{X}=\boldsymbol{x})=\frac{f(\boldsymbol{x}|\texttt{red soil})}{f(\boldsymbol{x}|\texttt{red soil})+\cdots+f(\boldsymbol{x}|\texttt{very damp grey soil})}\tag{5.1}$$

where the denominator is the sum of the conditional densities given each class. Thus we have six of these posterior densities, and the one with the highest value decides the class of an observation from the test set.

If we estimate 100 model, i.e. $R = 100$, for each class, we obtain the confusion matrix seen in R-code 5.1. This confusion matrix is estimated by taking the actual class against the predicted class based on test data.

```
confMatrixHLM <- table(testSat[,37],predTest) #HLM (Headlong method)
confMatrixHLM
              predTest
               red  cotton crop grey damp grey veg. stubble very damp grey
  red          445            0    8         0             5              0
  cotton crop    0          222    0         0             0              0
  grey           3            3  398        11             7              7
  damp grey      0            6   54        59             2             76
  veg. stubble   2           12    0         3           200             17
  very damp grey 0            6   25        27            25            377
```

Code 5.1. Estimating the confusion matrix.

In comparison with the confusion matrix for LDA in R-code 3.8 and QDA in R-code 3.10, we see that the above confusion matrix is better than the confusion matrix for LDA and close to the one for QDA. This is also reflected in R-code 5.2 where we calculate the sensitivity for each class and the overall accuracy.

```
> diag(prop.table(confMatrixHLM, 1))
    red soil    cotton crop    grey soil    damp grey soil  vegetation stubble
   0.9716157      1.0000000    0.9277389         0.2994924           0.8547009
very damp grey soil
          0.8195652
> accSatHLm <- sum(diag(prop.table(confMatrixHLM)))
> accSatHLm
[1] 0.8505
```

Code 5.2. Estimating the sensitivity and accuracy of the prediction.

From the above estimations we observe that the sensitivity for each class is not that different from the one when using QDA. The major difference is the sensitivity of damp grey soil which is slightly better when using HLM. We also observe that the overall accuracy when using HLM is 85,05% whereas the accuracy is 84,95% when using QDA. This indicates that classification based on the HLM is slightly better than QDA when considering the `Satellite` data. However, it should be taken into account that classification using the HLM is very time consuming since we have to use the function `cmod` to estimate the model found by the HLM.

Using (3.12) where $\widehat{\pi} = 0.8505$, we can estimate the standard error of the classification based on the HLM, i.e.:

$$\mathrm{SE}(\widehat{\pi}) = \sqrt{\frac{0.8505(1-0.8505)}{2000}} = 0.007973.$$

Hence the confidence interval is given by:

$$0.8505 \pm 2\mathrm{SE}(\widehat{\pi}) = 0.8505 \pm 0.015947.$$

The above indicates that the standard error does not change significantly relative to the standard error occurring when applying QDA.

Aside from considering the `Satellite` data we also apply the HLM to the `breastcancer` data. This data set contains 1001 variables where the last one is a binary classification variable taking the values `case` and `crontrol`. Additionally, the data are divided into a training and a test set each containing 125 observations.

We start by performing model selection on the training data using the HLM. In R-code 5.3 we apply the HLM method on the training data, where we only consider the `control` class. We observe that it only take around 38 second to go through 1000 iterations which is quite fast.

```
controlBC <- trainBC[trainBC[,1001]=="control",-1001]
initBC <- init(controlBC)

> system.time(
+   mBC <- step(initBC)
+ )
  bruger   system forløbet
   45.72     5.41    51.25
> mBC$it
[1] 1000
```

Code 5.3. Model selection on the `breastcancer` data performed by the HLM.

When we want to classify, the HLM can not be used directly on the training data since the number of variables is large. This is because we have to use the function `cmod` from the R-package `gRim` to estimate the precision matrix which involves estimating the inverse matrix. So to estimate a classifier we divide the variables in the training data into 50 random parts of 20 variables. We then perform model selection on each of the 50 parts for each class. The models found by the HLM are then fitted using the `cmod()` function which yields an estimated precision matrix J for each class in the 50 parts. Now we use the test data to estimate the conditional densities $f(\boldsymbol{x}|\texttt{case})$ and $f(\boldsymbol{x}|\texttt{control})$ given the precision matrix. Here the test data are divided into the exactly same 50 parts as the training set. Assuming that the probability of belonging to a class is is equal, we can estimate the posterior probabilities $f(\texttt{case}|\boldsymbol{x})$ and $f(\texttt{case}|\boldsymbol{x})$ using (A.1). By assuming that the 50 parts are independent the posterior probabilities for all 1000 variables are assumed to be the sum of the 50 densities. We use sum since multiplication yields very small posterior probabilities resulting in NAN values. To obtain some sort of random forest, we perform the above process 100 times and take the average. The classifier when using the HLM on the `breastcancer` data is given by the posterior probability with the highest value.

In R-code 5.4 we estimate the confusion matrix for the classifier when using the HLM.

```
> confMatrixBC <- table(testBC[,1001], predBC)
> confMatrixBC
         predBC
          case control
  case      20      10
  control   14      81
```

Code 5.4. Estimating the confusion matrix for the HLM classifier.

Recall from the description of the `breastcancer` data that `case` are patients with a p53 mutation and `control` are those without. From the above confusion matrix we see that 1/3 of patients with a p53 mutation are misclassified. This is not so good since we want the amount of misclassified patients to be small. This is also reflected in R-code 5.5 where the sensitivity and specificity are 66.6% and 85.26%, respectively.

```
> diag(prop.table(confMatrixBC, 1))
     case   control

0.6666667 0.8526316
> sum(diag(prop.table(confMatrixBC)))
[1] 0.808
```

Code 5.5. Estimating the sensitivity, specificity and accuracy of the HLM classifier.

From the above R-code we see that the accuracy of the HLM classifier is 80.8% when considering the `breastcancer`.

Since we are also interested in how good the HLM method is compared to other classifiers, we calculate the standard error of the HLM classifier using (3.12) where $\widehat{\pi} = 0.808$, that is:

$$\mathrm{SE}(\widehat{\pi}) = \sqrt{\frac{0.808(1-0.808)}{125}} = 0.0352.$$

Thus the confidence interval is given by:

$$0.808 \pm 2\mathrm{SE}(\widehat{\pi}) = 0.808 \pm 0.0705.$$

To obtain some sort of random forest the data are divided in 50 random parts 100 times. Hence the densities $f(x_i|\texttt{case})$ and $f(x_i|\texttt{control})$, for $i = 1, \ldots, N$, are both estimated 5000 times. Another version of the HLM could be to count the number of instances where $f(x_i|\texttt{case}) > f(x_i|\texttt{control})$ and save this in a variable called `votes`. The classifier is then given by the the conditional probability with the most votes. We call this version the HLMvotes.

In R-code 5.6 we estimate the confusion matrix based on the HLMvotes classifier.

```
> confMatrixBCVotes <- table(testBC[,1001], voteClass)
> confMatrixBCVotes
         voteClass
          case control
  case      25       5
  control   18      77
```

Code 5.6. Estimating the confusion matrix for the HLMvotes classifier.

IF we compare the above confusion matrix with the confusion matrix in R-code 5.4, we see that more `case`'s and fewer `control`s are predicted correctly. This indicates that the HLMvotes classifier predicts fewer patients with a mutation incorrectly than the HLM.

We see from R-code 5.7 that the sensitivity and specificity are 66.6% and 85.26%, respectively. Since the sensitivity and specificity are almost equal, test efficiency is almost independent of prevalence and equal to sensitivity or specificity.

```
> diag(prop.table(confMatrixBCVotes,1))
     case   control
0.8333333 0.8105263

> sum(diag(prop.table(confMatrixBCVotes)))
[1] 0.816
```

Code 5.7. Estimating the accuracy of the prediction given by the HLM which is based on votes.

From the above R-code we observe that the accuracy of the HLMvotes classifier is 81.6% which is a bit better than the HLM classifier. Using (3.12) where $\widehat{\pi} = 0.816$ we can estimate the standard error of the HLMvotes classifier. Hence we have that:

$$\text{SE}(\widehat{\pi}) = \sqrt{\frac{0.816(1-0.816)}{125}} = 0.03466,$$

and the confidence interval:

$$0.816 \pm 2\text{SE}(\widehat{\pi}) = 0.816 \pm 0.0693.$$

Since the number of observations is smaller than the number of variables in the `breastcancer` data, i.e. $N << p$, we can not use LDA or QDA as benchmark against the HLM. Instead we use *graphical lasso* which estimates a sparse inverse precision matrix using a lasso (L1) penalty. The graphical lasso is described in detail in section 3.3. We apply the `glasso()` function from the R-package `glasso` on the training data for each class, and thereby we obtain an estimate of the precision matrix for each of the two classes. Now using the test data we can estimate the conditional densities $f(\boldsymbol{x}|\texttt{case})$ and $f(\boldsymbol{x}|\texttt{control})$ given the precision matrices. Hence the posterior probabilities $f(\texttt{case}|\boldsymbol{x})$ and $f(\texttt{case}|\boldsymbol{x})$ are estimated by using the extended Bayes rule A.1, where we assume that the probability of belonging to one of the classes is equal. The glasso classifier is then given by the posterior probability with highest value.

The confusion matrix for the glasso classifier is estimated in R-code 5.8 where the regularization parameter in the `glasso()` function is set to 0.4.

```
> glassoconfM <- table(testBC[,1001], predglassoBC)
> glassoconfM
         predglassoBC
          case control
  case      28       2
  control   24      71
```

Code 5.8. Estimating the confusion matrix.

From the above confusion matrix we see that the glasso classifier only predicts 2 out of 30 patients with a p53 mutation incorrectly which is fewer than the HLMvotes classifier in R-code 5.6. This is good since we want the misclassification of patients with a p53 mutation to be low, but still without misclassifying too many patients without a mutation.

In R-code 5.9 we observe that the sensitivity and specificity are 93.33% and 74.74%, respectively. Since the glasso classifier has the highest sensitivity compared to the classifiers for HLM and HLMvotes, it doing the best job when we want the misclassification of patients with a mutation to be low.

```
> diag(prop.table(glassoconfM, 1))
     case   control
0.9333333 0.7473684

> sum(diag(prop.table(glassoconfM)))
[1] 0.792
```

Code 5.9. Estimating the accuracy of the prediction.

From the above R-code we also observe that the accuracy of the glasso classifier is 79.2%. This is lower than the accuracy of HLMvotes classifier which is 81.6%. Hence the HLMvotes classifier is performing best when the probability of belonging to a class is equal.

Using (3.12) where $\widehat{\pi} = 0.792$, we estimate the standard error of the glasso classifier, that is:

$$\mathrm{SE}(\widehat{\pi}) = \sqrt{\frac{0.792(1-0.792)}{125}} = 0.03578,$$

and the confidence interval:

$$0.792 \pm 2\mathrm{SE}(\widehat{\pi}) = 0.792 \pm 0.07155.$$

This indicates that the standard error of the glasso classifier is a bit higher than the standard errors of the HLM and HLMvotes classifiers.

To analyse the performance of the three binary classifiers (HLM, HLMvotes, and glasso) on the `beastcancer` data, we generate the ROC curve for each of the three models. In R-code 5.10 we use the `roc()` function from the R-package `pROC` to generate the three ROC curves. This function also estimates the AUC.

```
> rocglasso <- roc(testBC[,1001], controlglasso, levels=c("control", "case"))
> rocglasso

Call:
roc.default(response = testBC[, 1001], predictor = controlglasso,     levels =
     c("control", "case"))

Data: controlglasso in 95 controls (testBC[, 1001] control) > 30 cases (testBC
    [, 1001] case).
Area under the curve: 0.9186

> rocBC <- roc(testBC[,1001], predControlBC, levels=c("control", "case"))
> rocBC$auc
Area under the curve: 0.8923

> rocBCVotes <-roc(testBC[,1001], votesObs, levels=c("control", "case"))
> rocBCVotes$auc
Area under the curve: 0.8995
```

Code 5.10. The estimated ROC curve for the predicted classes.

We observe from the above R-code that the glasso classifier has the highest AUC.

Figure 5.1 illustrates the ROC curve for the three binary classifiers. We observe that

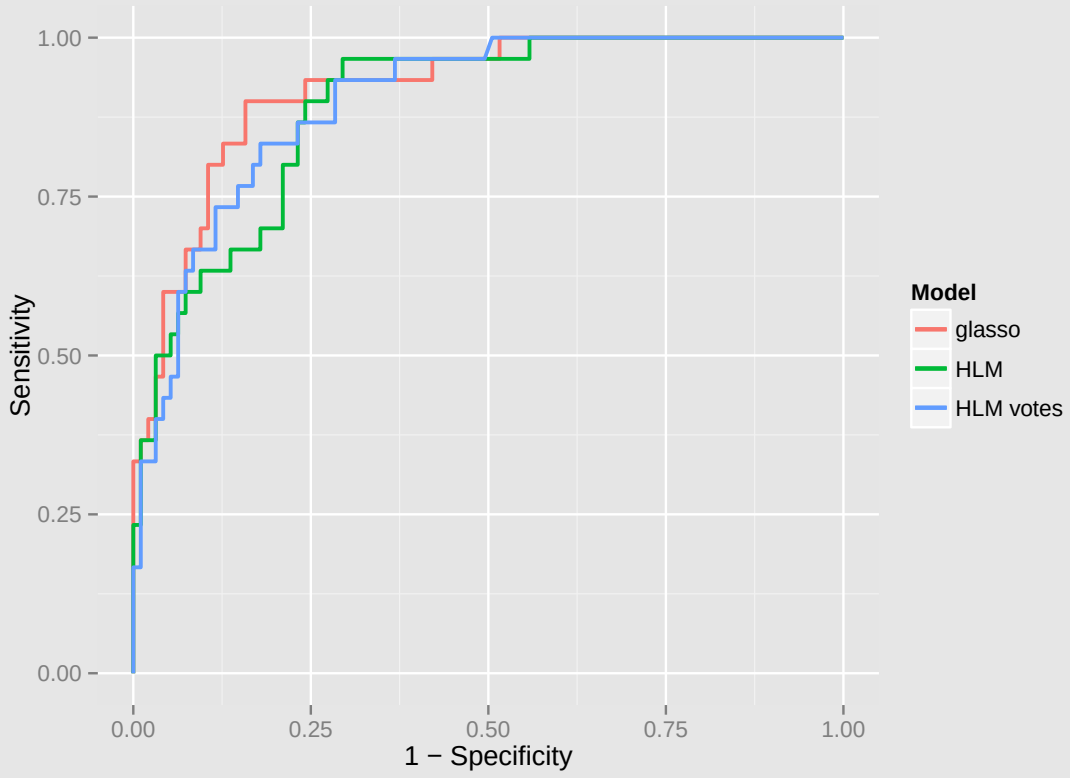

Figure 5.1. ROC curves for the three classifiers (HLM, HLMvotes, and glasso).

the ROC curve for the glasso classifier in general lies over both the HLM classifier and HLMvotes classifier which also is reflected in the AUC.

Recapitulation 6

The aim of this thesis has been to develop a classification method inspired by Random Forests, where the building blocks instead of trees are Gaussian graphical models. When doing model selection in Gaussian graphical models the most optimal procedure is based on likelihood ratio test. However, as the number of variables increases, this method becomes more and more time-consuming. Hence we suggest a model selection method which is based on a regression on the neighbors to a vertex in a graph. This method turns out to be fast and efficient compared to likelihood ratio test. We call the proposed method the headlong method (HLM) since it chooses an edge in graph a, $\mathcal{G}$, randomly.

In our approach of obtaining this method we introduce some basic statistical concepts. Here we cover important concepts such as conditional independence, graph theory, and the multivariate normal distribution which all together form the basis of a Gaussian Markov Random Field (GMRF) or Gaussian graphical model. A random vector $\boldsymbol{X} = (X_1, \ldots, X_p)^T$ following a multivariate normal distribution with mean $\boldsymbol{\mu}$ and precision matrix $\Sigma^{-1} = J$, is a GMRF with regard to an undirected graph $\mathcal{G} = (V, E)$ if it obeys the pairwise Markov property and:

$$\{i, j\} \notin E \quad \Longrightarrow \quad J_{ij} = 0 \quad \text{for all} \quad i \neq j.$$

This sparsity of the precision matrix provides a unified way of presenting the conditional independencies though a GMRF. It turns out the local and pair-wise Markov properties are equivalent in a GMRF which is essential when performing regression on the neighbors.

Since we consider a regression on the neighbors, we introduce general linear models, which may be written as $\boldsymbol{Y} = D\boldsymbol{\beta} + \boldsymbol{\varepsilon}$. Here D is a design matrix , $\boldsymbol{\beta}$ the parameters we want to estimate and $\boldsymbol{\varepsilon} \sim \mathrm{N}_N(0, \sigma^2 I)$. Additionally, we introduce the t-test to test the individual parameters for significance.

To benchmark the suggested HLM we introduce Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and graphical lasso. If the number of observations are larger than the number of variables, i.e. $N >> p$, in a statistical sample, we use LDA and QDA. Graphical lasso, on the other hand, is used when

$N << p$. In such a sample LDA and QDA might fail since the covariance matrix is singular and cannot be inverted.

The confusion matrix is introduced to compare the different methods, where we compute the sensitivity and specificity. The sensitivity and specificity are used to introduce Receiver Operating Characteristic (ROC) curves which visualize the performance of a binary classifier and estimate the area under curve (AUC). The AUC is used to compare the performance of different binary classifiers where high AUC is to prefer. Through the confusion matrix we also estimate the accuracy which is the proportion of correctly classified observations.

we benchmark the HLM based on the data sets `Satellite` and `breastcancer`. The `Satellite` data consist of 6435 observation, 36 variables and a classification variable of six classes. In addition The `breastcancer` data consist of 250 observation, 1000 variables and a binary classification variable. To perform classification both data set are decide into a training set and a test set.

First we perform classification on the `Sattellite` data using the HLM, LDA and QDA and compare the performance of the three methods. Table 6.1 shows the total accuracy ($\widehat{\pi}$), the standard deviation (SE) and confidence interval (CI) for the three classification methods. Table 6.1 shows that QDA and HLM are significantly better

Methods	$\widehat{\pi}$	SE($\widehat{\pi}$)	CI
LDA	0.826	0.008477	0.80905;0.84295
QDA	0.8495	0.0079953	0.8335;0.86549
HLM	0.8505	0.007973	0.83455;0.86645

Table 6.1. Estimates for the `Satellite` data.

that LDA since the accuracy of both QDA and HLM lay outside the confidence interval of LDA. We also observe that the accuracy of HLM is a slightly better than the accuracy of QDA. But QDA is still to prefer, since it is faster than the HLM. This is because the HLM only finds a suitable graph, which goes very fast, and therefore we need to estimate a model using the function `cmod()`. In general LDA and QDA are to prefer when $N >> p$, since they are operationally faster than the HLM, when we want to classify a sample of observations. If we on the other hand only are doing model selection, the HLM is very useful since it is very fast.

Considering the `breastcancer` the HLM is still very fast when perform model selection. One the other hand when we want to classify the HLM cannot be applied directly on data since the number of variables is large. This is because we have to use the `cmod()` function to estimate the model found by the HLM. Instead we divide the 1000 variables into 50 random parts of 20 variables, and use the HLM to find a suitable model on each of the 50 parts. We then estimate the models using the `cmod()` function and estimate the conditional densities $f(x_i|\text{case})$ and $f(x_i|\text{control})$. Assuming the 50 parts are independent, we sum the densities to obtain the density

of all 1000 variables. To obtain some sort of random forest we perform this process 100 times and take the average.

When data is divided in 50 random parts 100 times as above, the densities $f(x_i|\text{case})$ and $f(x_i|\text{control})$, for $i = 1, \ldots, N$, are both estimated 5000 times. An alternative to the HLM is to count the number of instances where $f(x_i|\text{case}) > f(x_i|\text{control})$, and the one with the highest number is our class. We call this method HLMvotes.

We compare the HLM and HLMvotes with graphical lasso which is relatively fast when considering a data set with 1000 variables. Table 6.2 shows the result of the classification when applying glasso, HLM and HLMvotes to the `breastcancer` data. Here we observe that all three methods lie within the confidence intervals of each

Methods	$\widehat{\pi}$	SE($\widehat{\pi}$)	CI	AUC
glasso	0.792	0.03578	0.72845;0.87155	0.9186
HLM	0.808	0.03523	0.73754;0.87846	0.8923
HLMvotes	0.816	0.03466	0.74668;0.88532	0.8995

Table 6.2. Estimates for the `breastcancer` data.

other. This indicates that none of the methods are significantly better than the others. We also notice that the HLMvotes has the highest accuracy, and graphical lasso the lowest accuracy which indicates that HLMvots perform better than lasso. But considering at the AUC glasso is better.

Hence we can deduce that the HLM is always to prefer when doing model selection, regardless of the data set, since it is very fast. In terms of classification the HLM method can be used as an alternative, when the number of variable is very large by dividing the variables into smaller parts. We observe that the HLM is slightly better than the other methods on the data sets that we consider.

If we look at what, we cloud have done to improve our classification. For example we cloud have used leave one out cross validation which mean we use the whole data set instead of a training and test set. This generally gives a lower standard error.

Bibliography

Gareth James, Trevor Hastie, R. T. [2013], *An Introduction to Statistical Learning: with Applications in R*, number ISBN: 978-1-4614-7137-0 *in* 'Handbook', Springer New York.

J. Friedman, T. Hastie, R. T. [2009], *The Elements of Statistical Learning*, number ISBN: 978-0-387-84858-7 *in* 'Handbook', Springer-Verlag New York.

Jerome Friedman, Trevor Hastie, R. T. [2007], 'Sparse inverse covariance estimation with the graphical lasso.', *Oxford Journals.* **9**.
URL: *http://biostatistics.oxfordjournals.org/content/9/3/432.short*

Koller, D. and Friedman, N. [2009], *Probabilistic Graphical Models: Principles and Techniques*, number ISBN: 978-0-262-01319-2 *in* 'Handbook', Massachusetts Institute of Technology.

Lance D. Miller, Johanna Smeds, J. G. [2005], 'An expression signature for p53 status in human breast cancer predicts mutation status, transcriptional effects, and patient survival.', *The National Academy of Sciences.* **102**.
URL: *http://dx.doi.org/10.1073/pnas.0506230102*

Lauritzen, S. L. [1996], *Graphical Models*, number ISBN: 978-0-19-852219-5 *in* 'Handbook', OXFORD University Press.

Leisch, F. and Dimitriadou, E. [2010], 'mlbench: Machine learning benchmark problems', CRAN.
URL: *http://cran.r-project.org/web/packages/mlbench/mlbench.pdf*

Olofsson, P. [2005], *Probability, Statistics and Stochastic Processes*, number ISBN: 978-0-471-67969-1 *in* 'Handbook', John Wiley & Sons inc.

Petersen, K. B. and Pedersen, M. S. [2012], 'The matrix cookbook'. Version 20121115.
URL: *http://www2.imm.dtu.dk/pubdb/p.php?3274*

Poirier, D. J. [1995], *Intermediate Statistics and Econometrics : A Comparative*, number ISBN: 978-0-262161497 *in* 'Handbook', MIT Press.

Rahul Mazumder, T. H. [2012], 'The graphical lasso: New insights and alternatives', *Electronic Journal of Statistics* **6**.
URL: *http://web.stanford.edu/ hastie/Papers/glassoinsights.pdf*

Rue, H. and Held, L. [2005], *Gaussian Markov Random Fields: Theory and Applications*, Vol. 104 of *Monographs on Statistics and Applied Probability*, Chapman & Hall, London.

Spence, L., Insel, A. and Friedberg, S. [2008], *Linear Algrebra - A Matrix Approach*, number ISBN: 978-0-13-158034-3 *in* 'Handbook', Pearson Education, Inc.

Thyregod, P. and Madsen, H. [2010], *Introduction to general and generalized linear models*, number ISBN: 978-1-4200-9155-7 *in* 'Handbook', CRC Press Incorporated.

Wikipedia [2015], 'receiver operating characteristic — wikipedia, the free encyclopedia'.
URL: *http://en.wikipedia.org/wiki/Receiver_operating_characteristic*

Appendix A

In this appendix are assembled some of the more general terms and calculations that have been omitted from the report.

A.1 Generalized inverse

The following is written with [Spence et al., 2008, Section 6.7] as source of inspiration.

Definition A.1
The *generalized inverse* of a matrix $A \in \mathbb{R}^{n\times m}$ is a matrix $A^g \in \mathbb{R}^{m\times n}$ which satisfies the condition $AA^gA = A$.

Notice that the usual matrix inverse is covered by this definition because $AA^{-1}A = A$. The term "generalized" inverse is used for a general rectangular matrix and to distinguish from inverse matrix that is for a square matrix. Generalized inverse is also called *pseudo inverse*.

A.2 Bayes Theorem

This section is based on the source [Olofsson, 2005, Section 1.6.1].

For events A and B, provided that $P(B) \neq 0$, *Bayes theorem* is formulated as:

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}.$$

Bayes theorem for continues random variables X and Y is given by:

$$f(x|y) = \frac{f(x,y)}{f(y)} = \frac{f(y|x)f(x)}{f(y)},$$

where $f(y) > 0$. Here $f(y)$ is obtained by integrating X out in the numerator.

For some partition $\{A_j\}$ of the event space the law of total probability state that:

$$P(B) = \sum_j P(B|A_j)P(A_j).$$

Bayes theorem may therefor be more generally written as:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)} \tag{A.1}$$

A.3 Transformation

In the following transformation is given in its general form and it is inspired by [Poirier, 1995, Section 4.4].

Let $\boldsymbol{X} \in \mathbb{R}^N$ be a continuous random vector with density $f(\boldsymbol{x})$. Let $g : \mathbb{R}^N \to \mathbb{R}^N$ be a one-to-one differentiable function and let $\boldsymbol{Y} = g(\boldsymbol{X})$. Then Y has density:

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \Big|J\Big| f_{\boldsymbol{X}}(g^{-1}(\boldsymbol{y})) \tag{A.2}$$

where $|J| \coloneqq |\det(J)| \neq 0$ and J is the Jacobian matrix:

$$J = \begin{pmatrix} \frac{\partial}{\partial y_1} g^{-1}(y_1) & \cdots & \frac{\partial}{\partial y_N} g^{-1}(y_1) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial y_1} g^{-1}(y_N) & \cdots & \frac{\partial}{\partial y_N} g^{-1}(y_N) \end{pmatrix}.$$

Now consider the following affine transformation:

$$\boldsymbol{Y} = A\boldsymbol{X} + \boldsymbol{b}$$

where $\boldsymbol{X} \in \mathbb{R}^N$ is a continuous random vector and A is an invertible matrix. By isolating $\boldsymbol{X}$ we obtain:

$$\boldsymbol{X} = A^{-1}(\boldsymbol{Y} - \boldsymbol{b}),$$

and due to (A.2) $\boldsymbol{Y}$ then has density:

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = |A^{-1}| f_{\boldsymbol{X}}(A^{-1}(\boldsymbol{y} - \boldsymbol{b})).$$

If $\boldsymbol{X} \sim \mathrm{N}_N(\boldsymbol{\mu}, \Sigma)$ this density is given as:

$$\begin{aligned} f(\boldsymbol{y}) &= \frac{1}{|A|} \frac{1}{(2\pi)^{N/2}|\Sigma|^{N/2}} \exp\Big(-\frac{1}{2}\Big(A^{-1}(\boldsymbol{y} - \boldsymbol{b}) - \boldsymbol{\mu}\Big)^T \Sigma^{-1} \Big(A^{-1}(\boldsymbol{y} - \boldsymbol{b}) - \boldsymbol{\mu}\Big)\Big) \\ &= \frac{1}{|A^T A|^{1/2}} \frac{1}{(2\pi)^{N/2}|\Sigma|^{N/2}} \exp\Big(-\frac{1}{2}\Big(\boldsymbol{y} - \boldsymbol{A}\mu - \boldsymbol{b}\Big)^T \Big(A^{-1}\Big)^T \Sigma^{-1} A^{-1} \Big(\boldsymbol{y} - A\boldsymbol{\mu} - \boldsymbol{b}\Big)\Big) \\ &= \frac{1}{(2\pi)^{N/2}|A^T \Sigma A|^{N/2}} \exp\Big(-\frac{1}{2}\Big(\boldsymbol{y} - (\boldsymbol{A}\mu + \boldsymbol{b})\Big)^T \Big(A^T \Sigma A\Big)^{-1} \Big(\boldsymbol{y} - (A\boldsymbol{\mu} + \boldsymbol{b})\Big)\Big) \end{aligned}$$

which shows that $\boldsymbol{Y}$ has the following distribution:

$$\boldsymbol{Y} \sim \mathrm{N}_N\Big(A\boldsymbol{\mu} + \boldsymbol{b}, A^T \Sigma A\Big).$$

Hence linear transformations preserve multivariate normality.

When A is a $k \times N$ matrix we have the following lemma.

Lemma A.2
Assume $\boldsymbol{Y} \sim \mathrm{N}_N(\cdot, \cdot)$, and A is a $k \times N$ matrix. Then:

$$A\boldsymbol{Y} \sim \mathrm{N}_k(\cdot, \cdot)$$

have the following mean and covariance matrix:

$$\mathrm{E}\left[A\boldsymbol{Y}\right] = A\mathrm{E}\left[\boldsymbol{Y}\right], \quad \mathrm{Var}\left[A\boldsymbol{Y}\right] = A\mathrm{Var}\left[\boldsymbol{Y}\right]A^T$$

Proof. Since the mean $\mathrm{E}\left[\cdot\right]$ is linear, it follows that $\mathrm{E}\left[A\boldsymbol{Y}\right] = A\mathrm{E}\left[\boldsymbol{Y}\right]$.

For the covariance matrix we have:

$$\begin{aligned}
\mathrm{Var}\left[A\boldsymbol{Y}\right] &= \mathrm{E}\left[(A\boldsymbol{Y} - A\mathrm{E}\left[\boldsymbol{Y}\right])(A\boldsymbol{Y} - A\mathrm{E}\left[\boldsymbol{Y}\right])^T\right] \\
&= \mathrm{E}\left[A(\boldsymbol{Y} - \mathrm{E}\left[\boldsymbol{Y}\right])(\boldsymbol{Y} - \mathrm{E}\left[\boldsymbol{Y}\right])^T A^T\right] \\
&= A\mathrm{E}\left[(\boldsymbol{Y} - \mathrm{E}\left[\boldsymbol{Y}\right])(\boldsymbol{Y} - \mathrm{E}\left[\boldsymbol{Y}\right])^T\right] A^T \\
&= A\mathrm{Var}\left[\boldsymbol{Y}\right]A^T,
\end{aligned}$$

thus $\mathrm{Var}\left[A\boldsymbol{Y}\right]$ is a $k \times k$-matrix. □

List of Corrections