Statistical evaluation of features in classification problems with applications to detection of hypoglycemic conditions based on EEG data

Laura Friis Frølich

Kongens Lyngby 2011
IMM-Master's Thesis-2011-60

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-Master's Thesis: ISSN

Independent components (ICs) are patterns in EEG data that are assumed to correspond to neural generators of electricity in the brain. However, some ICs represent artifacts such as eye movements. By denoising EEG data through removal of automatically classified artifactual ICs, we investigated the effect of artifactual signals on detection of seizures due to to falling blood sugar levels in type I diabetics. The classification methods binary LDA, multiclass LDA, binary QDA, multiclass QDA, SVM, logistic regression, logistic regression with forward selection, L1-regularized logistic regression, multinomial regression, multinomial regression with forward selection, decision trees, ADJUST [41] and an algorithm proposed by the Berlin BCI group [56] were compared to find the best automatic classifier of ICs. Variance of feature estimates over cross-validation folds and effects of features on classification performance were investigated.

Seizure detection performance was decreased when the model was trained on data without artifacts. This decrease in performance may either be because (1) the detection model relies on artifacts such as muscle twitches and eye movements during seizures, or (2) because neural activity was wrongly removed in the cleaning process. In the first case, seizure detection models that do not rely on artifacts must be found. In the second case, models trained on data cleaned by better noise removal methods will most likely increase seizure detection performance based on EEG recorded by a device implanted in the brain, since such a device cannot detect artifacts.

L1-regularized logistic regression and logistic regression with forward selection turned out to be the best methods. Almost all features were chosen in L1-regularized logistic regression as well as by a criterion based on mutual information between features and class assignments. This indicates that all features represent class relevant information. Small variances of feature coefficient estimates were seen, implying that the estimated models represent structures in data, and not chance relations.

Preface

This thesis was prepared at DTU Informatics, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the degree Master of Science in Engineering.

The thesis deals with the effects of non-neural signals in EEG on the detection of hypoglycemic seizures. Automatic identification of artifactual parts of the EEG signal is described in great detail, and used to denoise raw EEG data to derive the effect of artifacts on seizure detection. Classification methods are described, and effects of features on classification performance investigated.

Lyngby, August 2011

Laura Friis Frølich

would like to thank a number of people, without whom this project would not have been possible. My supervisors Tobias Andersen and Morten Mørup have provided immense help in the forms of constant support, guidance, help with technical details and overall involvement and enthusiasm for the project.

Additionally, I would like to thank Scott Makeig for hosting me at the Swartz Center for Computational Neuroscience (SCCN). It was during this stay, through conversations with Scott Makeig and Christian Kothe, that the problem of independent component classification, which was a large part of this project, came

to my attention. Scott Makeig and Christian Kothe were very inspirational and helpful in preliminary investigations of the problem, which I carried out during the stay at SCCN. My thanks are also due Klaus Gramman and Julie Onton for letting me use their data for investigation of the problem of independent component classification.

I am also grateful to the people at the company HypoSafe, who put their data at my disposal and answered all my questions rapidly and comprehensively.

My partner and beloved friend Morten Mølgaard has been a constant source of support, and remained patient even when I have been at my most impossible. In addition to emotional support, Morten is always ready to discuss technical details and do proof reading.

I would like to thank my brother Emil for proof reading the report.

My parents and in-laws have continually been very supportive and understanding, giving me space and time to work.

Furthermore, I thank my friends who have been there when I needed a day off, and understood when I could not take a day off.

I also gratefully acknowledge the funds that provided financial support during my stay at SCCN. These funds are, in no particular order, Otto Mønsteds Fond, Observator mag.scient. Julie Marie Vinter Hansens Rejselegat, Knud Højgaards Fond, Reinholdt W. Jorck og Hustrus Fond, and William Demant og Hustru Ida Emilies fond (Oticon fonden).

Laura Friis Frølich
Gudmindrup, August 2011

Contents

Abbreviations and notation

This chapter provides an overview of notation and abbreviations for easy reference.

Notation

The notation in this report will adhere to the following conventions.

Lowercase letters from the beginning of the alphabet denote constants, and lowercase letters from the end of the alphabet denote variables. A constant scalar can then be denoted by a and a scalar variable by x.

Bold symbols denote column vectors. Thus $\boldsymbol{x}$ is a column vector variable, and $\boldsymbol{a}$ is a constant column vector. Row vectors are written as the transpose of a column vector, such that $\boldsymbol{a}^T$ is a constant row vector.

Matrices are written as capital letters. Thus X is a variable matrix and A a matrix of constants.

Capital letters from the end of the alphabet are also used to denote random variables. When used to denote random variables, we use the font $\mathcal{Z}$ instead of the standard Z.

Elements of vectors and matrices are identified by indices. The i^{th} coordinate of a variable vector $\boldsymbol{x}$ is denoted by $\boldsymbol{x}_i$. The element in the i^{th} row and j^{th} column

of a matrix X is identified by $X_{i,j}$. The i^{th} row of the matrix X is denoted by $X_{i,:}$ and the j^{th} column by $X_{:,j}$.

Sometimes, we will need to refer to different vectors of the same type, but from different populations. Means of populations are a good example of such a case. We then use the same symbol to emphasize that the vectors represent the same quantity. In the case of the mean, we would choose $\boldsymbol{\mu}$. We use subscripts $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ to distinguish between the mean vectors from different populations. The i^{th} coordinate of the mean vector from class k will then be referred to as $\boldsymbol{\mu}_{k,i}$.

The indicator function will be denoted by $\mathcal{I}$. The indicator function of a boolean statement b is written $\mathcal{I}(b)$ and is equal to 1 if b is true, and 0 otherwise.

The $r \times r$ identity matrix is denoted by $I_{r\times r}$, I_r, or just I if the dimensions are clear. A vector of zeros with r elements is denoted by $\mathbf{0}_r$.

Special symbols

Some letters and symbols retain the same meaning throughout the report.

We use K to denote the number of classes in a classification problem.

Vectors of coefficient estimates are denoted by $\boldsymbol{\beta}$, with $\boldsymbol{\beta}_0$ referring to the intercept.

The number of observations is denoted by n. We use m to denote the number of channels in EEG recordings. We also use m to denote the number of variables in data containing a number of variables and a response.

We discuss two types of data throughout the report. EEG recordings constitute the first type of data, while the second type consists of observations with explanatory variables and a response. We denote data matrices containing EEG recordings by Y. Data matrices containing explanatory variables and a response are denoted by X.

Data from EEG recordings consist of observations of each channel over time. Each row contains the values from one channel. Thus the number of rows of Y is denoted by m and the number of columns by n. Conversely, X contains observations in the rows, while each column corresponds to an explanatory variable. The number of rows of X is then denoted by n and the number of columns by m.

Abbreviations

Abbreviations and uncommon words used in this report are shown in table 1.

BCI	Brain-computer interface
BBCI	Berlin BCI group
CSP	Common spatial pattern
EEG	Electroencephalogram
Hypoglycemia	Blood sugar levels under normal
IC	Independent component
ICA	Independent component analysis
logreg	Logistic regression
MNR	Multinomial regression
PC	Principal component
PCA	Principal component analysis
QDA	Quadratic discriminant analysis
SCCN	Swartz Center for Computational Neuroscience
SVM	Support vector machine

Table 1: Abbreviations

Part I

Introduction

Automatic prediction of lower-than-normal blood sugar levels (hypoglycemia) in type I diabetics based solely on their electroencephalographam (EEG) is investigated in this project. We infer the effect of artifacts in the EEG on the performance of predictions. To derive the effect of artifacts, we compare predictions of hypoglycemia based on raw data to predictions based on cleaned data without artifacts. A large part of the project concerns the identification of artifacts since only few automatic tools for this purpose have so far been proposed, and these are not good enough to substitute for human expert classifications.

The EEG is the distribution of electrical activity on the scalp which Hans Berger was the first to study in the 1920's [32]. Since the brain elicits electrical activity when active [42, p. 27], and some of this electrical activity can be detected on the scalp, recordings of the EEG can be used to study how the brain works. Changes in the EEG are then interpreted as changes in activity within the brain.

Studying the EEG is a non-invasive way to probe the inner workings of the mind. The instruments for recording EEG are easily portable, which makes EEG recordings an ideal choice for diagnostic and research purposes. In addition, EEG recordings are cheap compared to other techniques of observing brain activity such as fMRI and PET. Another advantage of EEG is the high temporal resolution. Unfortunately, the spatial resolution of EEG is low.

The EEG is recorded by fitting a cap designed for recording EEG on a person's head. Such a cap has holes in which electrodes are placed for recording. An image of a person with an EEG cap is shown in figure 1.

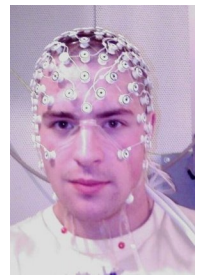

Figure 1: A person with an EEG cap. Downloaded from http://en.wikipedia.org/wiki/File:EEG_cap.jpg. The image has been released into the public domain by the creator of the image.

Since EEG is solely measured on the scalp, the underlying brain signals cannot be directly observed. Instead, locations of the electricity generating sources which are assumed to correspond to the active areas in the brain must be estimated. A common method exploits the approximately linear conduction of electricity within brain tissue. The approximately linear conduction implies that the signal measured at each electrode is approximately a linear combination of electricity generated by sources within the brain. Under the assumption that areas within the brain generate activity independently of each other at each instant of time, independent component analysis (ICA) [27, 27, 26, 13] identifies the activity of each generating source over time, as well as the scalp map of electricity due to each source. We can then estimate the location of each source within the brain by finding the positions of minuscule electrical generators that would give rise to the observed pattern of electricity on the scalp [33].

The patterns found by ICA are usually referred to as independent components (ICs). Typically, some of the ICs found by ICA do not represent neural activity. We will refer to such ICs as artifacts. Causes of artifacts include the pulse, the electrical grid, eye movements, and muscle twitches where the electrodes are attached. By removing artifacts from the recorded EEG, it becomes easier to find patterns in the neural activity. We illustrate this type of denoising of EEG recordings in figure 2. The effect of removing artifactual ICs is shown in figure 3. The left column of figure 3 shows raw data, while the right column shows the same data with artifactual components removed. The top left plot shows simultaneous activity increases at 5464 and 5465 in both the electrodes above the eyes (Fp1 and Fp2). The simultaneous activity at both eye channels is highly indicative of an eye-related artifact, which has been successfully removed in the top right plot.

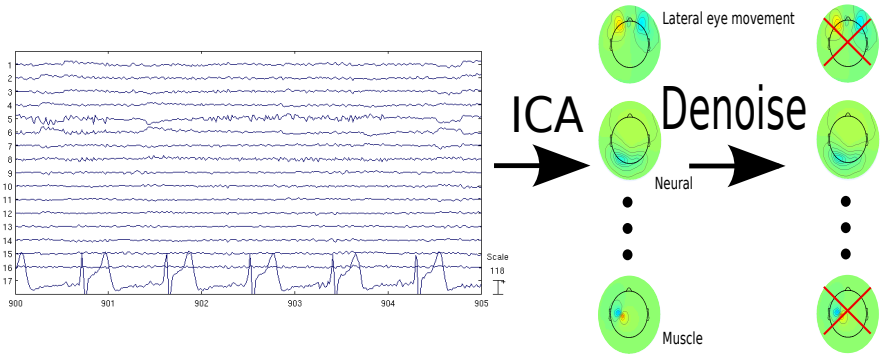

Figure 2: Removing artifactual activity from recordings of EEG.

Changes in the EEG preceding states of hypoglycemia were reported in 1988 [47]. Some such changes are referred to as seizures. Detecting seizures will then allow us to predict states of hypoglycemia. EEG recorded during normal blood sugar levels and during a seizure are shown in figure 3. The top row shows normal EEG, while the bottom row shows EEG from a seizure. The left column holds raw data, and the right column holds data that was cleaned by identifying and removing artifactual ICs. In the raw data, the EEG recorded during the seizure is clearly more erratic than that from the period of normal blood sugar levels. The difference persists, albeit less pronounced, in the cleaned data. The atypical EEG during seizures has been proven possible to detect previously [31]. The company HypoSafe aims to put the phenomenon of unusual EEG characteristics as blood sugar levels decrease into practical use through an implantable device that will record EEG continuously. This device will then warn of impending hypoglycemia in time for corrective measures to be taken. In the remainder of

this report, we will refer to detection of seizures interchangeably with prediction of hypoglycemia since the two are equivalent.

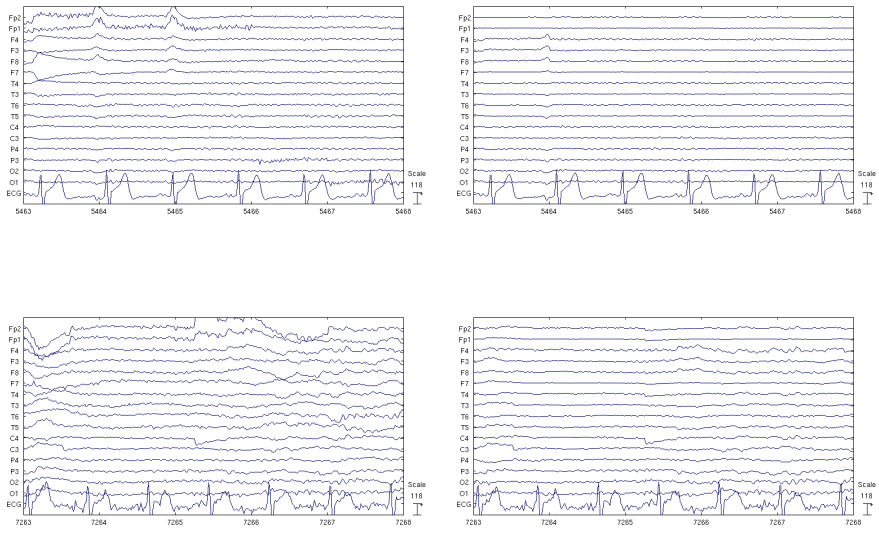

Figure 3: Top row: EEG during normal blood sugar levels. Bottom row: EEG recorded during a seizure. Left column: EEG that has not been denoised. Right column: denoised EEG. In the denoised normal EEG, the eye artifacts at 5464 and 5465 have been removed successfully. Differences between normal and seizure EEG are still present in the denoised data, although less pronounced than in raw data.

Warning diabetics of decreasing blood sugar levels at an early stage is important because the ill effects of hypoglycemia are severe. If blood sugar levels become too low, a coma may even be induced. Some diabetics have difficulty noticing when blood sugar levels start to fall, causing them to maintain constantly elevated blood sugar levels to avoid hypoglycemia. Unfortunately, levels of blood sugar that are too high over a longer period of time have long-term side effects such as failing eye sight and kidneys [59, 29, 11, 2, 19, 49]. For these reasons, it is important to maintain a stable level of blood sugar. The HypoSafe device will help accomplish this.

We will investigate whether artifacts in EEG recordings may explain part of the good performance in hypoglycemia prediction described in [31]. If this is the case, more work must be done to automatically detect solely neural changes. If it is not the case that seizure detection relies on patterns of artifacts, it should

be possible to improve seizure detection by learning from EEG data without artifacts. In order to remove artifacts, we need an automated method that distinguishes neural from artifactual ICs. Such a tool with reliable performance does not exist. Since we need a tool that automatically classifies ICs, we also investigate classification of ICs.

Chapter 1

State of the art

We now give an overview of existing methods for automatic classification of scalp patterns found by ICA and describe methods for automatic prediction of hypoglycemia in type I diabetics.

1.1 Artifact removal

Recently, a review of methods for artifact removal from EEG has been given [18]. Independent component (IC) classification is employed by some of these approaches, either as a step in a more elaborate scheme [15, 44], or as the sole artifact identification method. We treat the latter denoising approach which solely relies on automatic classification of ICs.

To gain an understanding of the current standing of research for automatic classification of ICs, google and google scholar searches with the search terms "eeg automatic artifact component", "eeg automatic component classification", "eeg classification of independent component overview", "eeg independent component classification", "ica eeg", and "ica eeg review" were undertaken. We did not find any studies comparing the different approaches on the same data set. Nor were any reviews of the current state of automatic component classification found.

Thus we give a brief overview based on individual papers, primarily of recent publication.

Most independent component classification approaches focus on the binary classification problem of differentiating between artifactual and neural components [56, 17, 15, 12, 52, 21, 64]. Some of the suggested features for this binary classification are characteristics of the power vs. frequency curve, complexity of the dipole fit, range, kurtosis and skewness of the time series of components, and peak amplitude relative to variance. Correct classification of about 90% has been reported [56], corresponding to the agreement between human experts [34].

Some approaches only attempt detection of one type of artifact, usually eye movements [12, 21, 30]. The identification of eye-related artifacts seems to be a relatively easy problem judging from these papers. Some features suggested for detection of eye-movement ICs are the power spectral density (linearly related to the square of the band-power derived from the spectrogram), variance and kurtosis in the temporal domain, and correlation with designated eye reference channels or channels close to eye areas.

Others, however take several types of artifacts into account [61, 4, 30, 41]. Features suggested for this classification problem include the distance between the distributions of time series of components and a reference heart-beat signal, measures of spatial characteristics of the IC activity on the scalp, and averages of temporal activation of components if the signal is split into small epochs.

Only two papers with the same goal as this project, namely that of automatically classifying ICs, were found.

The Berlin BCI group [56] proposed an algorithm for binary classification of ICs into artifactual or neural components. We will refer to this method as the BBCI (Berlin BCI) method in the remainder of this report. The method proposed in [56] uses support vector machines with a number of features specifically chosen to reflect differences between neural and artifactual ICs. A problem with this method is that the components that are mixes of artifactual and neural components are identified as artifactual. If all ICs that are classified as artifacts are removed, some neural activity is also lost. The algorithm has been further developed and described recently [64]. In this new approach, dimension reduction of data is performed before denoising. The dimension reduction will most likely decrease the risk of removing mixed ICs since mixed ICs will probably not be present in the dimension-reduced data. Due to the very recent publication of [64], which is still in its preliminary version as of 9/8/2011, we have not been able to incorporate the proposed method in our comparisons.

The algorithm ADJUST, described in [41], aims to detect different kinds of arti-

facts, but does not identify neural components. ADJUST has been implemented as a plug-in to EEGLab [14]. ADJUST works by finding thresholds of different features that distinguish the different types of artifacts. These thresholds are found for each new data set by applying Gaussian Mixed Models to the distributions of these values among all ICs for the data set. A problem with this method is that no class for neural components exists. Thus it is not possible to only retain ICs that solely represent neural activity. Also, it is not possible for the end-user to train the classifier on additional classes. Hence components can only be classified into the classes that ADJUST was built to handle.

1.2 Seizure detection

It has long been known that hypoglycemia induces changes in the activity in the EEG [42, p. 395] and [47]. With the introduction of the HypoSafe device [5], this knowledge has been put to use. The authors of [31] use a two-layer neural network with the hyperbolic tangency function to determine one-second intervals consistent with impending hypoglycemia. A threshold for the number of such intervals in a period of time must then be exceeded before impending hypoglycemia is set to be detected. Features used are the power in the α-, β-, γ-, and θ-bands.

Serious attempts at automatic detection of hypoglycemia warning signs in the EEG seem to lack in literature previous to [31]. Common spatial patterns and variants thereof have won wide-spread usage in many EEG classification schemes [8]. Since this method performs well in many tasks, we chose to use this to detect seizures.

Chapter 2

Pipeline to learn seizure prediction model

An outline of the process to construct a seizure detection model is shown in figure 2.1.

As shown in figure 2.1, we allow for input data with corrupted values. Corrupted values often occur in EEG recordings from clinical settings since subjects that come to clinics for diagnosis are not used to having their EEG recorded. Also,

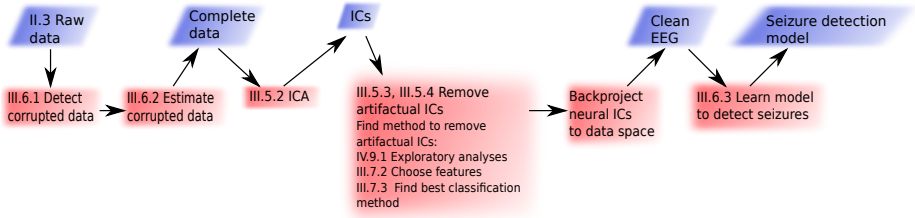

Figure 2.1: Flowchart showing the route from raw data, possibly containing missing values, to the final seizure prediction model. Red rectangles represent actions and blue parallelograms represent output and input.

short sessions are desirable in order to save time for both the subject and the EEG technician. This increases the risk that electrodes fall off at times, leading to corrupted values in the EEG recording. The first step in the process must then be to detect and reconstruct corrupted values. This is explained in more detail in section 6.2.

The clinical setting also increases the risk that artifacts such as muscle twitches and eye movements occur since the subject is inexperienced. To obtain a model that does not depend on such random noise, we must clean the data of artifacts. To accomplish this, we decompose the data into ICs. Once this is done, we can identify artifactual ICs and remove these. By back-projecting the resulting matrix of independent components to the original data space of channels$\times$time, we obtain a data matrix without signals from artifacts. We then pass this cleaned data along with the manually labeled intervals of seizure activity to a common spatial patterns (CSP) classifier to learn a model.

To test whether denoising is beneficial, we also perform the process shown in figure 2.1, but skipping the steps of ICA and removal of non-neural ICs. We then test the two models on raw data, resembling the online setting. Thus we will learn whether denoising through ICA improves upon the ability to automatically predict hypoglycemia.

By also testing the model from denoised training data on test data that was also denoised by IC classification, we will learn whether seizure detection is really a result of neural activity. If the case with denoised training and test data performs worst, it is a sign that successful seizure detection relies on a different pattern of artifacts before hypoglycemia from that during normal levels of blood sugar.

2.1 Aims

The aims of this project are three-fold. Firstly, we wish to investigate the effect of artifacts in EEG in relation to prediction of hypoglycemia. We have two issues related to this aim. One, do artifacts impede seizure detection learning? Two, does seizure detection partially rely on patterns of artifacts?

In order to investigate effects of artifacts, we need an automatic classifier of ICs with reliable performance, which does not yet exist. The second aim is then to construct such a classifier.

The third aim consists of assessing feature importance in IC classification. By

investigating which features relate best to the classes of ICs, we obtain a better understanding of the structure of the classification problem. Additionally, we may be able to improve classification by only choosing the features that best relate to classes.

We will also investigate aspects of component classification such as whether classes can be linearly separated, and whether it is an advantage to include a class of ICs that experts do not classify. These ICs without manual labels represent mixes of artifactual and neural activity.

2.2 Structure

The structure of the report is as follows.

In part II, we describe the two data sets analyzed in this project.

We then move on to describe the theory used throughout the report in part III. We also describe the features used for classification of ICs in this part.

Next, results are given in part IV. All results regarding classification of ICs are described in chapter 7. In this chapter, we describe the exploratory analyses in section 7.1. The results from comparing classification methods are described in section 7.2, while analyses of features are described in section 7.4.Results relating to seizure detection are given in chapter 8. We evaluate the data reconstruction method in section 8.1. The performance of CSP trained on the raw data and the performance of CSP when trained on cleaned data are compared in section 8.2.

Finally, we draw conclusions in chapter 9.

To maintain a natural flow in the presentation, we have chosen to give the mathematical details in appendices. These details, however, do constitute necessary work for the project as a whole to succeed, and are thus an integrated part of the project. The relevant appendices will be referred to from the related sections.

Part II

Data

Chapter 3

Seizure detection

Data for investigating seizure detection was supplied the by company HypoSafe.

This data set consists of measurements from 16 EEG channels, the electrocardiogram (ECG), and, for some time points comments from a neurologist defining whether or not the brain activity at that time is normal or shows characteristics of a seizure. Recordings from six type I diabetes patients are available. For two of the patients, measurements were taken twice, with about 14 months between measurements. The eight sessions lasted between 77 and 300 minutes (127±72, mean±sd).

Classifications of brain activity from the neurologist were recorded at intervals of approximately five minutes throughout the session during non-seizure activity. During seizures, approximately two or three classifications were reported each minute.

EEG measurements and the data containing indications of seizures were given separately. This made manual merging of the two separate data files necessary. Registrations of time were recorded along with the manually collected blood glucose levels and pre-hypoglycemic state markers. However, time was not registered simultaneously with the automatic EEG and ECG recordings. In the merging process we assumed that the first recording in the manually collected data corresponded to the first observation of the EEG and ECG signals. How-

ever, the lengths of the time intervals covered by the two data sets were not identical, showing differences of up to 30 seconds. This should not be a problem for the analysis, though, since we wish to detect seizures that last between 10 and 20 minutes. At time intervals of such lengths, uncertainties of 30 seconds are negligible.

About 75% of the data contains non-seizure activity, while about 25% contains periods of seizure activity.

3.1 Experimental setup

The data for each subject was obtained in a session in which insulin was given intravenously, thereby inducing a hypoglycemic state. The subjects were asked to count and calculate aloud for monitoring purposes. When they were no longer able to do this, glucose was administered in order to return the subject to a normal level of blood glucose [24].

Electrodes were placed on the scalps of experimental subjects according to the international 10-20 system [28], as shown in figure 3.1. Also, an additional electrode was used to record the electrocardiogram (ECG), i.e. the heart beat. An average reference was used, such that the measured potentials are the differences between the potentials measures at the electrodes and the average of all electrodes.

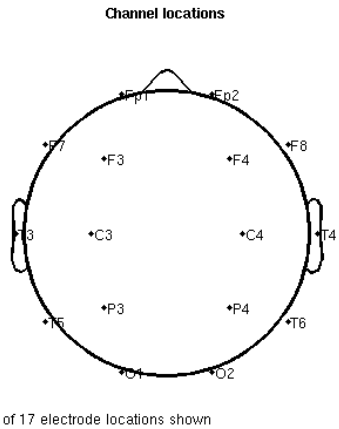

Figure 3.1: Electrode placements on scalp along with 10-20 system labels. The image is from EEGLab [14]. The electrode that is not shown is the ECG recorder.

Chapter 4

Artifact removal

During a visit to the Swartz Center for Computational Neuroscience (SCCN) in the fall of 2010, two studies containing manually labeled ICs were made available. This data made it possible to investigate how ICs can be automatically classified.

Each study contains data recorded from several subjects. Hence several data sets are included in each study. An ICA decomposition is associated with each data set. Approximately 20 ICs from each data set were manually labeled. We assume that the unlabeled components are mixes of neural and artifactual activity.

Acquisition of data is described in the papers [46, 23].

One data set was acquired for the purpose of studying the EEG during different emotional states [46]. This study contained recordings from 35 subjects. These recordings were taken with an active reference(Biosemi) [46]. Recordings were taken using 250 scalp channels. Channels that showed highly abnormal activity were removed, leaving 134-235 channels for each subject. The ICA decomposition for these data sets were obtained by "full-rank decomposition by extended infomax ICA using the binica function" [46]. We will refer to this data set as the "emotion data set" or simply "emotion data" in this report.

The other data set was recorded to investigate how attention is guided early in visual processing. These recordings were "referenced to Cz and re-referenced off-line to linked mastoids" [23]. EEG was recorded from 64 scalp channels. This study contained data sets from 12 subjects. ICA was performed with the implementation in the Brain Vision Analyzer software (Brain Products), which uses the infomax algorithm [22]. We will refer to data from this study as the "cue data set" or "cue data".

Since both more subjects and more channels were used in the emotion study than in the cue study, substantially more independent components are present in the emotion study.

The data sets in both studies contained ICs with the labels eye blink, neural, heart, lateral eye movement, and muscle. The ICs that were not labeled seemed to, based on visual inspection, represent mixed ICs containing both artifactual and neural signals. We will refer to the unlabeled ICs as mixed or unlabeled ICs interchangeably. Neural ICs are the ICs that correspond to activity generated by neural sources within the brain. The other labeled ICs represent artifacts, that we wish to remove from data.

Part III

Methods

This part is devoted to the theory used in the project. We both describe the theory that was used, and how we employed that theory to reach the aims of the project. Chapter 5 contains all material related classification of ICs. The theory related to detection of seizures is described in chapter 6.

Chapter 5

Artifact removal

This chapter is dedicated to explaining how we built automatic classifiers of ICs, and how we determined the best classifier. The relation of this work to the detection of seizures lies in improving the seizure detection rate by learning a model from data without artifacts instead of using a model based on data with artifacts. The step of classifying ICs and removing artifacts is the part that is not grayed out in figure 5.1.

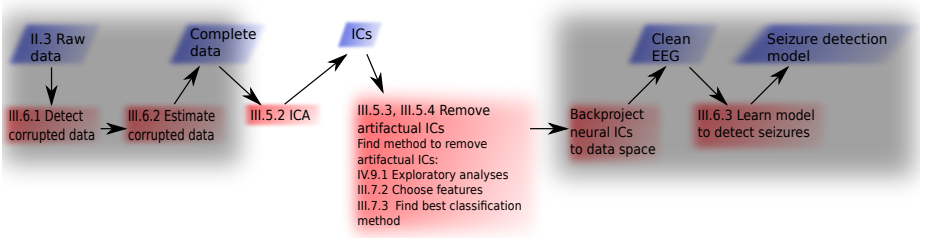

Figure 5.1: This flowchart shows the relation of the removal of artifacts to the rest of the project. The parts that are gray relate to seizure detection, and are not covered in this chapter. Conversely, the part that is not gray shows the steps that are described in this chapter.

The HypoSafe device will be implanted under the scalp to minimize the inter-

ference in daily life of patients. Also, the HypoSafe device will consist of just one sensor, which will only be able to record neural activity within the brain. Since seizure detection must then rely solely on neural activity, it is important to know whether data without artifacts can be classified as well as data with artifacts. Automatic classification of ICs will facilitate comparisons of detection rates in data with artifacts to detection rates in data without artifacts.

More generally, automatic classification of ICs will enable non-experts to use methods that rely on ICA, and will save experts considerable time currently spent on classification of ICs.

5.1 Construction of classification methods for independent components

Our strategy to find an automatic classifier of ICs is simply to compare different classifiers that tend to perform well in most contexts. The model that achieves the best performance will be chosen.

Before ICs can be classified, it is obviously necessary to find the ICs in a data set. ICs are found using the method ICA, which is described in section 5.2.

The features that were chosen as the basis for classification are described in section 5.3. We then discuss model complexity in subsection 5.3.3. Next, we describe a criterion for choosing the features to include in subsection 5.3.4.

In section 5.4, we discuss issues related to classification. First, we must determine the classes of ICs that we want to be able to detect. We discuss different choices in subsection 5.4.1. The influence of misclassification costs on the final classifier is discussed in subsection 5.4.3. Cross-validation, which was used to enable statistical comparisons of models, is described in subsection 5.4.4. The classification methods that we used are described in subsection 5.4.5. These classification methods are compared qualitatively in subsection 5.4.6. Finally, we describe how we compare models to determine the best one empirically in subsection 5.4.7.

5.2 Independent component analysis

Independent components (ICs) of data are found by independent component analysis (ICA) algorithms. After describing ICA, relate the mathematical quantities to biophysical concepts. Independent component analysis (ICA) algorithms often assume that data has been centered and "pre-whitened" [25]. This just means that rows of the data should be uncorrelated, have zero means, and variances of one for each row. An easy way to achieve this is to first subtract the row means from the columns, ensuring that data is centered. A principal component analysis (PCA) of the centered data then gives the desired centered and pre-whitened data. By dividing each row by its own variance, we also obtain unit variances.

We start out by explaining how to center data before moving on to PCA. Then we describe ICA and the interpretation of results from ICA on EEG data. Since we use ICA to find ICs in EEG data, we use the notation for EEG recordings.

5.2.1 Centering data

Let the $m \times n$ matrix Y denote the raw data of n observations of m explanatory variables. The vector of row means $\bar{Y}$ of Y is

$$\bar{Y} = \frac{1}{n} \sum_{j=1}^{n} Y_{:,j}. \tag{5.1}$$

The centered data $\tilde{Y}$ is then found by subtracting $\bar{Y}$ from each column of Y.

$$\tilde{Y}_{:,j} = Y_{:,j} - \bar{Y}.$$

5.2.2 Standard Principal Component Analysis

Principal component analysis (PCA) is a method that finds an alternative representation of data by rotating the coordinate system. All axes in the rotated coordinate system are orthogonal and thus pairwise uncorrelated. Each axis is referred to as a principal component (PC). These PCs have a natural ranking determined by the variance of the projections of the observations onto each PC.

The first PC is the axis along which observations vary most. The second PC is the axis along which observations vary second most, with the condition that the second PC must be uncorrelated with the first PC. By choosing only the first two or three components, the directions in high-dimensional data that explain most variance can be visualized.

To find $r \leq m$ PCs, we need to find a linear transformation of data that complies with the demand of no pairwise correlation. Also, the first PC must explain most variance, the second PC second most, and so on. To begin, we need the covariance of Y. The maximum likelihood estimate of $Cov(Y)$ is [57, p. 66]

$$Cov(Y) = \frac{1}{n}\tilde{Y}\tilde{Y}^T. \tag{5.2}$$

Let $\boldsymbol{u}_1$ denote the first principal component. To maximize the variance of the observations projected onto $\boldsymbol{u}_1$, we set

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}}(Cov(\boldsymbol{u}^T Y)) = \arg\max_{\boldsymbol{u}}(\boldsymbol{u}^T Cov(Y)\boldsymbol{u}).$$

Increasing the magnitude of $\boldsymbol{u}_1$ would clearly increase the value $\boldsymbol{u}_1 Cov(Y)\boldsymbol{u}_1^T$. Since only the direction of $\boldsymbol{u}_1$ is interesting, we introduce the constraint $\|\boldsymbol{u}_1\|_2 = 1$ to ensure that a non-trivial solution is found [6, p. 563]. This results in the constraint

$$\boldsymbol{u}_1^T\boldsymbol{u}_1 = 1 \Leftrightarrow 1 - \boldsymbol{u}_1^T\boldsymbol{u}_1 = 0,$$

on the maximization problem. We reformulate to the unconstrained optimization problem

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}}(\boldsymbol{u}^T Cov(Y)\boldsymbol{u} + \lambda(1 - \boldsymbol{u}^T\boldsymbol{u})),$$

where λ is a Lagrange multiplier [6, pp. 562, 707]. The solution $\boldsymbol{u}_1$ to the maximization problem is found by setting the derivative of $\boldsymbol{u}_1^T Cov(Y)\boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^T\boldsymbol{u}_1)$ with respect to $\boldsymbol{u}_1$ equal to zero, and solving for $\boldsymbol{u}_1$.

$$\begin{aligned}\frac{\partial \boldsymbol{u}_1^T Cov(Y)\boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^T\boldsymbol{u}_1)}{\partial \boldsymbol{u}_1} = 0 \Leftrightarrow \\ Cov(Y)\boldsymbol{u}_1 - \lambda_1\boldsymbol{u}_1 = 0 \Leftrightarrow \\ Cov(Y)\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_1.\end{aligned}$$

The equation $Cov(Y)\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_1$ is an eigenvalue problem, where $\boldsymbol{u}_1$ is the eigenvector and λ_1 is the eigenvalue. Since λ_1 is a scalar and $\|\boldsymbol{u}_1\|_2 = 1$, the following manipulations are possible

$$\begin{aligned}Cov(Y)\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_1 \Leftrightarrow \\ \boldsymbol{u}_1^T Cov(Y)\boldsymbol{u}_1 = \boldsymbol{u}_1^T\lambda_1\boldsymbol{u}_1 \Leftrightarrow \\ \boldsymbol{u}_1^T Cov(Y)\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_1^T\boldsymbol{u}_1 \Leftrightarrow \\ \boldsymbol{u}_1^T Cov(Y)\boldsymbol{u}_1 = \lambda_1.\end{aligned}$$

Thus the variance of data projected onto $\boldsymbol{u}_1$ is equal to the eigenvalue λ_1. By choosing the first principal component to be the eigenvector with the largest eigenvalue, we maximize the variance of the projected data.

Since eigenvectors are orthogonal to each other, the eigenvectors of $Cov(Y)$ give the PCs. The corresponding eigenvalues are the variances of the projections of data onto each PC, making it easy to order the PCs by variance.

Let U be the matrix of eigenvectors in its columns, such that the j^{th} column of U is the J^{th} PC. The data in the rotated coordinate system $\check{Y}$ is found by projecting the data onto the eigenvectors

$$\check{Y} = Y \times U.$$

The final whitened data matrix $\hat{Y}$ must also have unit variances of each column. This can be achieved by dividing column j of $\check{Y}$ by the j^{th} eigenvalue. Then $E(\hat{Y}) = \mathbf{0}_{1\times r}$ and $Var(\hat{Y}) = I_{r\times r}$.

A more efficient approach to find the PCs exploits the Singular Value Decomposition of Y [57, p. 66].

5.2.3 Independent component analysis

Independent component analysis (ICA) is similar to PCA in that an alternative data representation in a rotated coordinate system is found. The assumption behind ICA is that the observed data is a linear mix of a number of processes, out of which at most one is Gaussian.Since the sum of random variables is closer to the normal distribution than any one of them (the Central Limit Theorem), the sources that generate data should be estimable by maximizing the non-Gaussianity of each row of S. Each row of S contains the time series of an estimated source. We denote the row-centered matrix of sources by $\hat{S}$. Then we have

$$\hat{S} = W\hat{Y},$$

where W is the unmixing matrix that gives the linear combinations of observed signals to yield the independent sources. Since uncorrelated Gaussian variables are independent, a unique solution cannot be found if more than one row of $\hat{S}$ is Gaussian.

To minimize the resemblance to Gaussian variables, it is necessary to quantify this resemblance. A common measure of non-Gaussianity is "negentropy". This term is defined as [25]

$$J(\hat{S}_{i,:}) = H(\mathcal{Z}) - H(\hat{S}_{i,:}),$$

where $\mathcal{Z} \sim \mathcal{N}(\mathbf{0}_{1\times n}, I_{n.n})$, $H(\mathcal{X})$ is the entropy of a random vector $\mathcal{X}$, and $J(\mathcal{X})$ the negentropy of $\mathcal{X}$. The standard normal random vector $\mathcal{Z}$ can be simulated, but H must be estimated by a non-quadratic function [25]. Common choices are hyperbolic tangent, the fourth moment, or the log of hyperbolic cosine [25, 14].

ICA is only unique up to scale and sign since W and $\hat{S}$ can be multiplied by the same factor without disrupting the equality

$$\hat{S} = W\hat{Y}.$$

To obtain uniqueness up to sign, a constraint on the norm of one of $\hat{S}$ or W can be used. In the algorithm fastICA, for example, the constraint that the 2-norm of each row of $\hat{S}$ must be one is used [25].

Once the unmixing matrix W that maximizes the non-Gaussianity of the rows of the matrix $\hat{S}$ has been found, the mixing matrix

$$A = W^{-1},$$

which mixes the generating sources (rows of $\hat{S}$) to produce the observed data Y, can be found. The matrix A is found as follows

$$\hat{S} = W\hat{Y} \Leftrightarrow W^{-1}\hat{S} = \hat{Y} \Leftrightarrow A\hat{S} = \hat{Y}.$$

To obtain the source activity S, we apply the unmixing matrix to the raw data Y

$$S = WY.$$

5.2.4 Independent component analysis of EEG data

When ICA is applied to EEG data, the matrices A and S have biophysical interpretations. The source activity in the rows of S can be understood to be the activity of areas in the brain that control biological functions. Since the rows of the matrix Y represent the activity recorded at each electrode, the matrix A gives the linear combination of signals from inside the brain. The element $A_{i,j}$ in the matrix A gives the contribution of the j^{th} generating source to the mix recorded at electrode i.

Each IC is expressed in two ways. Firstly, the activity of the j^{th} source over time is found in the j^{th} row of the matrix S. Secondly, the electrical activity on the scalp caused by the j^{th} source is contained in the j^{th} column of the matrix A. The electrical activity on the scalp can be visualized by color coding each degree of electrical activity in a color. By interpolating the electrical activity between electrodes, based on that observed at electrodes, a complete colormap of the electrical activity over the scalp is obtained. An example of such a colormap is shown in figure 5.2.

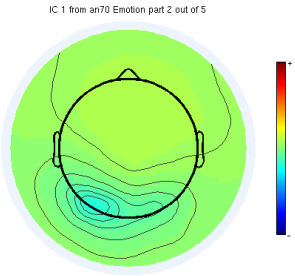

Figure 5.2: Example of the map of electrical activity over the scalp of a generating source, also referred to as an IC.

5.2.5 Standardization of scalp maps

For the purpose of classification of ICs, it must be possible to compare ICs. Thus some normalization of each IC is required. We require that the 2-norm of each scalp map must be one. The normalization factor a for the j^{th} component is then

$$a = \sqrt{\sum_{i=1}^{m} A_{i,j}^2}.$$

To obtain the normalized IC, the j^{th} column of A is divided by the factor a. For consistency, the j^{th} row of S multiplied by a.

5.3 Features of independent components

We now describe the features that were extracted from data and used for classification. The features were chosen to reflect hypotheses about structures in data useful for distinguishing between classes. Some of the features described here have already been described in a previous report in which the problems of classifying ICs and distinguishing between individuals based on their EEG were treated.

After describing the features, we state how the feature matrix was standardized.Then we discuss the trade-off between more complex models that fit data

better, and the risk of over fitting. Next, we describe a way of selecting features that we evaluated.

5.3.1 Features

Temporal and spectral features of the i^{th} IC are calculated using the time series of activation of that component, which is the i^{th} row in the matrix S. The i^{th} column of the matrix A is used to calculate spatial features.

Some features were taken from the literature on IC classification [41, 56] while the use of wavelet analysis for the ECG feature was suggested by Scott Makeig.

First we describe temporal features, followed by spectral and then spatial features. Next, we give an overview of how features were implemented. We then end this subsection by listing the correspondence between the features that we use, and identifying numbers that we use to refer to the features in the remainder of the report.

5.3.1.1 Temporal features

- Temporal kurtosis

  Kurtosis is related to the fourth moment of a distribution, and quantifies the peakedness of a distribution. Thus high kurtosis in the time series of a component indicates that the time series has few large values, while most values lie close to the mean. Such a pattern is characteristic of artifacts such as eye blinks and the pulse due to the QRS complex in the heart rhythm.

- Maximal variance

  The maximal variance is calculated by taking the epoch with highest variance and dividing this variance by the mean of variances over epochs. If there are no epochs, but one continuous signal, then this feature will be 1 for all components, which is clearly non-informative. We remedied this by splitting data into intervals if epochs were not present. The split of data was only used for the calculation of this feature.

  The time series of a vertical eye movement IC attains high values consistently and infrequently throughout a trial. This behavior is similar to that of the time series for a blink component. However, the duration of vertical eye movements is longer than that of blinks. Thus kurtosis values are higher for blinks than for vertical eye movements. This means that the

highest kurtosis values distinguish blinks, but not vertical eye movements. Instead, the variance of the time series may be used. High maximal variances together with medium kurtosis values can then be used to detect vertical eye movements.

- Temporal skewness

  The skewness of a distribution is related to the third moment and quantifies the degree to which the distribution is asymmetrical. As suggested in [56], this feature was calculated by splitting the time series into intervals of 15 seconds and then finding the average of the skewness in these intervals. Since eye blinks cause high potentials for a short interval, eye blinks increase the asymmetry of the distribution. This causes skewness for ICs representing eye blinks to be either highly negative or positive [53].

- Logarithm of range of time series

  The range of a time series is higher for some artifacts, such as loose electrodes, than for all other types of components. By taking the logarithm of the range, differences of orders of magnitude are emphasized.

- Wavelet based features for ECG detection

  In previous attempts at component classification, components representing the ECG turned out to be the most frequently misclassified components. Thus we introduce a feature specifically aimed at detecting ECG components by exploiting the shape of the QRS complex. The QRS complex is a pattern that occurs once in every heart beat, and is the part of the heart's cycle with the largest amplitude. Thus components representing the pulse are likely to exhibit patterns similar to the QRS complex at regular intervals.

  Discrete wavelet analysis (see section I.1 for background on wavelet analysis) was used to search for the QRS complex, which is a part of the ECG. The ECG detection features quantify the degree to which the time series of an IC resembles the QRS pattern in ECG. Details are given in appendix I.

5.3.1.2 Spectral features

EEG is often spoken of in terms of the frequency of the signal. The frequency bands that EEG is usually split into are shown in table 5.1.

- Logarithm of mean of spectrogram

  The spectrogram quantifies the amount of activity in each frequency band. In ICs, the bands that show most activity is highly dependent on the type

Band name	Frequency range (Hz)
Delta (δ)	<4
Theta (θ)	4 - 7
Alpha (α)	8 - 12
Beta (β)	12 - 30
Gamma (γ)	>30

Table 5.1: The generally accepted EEG frequency range splits and the names usually given to these bands.

of component. Muscle components, for example have highest means in the β- and γ-bands, whereas neural components tend to have more activity in the lower frequency bands.

A spectrogram was calculated at every 0.5 frequency starting from 0.5 and ending at 40Hz. The mean was then calculated in frequency bands known to have biological relations (δ-band ($<$4Hz), the θ-band (4-7 Hz), the α-band (8-12Hz), the β-band (12-30Hz), and the γ-band ($>$30Hz)). Next, the logarithm was taken to emphasize differences in orders of magnitude. The frequencies in each band are summarized in table 5.1.

The spectrogram was calculated using the MatLab function *spectrogram* with defaults. The defaults specify the number of windows that the time series is split into (eight), and the type of windowing that is used (Hamming).

- Closeness of power curve to typical EEG power curves

  Since components representing neural activity should exhibit characteristics typical of normal, clean EEG, this measure is expected to discriminate between neural and artifactual components.

  A power curve is a relation between the frequency and the amount of activity at that frequency, i.e. the power. To quantify the resemblance between a time series and typical EEG using the power curve, we first calculate the spectrogram as above. A curve is then fitted to the relation

$$power = f(frequency) = \frac{constant}{frequency^{\lambda}}.$$

  This fit is based on three (frequency, power) coordinates, at the frequencies: 2Hz, the frequency at which the power is minimal between five and 13 Hz, and the frequency at which the power is minimal between 33 and 39 Hz. $\log(\lambda)$ is then a measure of the closeness to typical EEG power curves [56], and this is used as a feature.

- Fit error of power curve to typical EEG power curves

  The difference between the power curve fitted as described above and the actual power curve, quantified as the sum of squared residuals, is also used as a feature. High values of fit errors are expected to indicate artifacts since such values indicate that it was not possible to fit even the shape of the typical curve.

- Hurst exponent

  The Hurst exponent (sometimes referred to as the H-exponent in the literature) gives a measure of the autocorrelation of a time series [50]. We hypothesize that different kinds of components exhibit autocorrelation to different degrees, motivating the inclusion of the Hurst exponent as a feature. Three different estimates of the Hurst exponent, provided by the Matlab function *wfbmesti*, are used. All three estimates are used because one estimate might be better suited to EEG data than the others. By including all three, and performing feature selection, we hope to identify the best estimate. This redundancy also provides a check on the validity of the feature selection.

5.3.1.3 Spatial features

The nature of independent components allows the inclusion of spatial features, which cannot be included meaningfully in analysis of single channel signals. Informally, an independent component is a spatial pattern of activity in the brain that is active independently of other patterns of activity, and that reappears consistently in data. For formal details, see subsection 5.2. Such spatial distribution information is helpful in the classification of some types of components.

- Generic discontinuity

  An independent component that represents a generic discontinuity, such as a loose electrode, is spatially concentrated in a small area on the scalp. The difference between the IC activation at an electrode and the average IC activation in the ten nearest electrodes quantifies the degree of spatial concentration. By calculating this measure for all electrodes, and using the maximum as the feature, the overall degree of peakedness of the spatial distribution of activity in the component is quantified.

- Spatial eye difference (SED)

  This feature is the absolute value of the mean of the weights of the left eye electrodes minus the mean of weights of the right eye electrodes. For the j^{th} component, the mathematical expression is

$$\left| \frac{1}{|E_{left\_eye}|} \sum_{i=1}^{|E|} a_{ij} \cdot \mathcal{I}(e_i \in E_{left\_eye}) - \frac{1}{|E_{right\_eye}|} \sum_{i=1}^{|E|} a_{ij} \cdot \mathcal{I}(e_i \in E_{right\_eye}) \right|,$$

where E is the set of all electrodes, E_{left_eye} is the set of electrodes around the left eye, E_{right_eye} is the set of electrodes around the right eye, and e_i is the i^{th} electrode. $\mathcal{I}(\cdot)$ is the indicator function which is 0 if the argument is false, and 1 if the argument is true.

This feature is meant for detection of lateral eye movements which generate high-amplitude activation around both eyes, with opposite signs, making the SED high.

- Average electrode weights over eye areas separately

  In addition to the absolute value of the difference of the means of electrode weights over the eye areas, each separate mean was also used as a feature. That is, the two features

$$\frac{1}{|E_{left\_eye}|} \sum_{i=1}^{|E|} a_{ij} \cdot \mathcal{I}(e_i \in E_{left\_eye}) \quad \text{and} \quad \frac{1}{|E_{right\_eye}|} \sum_{i=1}^{|E|} a_{ij} \cdot \mathcal{I}(e_i \in E_{right\_eye}) \tag{5.3}$$

  were also included.

- Spatial average difference (SAD)

  Independent components representing eye artifacts have much higher activation in frontal areas than in posterior areas of the scalp. Using the same principle as for lateral eye movement detection, the difference in activation between frontal and posterior electrodes can be quantified by the absolute value of the difference of the means of activation in frontal and posterior electrodes. This feature is calculated as

$$\left| \frac{1}{|E_{frontal}|} \sum_{i=1}^{|E|} a_{ij} \cdot \mathcal{I}(e_i \in E_{frontal}) - \frac{1}{|E_{posterior}|} \sum_{i=1}^{|E|} a_{ij} \cdot \mathcal{I}(e_i \in E_{posterior}) \right|,$$

  where $E_{frontal}$ is the set of frontal electrodes and $E_{posterior}$ is the set of posterior electrodes.

- Mean of weights of frontal electrodes

  In addition to the magnitude of the difference in the means of weights of frontal and posterior electrodes, the means of each set were used. These two features are calculated as shown in (5.4), similar to the calculation of component activation for eye areas (5.3).

$$\frac{1}{|E_{frontal}|}\sum_{i=1}^{|E|} a_{ij}\cdot\mathcal{I}(e_i\in E_{frontal}) \quad \text{and} \quad \frac{1}{|E_{posterior}|}\sum_{i=1}^{|E|} a_{ij}\cdot\mathcal{I}(e_i\in E_{posterior}) \tag{5.4}$$

- Variance of weights of frontal and posterior electrodes

  We have already described the feature SAD, which has a high value for eye artifacts. The SAD attains a high value for eye artifacts since the magnitude of the average activation of frontal electrodes is much higher than that for posterior electrodes when a component represents eye activity. However, a high value for SAD may also occur if frontal electrodes have high activity, and one half of the posterior electrodes have negative weights while the other half has positive weights. The variance of posterior electrodes is useful in identifying such cases. The variance of frontal electrodes is also used as a feature since this gives a basis for comparison for the variance of posterior electrodes.

- Number of Talairach areas assigned to the dipole fit coordinates of an IC

  This feature counts the number anatomical areas identifiable as the source of the component activation. The assignment of anatomical areas is based on the coordinates of the fitted dipole source, i.e. the 3D coordinate in the brain from which the activity in the component is most likely generated. The relation between the dipole fit and anatomical areas is found using the Talairach Atlas, as implemented in freely available software [38, 37]. If an IC represents an artifact, it is unlikely that this will have anatomical areas associated with it. Thus this feature should be helpful in identifying neural components.

- Coordinates of dipole fit

  The coordinates of a fitted dipole lie far from the scalp for neural components, but closer to the scalp for many artifactual components, e.g. muscle and loose electrodes. Also, heart components tend to have deep dipole fits, i.e. dipoles that are close to the neck, and central on the x and y axes. We

hope that a classifier can learn these differences if the dipole coordinates are used as features.

- 2D DFT of weights of electrodes

  A spatial pattern with few peaks is indicative of an artifactual component, especially of loose electrodes or muscle artifacts. Conversely, physiological components tend to have smoother distributions. A smooth distribution of spatial distribution of activity is connected with slowly varying (low-frequent) spatial patterns. Thus the spatial frequency of a pattern should aid in classification.

  The spatial frequency is quantified by interpolating A^t to get a 64×64 matrix. A 2D discrete Fourier transform (DFT) is then applied, and the average of the logarithmic band power in higher frequencies on the right half of the DFT calculated [56].

- Central activation of electrodes

  This is the logarithm of the average of the activation of a group of 13 electrodes placed symmetrically around the center of the top of the scalp [56].

- Activation of electrodes on borders

  This is another feature aimed at detection of artifacts with a large amount of activation near the scalp such as muscle components. It is defined as 1 if the most active electrode in the pattern is near the periphery of the scalp, or if the activity of such a peripheral electrode differs more than two standard deviations from the average of the closest electrodes. Otherwise, it is set to -1 [56]

- Range within pattern

  The range of activities of electrodes in a pattern is small for neural components since most electrodes will have little or no activity, while a few will show moderate activity. For some artifacts, such as heart components, however, large negative fluctuations will be seen in one half of the head, while electrodes with positive activities will be placed on the other half. Thus the range of activity of electrodes in patterns might also be helpful for classification [56].

- Residual variance of dipole fit

  The dipole fit for a component in EEGLab is restricted to consist of one dipole, or two dipoles that are symmetrical with respect to the midline. Using conductance properties of the brain, these dipoles are found as the sources from electrical activity might arise, that best explain the activation pattern observed on the scalp. The residual variance is a measure of the misfit between the actual activity in a pattern and that explained by

the dipole fit. This number has been observed to be small for neural components ($< 15\%$) and larger for artifactual components. The threshold of 15% residual variance is typically used by human experts when classifying components.

5.3.1.4 Implementation

Features 1 to 11 were taken from the plug-in ADJUST [41] to EEGLab [14]. The code from ADJUST was modified slightly. We modified the code to allow a different number of ICs from the number of channels. Also, we changed the code for the few features that require data in intervals such that continuous data without epochs is split into smaller intervals. Finally, one of the data sets in the emotion study did not have any electrodes over left eye, as defined in ADJUST. To not throw the entire data set away, we expanded the area over the left eye by four degrees to each side when no electrodes over left eye as defined in ADJUST were available.

The dipole fitting utilities available by default in EEGLab were used to obtain dipole fits.

Christian Kothe implemented the code to calculate the central and border activation features, which were suggested in [56].

The Talairach Atlas, as implemented in the freely available software [38, 37], was also used.

The remaining features were implemented using MatLab, using built-in functions and functions from the Stats and Wavelets toolboxes.

5.3.1.5 Identifiers of features

In the following, we will refer to features by numbers instead of by names. In particular, we avoid cluttered plot axes by referring to features by numerical identifiers instead of by names. The relation between identifiers and feature names are shown here:

1. Generic discontinuity
2. Spatial eye difference

3. Average electrode weights over left eye
4. Average electrode weights over right eye
5. Spatial average difference
6. Variance of weights on frontal electrodes
7. Variance of weights on posterior electrodes
8. Mean of weights on frontal electrodes
9. Mean of weights on posterior electrodes
10. Temporal kurtosis
11. Maximal variance
12. Temporal skewness
13. Logarithm of range of time series
14. Logarithm of mean power in the α band
15. Closeness of power curve to typical EEG power curves
16. Number of Talairach areas assigned to the dipole fit coordinates of an IC
17. z-coordinate of dipole fit
18. Fit error of power curve to typical EEG power curves
19. 2D DFT of weights of electrodes
20. Central activation of electrodes
21. Activation of electrodes on borders
22. Logarithm of mean power in the δ band
23. Logarithm of mean power in the θ band
24. Logarithm of mean power in the β band
25. Logarithm of mean power in the γ band
26. x-coordinate of dipole fit
27. y-coordinate of dipole fit
28. Hurst exponent, *wfbmesti* estimate 1
29. Hurst exponent, *wfbmesti* estimate 2

30. Hurst exponent, *wfbmesti* estimate 3
31. Number of fitted dipoles, either one or two
32. Mean interval between peaks of approximation coefficients found through discrete wavelet analysis at lowest level
33. Variance of intervals between peaks of approximation coefficients found through discrete wavelet analysis at lowest level
34. Fifth percentile of intervals between peaks of approximation coefficients found through discrete wavelet analysis at lowest level
35. 95th percentile of intervals between peaks of approximation coefficients found through discrete wavelet analysis at lowest level
36. Features 36 to 43 are the same features as the above four in the same order, but the wavelet analysis is performed at the middle and highest level.

5.3.2 Standardization of features

Before using the features for either exploratory analyses or classification, we standardize the features. For each feature, we subtract the mean of the feature values. We then divide the centered feature vectors by their variance. The feature vectors that we use thus have zero means and unit variance. We denote the standardized matrix of features by X. Each column corresponds to a feature, and each row to an observation of a vector of features. The number of columns of X is then the number of features, and is referred to as m. The number of rows of X is equal to the number of observations, n.

5.3.3 Bias-variance trade-off

By incorporating more features in a model, the model complexity increases, and becomes better at describing data. A trade-off between describing data well and the risk of over-fitting is always present when modelling data. The more complex a model is, the less the training error will become. The extreme of this scenario is that each observation is modelled separately, achieving a training error of zero. Clearly, though, this approach is insensible, especially for purposes of predicting a new response based on explanatory variables. In this scenario, the variance of fitted values is maximal. On the other hand, the opposite extreme affords minimal variance by e.g. solely including an intercept in a model. A high bias,

i.e. high average difference between the true and the fitted or predicted value, is the side-effect of this strategy. In general, the squared error of a model can be split into a contribution from the bias and a contribution from the variance of the model around any point. The expectation of the squared error is called the "risk" [62, pp. 304-305]. To be more precise, the relations are [62, pp. 304-305]

$$\begin{aligned} Risk =& E\left(\int (f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}))^2 dx\right) = \\ & \left(E(\hat{f}(\boldsymbol{x})) - f(\boldsymbol{x})\right)^2 + E\left(\hat{f}(\boldsymbol{x}) - \left[E(\hat{f}(\boldsymbol{x}))\right]^2\right) = bias^2 + variance, \end{aligned}$$

where f denotes the true relation between explanatory variables and response and $\hat{f}$ the estimate of f. Since both a high variance and a high bias are undesirable, a balance between the two must be found. This is usually achieved by choosing to use the complexity of model that gives the least average squared error over several cross-validation folds.

5.3.4 Choosing the best features

To decrease the computational demands involved in model construction, it is desirable to pick out the best features and use only those when constructing the model. Mutual information (MI) is a measure of the degree to which two quantities vary together, which also takes non-linear changes into account. The MI between class assignments and feature values may then be informative as to which features have good predictive powers of class membership. This relation was also used in the exploratory analyses, and has been suggested by others as a way to identify best features [10, 20, 60].

We will refer to the feature selection method described here as the MI-criterion. First we describe what MI is, and how to calculate it. Then we go on to explain how we use MI to select features.

5.3.4.1 Mutual Information

Mutual information (MI) is a measure of independence between two probability distributions. It measures the difference between the actual joint distribution of the two distributions and the product of each their marginal distributions. If the two random variables were independent, their joint distribution would equal

the product of the marginals. A higher mutual information between two random variables means that more information is gained about one of the two when the other is observed. For independent random variables, the information gained about one is non-informative with respect to the other. Independent random variables have the lowest MI, of zero.

Let $\mathcal{V}$ be a random variable with density $f_{\mathcal{V}}$ and $\mathcal{W}$ a random variable with density $f_{\mathcal{W}}$. Denote by $f_{\mathcal{V},\mathcal{W}}$ the joint density of $\mathcal{V}$ and $\mathcal{W}$. The MI between $\mathcal{V}$ and $\mathcal{W}$ is then defined as 5.5 [6, p. 57]

$$MI(\mathcal{V},\mathcal{W}) = -\int\int f_{\mathcal{V},\mathcal{W}}(v,w)\log\left(\frac{f_{\mathcal{W}}(w)f_{\mathcal{V}}(v)}{f_{\mathcal{V},\mathcal{W}}(v,w)}\right)dvdw, \tag{5.5}$$

where the logarithm is the natural logarithm. Naturally, the integrals are taken over the entire domains of $\mathcal{V}$ and $\mathcal{W}$. We briefly note that the mutual information can also be defined as the Kullback-Leibler divergence between the distributions $f_{\mathcal{V}}(v)f_{\mathcal{W}}(w)$ and $f_{\mathcal{V},\mathcal{W}}(v,w)$. However, the Kullback-Leibler divergence will not be needed for other purposes here, and we refer the interested reader to other references [6, p. 55],[57, p. 561] for an explanation of this concept.

5.3.4.2 Estimating mutual information between variables in data

To estimate the mutual information between observed quantities, it is necessary to estimate both marginal and joint distributions. Once this has been accomplished, we may plug these estimates into (5.5), substituting integrals with sums. The problem is then reduced to finding estimates of the distributions of the concerned quantities.

Denote by $\hat{f}_{\mathcal{V}}$ the estimate of the distribution $f_{\mathcal{V}}$, by $\hat{f}_{\mathcal{W}}$ the estimate of $f_{\mathcal{W}}$ and by $\hat{f}_{\mathcal{V},\mathcal{W}}$ the estimate of the joint distribution $f_{\mathcal{V},\mathcal{W}}$. We will find $\hat{f}_{\mathcal{V},\mathcal{W}}$ first and then use this estimate to estimate the marginal distributions. We will think of the joint distribution as a number of rectangles in each of which a joint observation of $\mathcal{V}$ and $\mathcal{W}$ has a probability of falling.

Clearly, we must both decide how to define these rectangles and how to estimate the probability that an observation $(\mathcal{V},\mathcal{W})$ falls into any of these. We decided to split the range of $\mathcal{V}$ and the range of $\mathcal{W}$ into equally many intervals. We denote the number of intervals by r. To obtain a uniform coverage with observations of the joint outcome space, we split the ranges using estimated percentiles. The $(i,j)^{th}$ rectangle then contains observations (v,w) such that $f_{\mathcal{V}}^{-1}((i-1)/r) < v \leq f_{\mathcal{V}}^{-1}(i/r)$ and $f_{\mathcal{W}}^{-1}((j-1)/r) < w \leq f_{\mathcal{W}}^{-1}(j/r)$. In this presentation we assumed

that both $f_{\mathcal{V}}$ and $f_{\mathcal{W}}$ are bijective density functions, such that $f_{\mathcal{V}}^{-1}(q)$ gives the q^{th} quantile, and $\mathbb{P}(V \leq f_{\mathcal{V}}^{-1}(q)) = q$. By counting how many observations each rectangle contains, and dividing by the total number of observations, we get a stepwise estimate of the joint density.

By summing over all values of w for each interval of v values, we get a stepwise function that estimates the marginal distribution of v. Conversely, interchange the roles of v and w to get a stepwise function estimating the marginal distribution of w. This provides the estimates $f_{\mathcal{V}}$, $f_{\mathcal{W}}$, and $f_{\mathcal{V},\mathcal{W}}$ needed to estimate the mutual information.

Since we make the bins contain the same fraction of observations for each variable, the mutual information between a variable and itself will be identical for all variables.

5.3.4.3 Implementation

Some practical issues arise when the estimation of mutual information is implemented.

Firstly, the number of bins needs to be determined. We were not able to find texts discussing the choice of the number of bins when widths are chosen by percentiles. We ended up taking the number of bins used in the code *minfo.m* by Jason Palmer. The number of bins r used in *minfo.m* is the minimum of 50 and $3\log_2(1 + n/10)$, where n is the number of observations of each variable.

Another issue arises when variables only take on a few distinct values. The strategy of estimating the probability distribution through the cumulative probabilities fails in this case since several percentiles will take the same value. Thus an observation will be placed in several boxes. To avoid this problem, we counted the number of observations in each box instead of estimating the cumulative distribution. When a variable only has few distinct values, equal proportions of observations will not be put into each box. Thus variables with few values will not have the same MI with themselves, as all other variables do.

The code, which has not been optimized, is given in appendix L.

5.3.4.4 Constructing the null distribution

Mutual information was used to identify best features by finding the null distribution of mutual information between class assignment and feature value for each feature. The null-distribution corresponds to the null hypothesis that the feature has no relation to the class assignment. We can draw from the null distribution by randomly permuting the vector of class labels and calculating the mutual information between this vector and the values of feature currently being evaluated. An empirical estimate of the null distribution can be obtained by making a large number of such draws. This null distribution can then be used to test the null hypothesis that the mutual information between the feature and the non-permuted vector of class labels was drawn from the null distribution. If rejected, we have evidence that the feature does vary with the class label.

5.3.4.5 Testing in the null distribution

We do not wish to make assumptions on the distribution of the values of mutual information, and thus use a non-parametric test. If the mutual information between the feature concerned and the vector of class labels is significantly higher than the median in the null distribution, then we have evidence that the feature is not from the null distribution, and performs better than would be expected under the null distribution. We have the following hypotheses

$$H_0 : MI_{observed} \leq \tilde{\mu}_0$$
$$H_1 : MI_{observed} > \tilde{\mu}_0,$$

where $\tilde{\mu}_0$ is the median of the null distribution and $MI_{observed}$ is the mutual information between the concerned feature and the observed vector of class labels. If the null hypothesis is true we would expect half the drawn samples from the null distribution to be greater than or equal to $MI_{observed}$. Denote the number of draws from the null distribution that are greater than or equal to $MI_{observed}$ by $n_{\geq MI_{observed}}$. Assuming the null hypothesis is true, the random variable $N_{\geq MI_{observed}}$, denoting the number of draws greater than or equal to $MI_{observed}$, follows the binomial distribution with the number of trials equal to 100 since this is the number of draws from the null distribution. The parameter denoting the probability of success is equal to 0.5 since the probability of drawing an element less than or equal to the median is per definition 0.5, implying that the probability of drawing an element greater than the median is 1-0.5=0.5. The

probability of getting $n_{\geq MI_{observed}}$ or fewer observations greater than or equal to $MI_{observed}$ under the null hypothesis is then $\mathbb{P}(N_{\geq MI_{observed}} \leq n_{\geq MI_{observed}})$.

We use the significance level 0.05 with the Bonferroni correction for multiple hypothesis tests. The Bonferroni correction for multiple hypothesis test simply divides the significance level α by the number of tests, and uses this quotient as the significance level in each test. Then the significance level for all the tests is at most the desired level α. The Bonferroni correction is very conservative, meaning that the significance level used in each test may be too low. More advanced corrections for multiple hypothesis tests exists, but delving into these were out of the scope of this project. Also, the choice of a conservative correction method ensures that informative features are retained.

There are 43 features, and hence 43 hypothesis tests. Thus a feature is deemed to have significant predictive power if $\mathbb{P}(N_{\geq MI_{observed}} \leq n_{\geq MI_{observed}})$ is less than $\frac{0.05}{43}$. If a feature has significant predictive power, we choose it for inclusion. We will refer to this feature selection method as the MI-criterion.

5.4 Classification of independent components

In this section, we describe different issues related to classification of ICs.

Each observation is of the form $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is a vector of features and y the class of the object from which the features were calculated. Based on a number observations of both $\boldsymbol{x}$ and y, a classification rule is sought that will allow prediction of y based on observations of features.

First, we discuss how to set up the classification problem in terms of which classes of ICs we wish to automatically identify in subsection 5.4.1.

Next, we discuss how binary classifiers can be used to solve classification problems with more than two classes in subsection 5.4.2.

Due to large differences in the sizes of the different classes, we need to tackle the problem of disproportionate classes. We explain how this problem may negatively affect classification methods and how we tackled the issue in subsection 5.4.3.

In subsection 5.4.4, we explain the concept of cross-validation, which we use to determine parameters in model estimation, and to enable statistical comparisons of models.

We give descriptions of the classification methods that were used in subsection 5.4.5. In addition to describing how the classification methods work, we explain how the variance of feature coefficient estimates can be found analytically, where possible. The variance estimates can then be used to find the coefficient estimates that are not significantly different from zero. Features with coefficient estimates not significantly different from zero can either be removed from the model, or at least be assigned less importance when interpreting the model.

5.4.1 Types of independent components considered

A natural limit on the classes that can be incorporated in a classification method is the type of class labels available in training data. Obviously, a classification method cannot learn a class that is not present in the training phase. In the data from the Swartz Center for Computational Neuroscience (SCCN), five types of labels for ICs were present. These were eye blink, neural, heart, lateral eye movement, and muscle. Additionally, about a little more than half the ICs were unlabeled.

We need to choose which classes to include in training, since this will determine which types of ICs can henceforth be automatically classified. In making these choices, different concerns must be taken into account.

5.4.1.1 Possible splits of classes

First, we need to consider which classes might be advantageous to identify. For most purposes, it is desirable to identify ICs that are very likely to be artifacts.

A classification method that distinguishes between each type of artifact and neural components would be able to detect artifactual ICs, but ignore ICs that contain both neural and artifactual activity.

Binary classification into the classes neural or artifactual would suffer from the same limitation of not being able to properly handle mixed ICs.

In both scenarios, we could include a sixth or third case, respectively, containing unlabeled ICs.

Yet another option would be to include the unlabeled ICs in the non-neural or neural class in the binary setting.

5.4.1.2 Difficulty of problem

The choice of which classes to include is likely to influence the difficulty of the problem. Since the class of unlabeled ICs is not well defined, it is likely that inclusion of this class will make classification more difficult. This extra difficulty will probably ensue both if unlabeled ICs are included as a separate class, or as part of another class.

The combination of all artifacts into one class may either make the problem easier or harder. It is difficult to reason whether the combination of artifacts in one class will make the problem easier due to fewer classes in the problem, or more difficult since the examples of the class differ more.

5.4.1.3 Retaining neural activity

For most purposes in EEG analyses, it is important not to lose information on neural activity. Hence retaining all activity that is not artifactual should be a priority. This calls for including a class of mixed ICs to catch those ICs that contain some neural activity, but are not neural components. The inclusion of such a class would allow denoising by removing only artifactual ICs, while ensuring that neural activity is not lost.

In real data, mixed ICs will almost always be present. Investigation of the problem without mixed ICs is interesting since this will give an idea of how easy it is to separate neural from artifactual components. Also, as explained below, probabilities of class membership from a model trained only on labeled ICs will probably also be useful for real data that includes mixed ICs.

5.4.1.4 Probabilities in identification of class ambiguity

Assuming that the chosen classifier gives probabilities for membership of each class, such probabilities could aid the retention of mixed ICs. Instead of just assigning the class with highest probability, a threshold probability might be used to determine when to assign a class. If the probability of membership of any class does not exceed the threshold, no assignment would be made. No class assignment would then indicate that an IC is of the mixed type. The model could then be trained on data with no mixed ICs, since the absence of mixed ICs will most likely make it easier for a model to learn the different classes.

Such a procedure would guard against the loss of neural activity by assignment of mixed components to artifactual classes.

Additionally, such a solution would be applicable in any of the scenarios described. Thus the problem of mixed ICs could be circumvented through the use of probabilities.

5.4.1.5 Classes considered in this project

To limit the project, we focused on four of the scenarios described here. The classification problems that we investigated are shown in table 5.2.

	Inclusion of labeled ICs only	Inclusion of all ICs
Binary classification	Two classes, neural vs. non-neural	Two classes, neural vs. non-neural and mixed ICs
Multiclass classification	Five classes, the types of labels present in data	Six classes, the types of labels present in data and mixed ICs

Table 5.2: The four classification problems investigated in this project.

We investigated classification of neural vs. artifactual ICs, both without including unlabeled ICs, and with inclusion of unlabeled ICs in the class of artifacts.

In addition to the two scenarios of binary classification, we investigated classification into multiple classes. Two sets of classes were considered for this. The first set included only the ICs that were labeled, amounting to the classes blink, neural, heart, lateral eye movement, and muscle. The second set included the class of unlabeled ICs in addition to the five classes of labels present in data.

5.4.2 Handling multiple classes

Several of these classification methods were originally designed to solve binary classification problems, where each observation belongs to one of just two classes. An issue then arises when observations may be classified into one of three or more classes. Two strategies easily generalize binary classifiers to multiple class problems. Denote the number of classes by K.

One-vs-one (1v1) Firstly, one may use what is called the one-versus-one (1v1) strategy. In this strategy, an observation is classified using the binary model arising from each combination of two classes. A number between zero and one can then be found for each class. If each binary model predicts a class number, a number between zero and one can be found by counting the number of times an observation was classified into each class, and dividing by the total number of binary models.If the binary models give probabilities for the observation belonging to each class, these can be summed over the binary models for each class and normalized. The number found in this way for class k may be interpreted as the posterior probability that the observation belongs to class k.

One-vs-rest (1vR) Secondly, the one-versus-rest (1vR) strategy can be employed. In this setting, K models are calculated, one for each class. The model for class k classifies an observation as belonging to class k or not belonging to class k. An observation is then classified by classifying the observation using each of the K models. If the binary classifier returns the class which an observation is assigned to, the final result can be understood as equal probabilities of the observation belonging to each class for which the binary classifier assigned the observation to that class instead of the "the rest" class. On the other hand, the binary classifier might return probabilities of class membership. In this case, the probability that the observation belongs to class k may be taken as the probability returned by the k^{th} binary model. To retain the interpretation of probabilities, these probabilities must be normalized by dividing by the sum of the probabilities of class membership over all classes.

Alternatives to 1v1 and 1vR However, both these strategies have inherent problems [6]. An obvious problem for the 1vR strategy is if an observation is assigned to the "the rest" category by each of the K binary models. Another problem is that of overlapping regions such that an observation is assigned to several classes. This is handled, but not properly solved, by the probability interpretation described above.

Some classification methods are easily adapted to handle multiple classes at once, entirely avoiding the problem of adapting binary models to multi-class models. Linear and quadratic discriminant analysis (LDA and QDA), logistic regression, and decision trees are examples of such models. Support vector machines (SVM), however, are not as easily adaptable.

There is a drawback, though, of taking all classes into account simultaneously. If only few observations from some class are available, then the estimation may

be under determined.

This occurred for quadratic classification, and it was necessary to remove features at random until few enough were left that the model could be estimated.

Out of curiosity to see the performance of the less-than-optimal 1v1 and 1vR schemes, we use the 1vR scheme for LDA and QDA and the 1v1 scheme for logistic regression, as well as the multi-class versions of LDA, QDA, and logistic regression. Since the computational complexity of these models is low, these are cheap comparisons. We use the 1v1 strategy for SVM. We solely use decision trees with the natural ability to handle multiple classes.

Forcing probabilities from classifiers Most of the classifiers investigated here predict class memberships, and not probabilities of class membership for each class. That is, these models return a single number, which is the predicted class for the concerned observation. The only exception is logistic regression, which returns probabilities. However, it can be desirable to get class membership probabilities instead of class assignments in some circumstances, even though the method to get those probabilities may not be theoretically rigorous. In bcilab, LDA, QDA, and SVM are altered to return such probabilities. These are all implemented as binary methods, and the schemes described above then used for multiple classes. Each of these methods returns a real number x. Usually, the observation would be assigned to class one if that number is negative ($x < 0$), and to class two otherwise. To get probabilities, some extra processing is performed. Firstly, the number is changed to -1 if smaller than -1 and to 1 if larger than 1. If it is between -1 and 1, it is unchanged. Thus we may be sure that the number lies between -1 and 1 at this point, $-1 \leq x \leq 1$. The probability that the observation belongs to class one is then calculated as $(1-x)/2$. The probability that the observation is from class two is then defined as $1 - ((1-x)/2)$.

5.4.3 Misclassification costs

Standard assumptions in classification problems are that classes are of the same size, and that all misclassifications induce equal losses. Equal misclassification costs give rise to the objective of minimizing the total number of misclassifications [57, p.20],[6, p.41]. For some classification problems, however, this is not desirable.

When class sizes differ by a large factor, trivial classifiers that classify all observations into the large class can achieve low misclassification rates. Imagine,

for example, a binary classification problem in which 99 training samples are available from the large class and the small class has only one training sample available. Then a training misclassification rate of 1% is achieved by classifying all observations into the large class. Conversely, misclassification of just two observations from the large class into the small class will cause a misclassification rate of 2%. This imbalance in the effects of misclassifying samples from the two classes makes it highly probable that a classifier will learn to classify all observations into the large class if misclassification rate is used as the objective function to be minimized.

In the IC classification problem, the number of samples from different classes differs by up to three magnitudes of ten. Thus it is obvious that we need to take the problem of unequal classes into account.

5.4.3.1 Observation weights

To avoid trivial classifiers, we tweak the learning phase of classifiers by making it more expensive to misclassify samples from the smaller classes. We do this by multiplying the cost induced by each sample by the inverse proportion of the class that the sample is from. Let n denote the total number of training samples, and n_k the number of samples from class k. Let f be the objective function that we seek to minimize. If a sample $\boldsymbol{x}$ is from class k, we then use the cost $(n/n_k)f(\boldsymbol{x})$ instead of $f(\boldsymbol{x})$.

5.4.3.2 Standardization

To keep the costs at manageable magnitudes and decrease the risk of unstable solutions caused by large weights, we divide the vector of observation weights by the sum of the observation weights. Thus the sum of all observation weights in any of the classification problems we solve is one.

5.4.3.3 Weights in 1v1 and 1vR cases

In 1v1 and 1vR voting schemes, explained in detail in subsection 5.4.5, several binary classification problems are solved instead of one multiclass problem. However, the weighting scheme is consistent since the relative weight between any two classes remains constant. In the multiclass problem, the relative weight between class 1 and class 2 is

$$\left(\frac{n}{n_1}\right) / \left(\frac{n}{n_2}\right) = \frac{n_2}{n_1}.$$

In the 1v1 voting scheme, the denominator of the weight of an observation is still the number of observations from the class that the observation is from. The training samples available in the solution of the binary classification between classes 1 and 2 is $n_1 + n_2$, which is thus the numerator of the weight. We find the same relation between the weights of classes 1 and 2

$$\left(\frac{n_1 + n_2}{n_1}\right) / \left(\frac{n_1 + n_2}{n_2}\right) = \frac{n_2}{n_1}.$$

Likewise, the proportion remains the same in the 1vR case. The 1vR scheme solves the binary classification problems of one class versus all others. Hence all training samples are used, and the numerator in the weight of an observation is n while the denominator is still the number of training samples from the class that the observation is from

$$\left(\frac{n}{n_1}\right) / \left(\frac{n}{n_2}\right) = \frac{n_2}{n_1}.$$

This shows that the weighting scheme is consistent no matter how the multiclass problem is solved.

5.4.4 Cross-validation

Cross-validation is a technique that can be used to find the best values of parameters in model estimation, and to determine estimates of uncertainty. The method consists of splitting data into disjoint partitions. By training on some of these partitions, and testing on the data that was left out of the training partitions, we obtain several evaluations of the concerned model on an independent test data set. Different cross-validation schemes have different ways of splitting data. We used N-fold cross-validation.

Having several estimates of a quantity allows us to perform statistical analyses of various quantities. Relevant quantities include the performance of the model on new data, distributions of the coefficients of variables, and so on.

To find the best values of parameters in model estimation, we choose the values that give the highest average performance of the model when cross-validating.

5.4.4.1 N-fold cross-validation

In cross-validation with N folds, data is split into N disjoint partitions. By collecting $N-1$ of these partitions into one data set, and using this as training data, the last partition is available as an independent test data set. By switching the roles of the partitions, we obtain N evaluations of the concerned model on an independent test data set.

5.4.4.2 Choosing N

When setting the value of N, we must balance advantages and disadvantages. A higher value of N results in more estimates, which implies less variance on the means of the quantities of interest. On the other hand, higher values of N result in more computationally expensive cross-validation schemes. Common choices for N are five or ten.

5.4.4.3 Use of cross-validation in this project

We used cross-validation both to determine the best parameters in model estimation, and to obtain uncertainty estimates of estimated quantities. The use of cross-validation to determine the best parameters was left to the built-in routines of BCILab [16].

We used N equal to ten to obtain estimates of uncertainty of coefficient estimates and model performances. We used the matlab function *cvpartition* to get the ten partitions. By passing the vector of class assignments to *cvpartition* we obtained partitions with roughly the same proportions of classes.

5.4.5 Classification methods

In this section, we describe the classification algorithms that were compared. We chose these classification methods since they often perform well on diverse problems. Additionally, it is simple to use different misclassification costs for different observations with these method. For more complex models such as

Hierarchical kernel learning, Gaussian mixed models, and neural networks, it is difficult to use different misclassification costs for different observations. Also, most of the methods used here are easily interpreted, which is an advantage both to check the plausibility of models, and to gain a better understanding of the classification problem.

For each method, we briefly describe how observations are classified. Then we discuss the uncertainty of model estimates. Finally, we describe the implementation(s) used to test the method, and the computational difficulty of model estimation. We do not go into formal analyses of running times, but stick to giving an idea of the computational difficulties of model estimation.

5.4.5.1 Linear discriminant analysis

Linear discriminant analysis (LDA) is a linear classifier that assumes normally distributed data and common covariance matrix between classes. We limit ourselves to presenting LDA for binary classification, such that the number of classes K is equal to two. However, it is trivial to generalize the method to handle several classes directly without resorting to voting schemes [57].

LDA classifies an observation to the class that results in the highest likelihood of the observation. Through a number of algebraic manipulations, it can be shown that an observation $\boldsymbol{x}$ should be classified into class one if

$$\boldsymbol{x}^T \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - \frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma^{-1}\boldsymbol{\mu}_2 \geq 0.$$

Otherwise, class two is assigned. Details are given in appendix B.

Uncertainty on coefficient estimates Theory on analytical ways to estimate the uncertainty of coefficient estimates is sparse. We were able to find one paper [9] that analyzes LDA. However, this paper is very theoretical, and computationally easy estimates are not described. For the univariate case, we used the rule of error propagation, described in appendix E, to obtain simple estimates of uncertainty on coefficient estimates, as described in appendix B.

We found the uncertainty on the intercept and coefficient estimates to be

$$\frac{\bar{X}_1^2}{n_1 s^2} + \frac{\bar{X}_2^2}{n_2 s^2} + \frac{1}{2(n-1)s^4}\left(\bar{X}_1^2 - \bar{X}_2^2\right)^2$$

and

$$\frac{1}{n_1 s^2} + \frac{1}{n_2 s^2} + \frac{2(\bar{X}_2 - \bar{X}_1)^2}{(n-1)s^4},$$

respectively. We use s^2 to denote the sample variance, $\bar{X}_k$ the sample mean of class k, n_k the number of observations from class k, and n the total number of observations. The analytical uncertainty estimates are lower when the sample variance is higher, and also lower when the two population means are close. These observations seem counterintuitive since classification problems are usually easier when classes are more distinct. One way to explain the smaller uncertainty when data is lumped closer together is that the interesting part of data space, where classes should be separated, is well described by training data in such cases.

Implementation The LDA model is calculated from the means and covariance matrices of training data, so no iterative methods are necessary to estimate this model. Hence it is very fast to estimate the LDA model.

Three different implementations were used. Firstly, the implementation of LDA in the Stats toolbox was used. The Stats toolbox implementation handles several classes directly, without resorting to a voting scheme.

The two other implementations were used in the 1vR voting scheme. One implementation uses the standard formulation of LDA, as described above. This was implemented in the BCILab framework by the author. The other implementation was the default in BCILab, which is cast in a different formulation from the standard one. More details on the alternative formulation are given in appendix B.

All three implementations were used out of curiosity to compare the results, and see the amount of improvement gained by directly taking all classes into account. Also, we wished to verify that the two formulations of binary LDA would yield similar results.

5.4.5.2 Quadratic discriminant analysis

Quadratic discriminant analysis (QDA) is very similar to LDA and can also be generalized to handle several classes directly [57]. However, we restrict the presentation to the binary case.

The difference between LDA and QDA is that equal covariance matrices are not assumed in QDA. This keeps the quadratic terms of the logs of the likelihoods from cancelling each other out. The classification rule is then to classify an observation $\boldsymbol{x}$ into class one if

$$\frac{1}{2}\left[\boldsymbol{x}^T\left(\log\left(|\Sigma_2|^{-1/2}\right)\Sigma_2^{-1}-\log\left(|\Sigma_1|^{-1/2}\right)\Sigma_1^{-1}\right)\boldsymbol{x}\right]-\boldsymbol{x}^T\left(\Sigma_1^{-1}\boldsymbol{\mu}_1-\Sigma_2^{-1}\boldsymbol{\mu}_2\right)$$
$$+\frac{1}{2}\left[\boldsymbol{\mu}_2^T\log\left(|\Sigma_2|^{-1/2}\right)\Sigma_2^{-1}\boldsymbol{\mu}_2-\boldsymbol{\mu}_1^T\log\left(|\Sigma_1|^{-1/2}\right)\Sigma_1^{-1}\boldsymbol{\mu}_1\right]\geq 0.$$

Otherwise, class two is assigned. Details can be found in appendix C.

Uncertainty on coefficient estimates We were unable to find literature describing analytical estimates of coefficient estimates. Using the same principle as in appendix B, analytical estimates could be derived.

Implementation As in LDA, the QDA model is based on means and covariance matrices of training data, and no iterative method is needed. Thus it is quick to estimate the QDA model. More data is required to estimate the QDA model than the LDA model since a covariance matrix for each class is needed for QDA.

Two different implementations were used. Firstly, the implementation of QDA in the Stats toolbox was used. The implementation from the Stats toolbox handles several classes directly, without resorting to a voting scheme.

Secondly, the default implementation of QDA in BCILab, which is binary, was used. Treatment of several classes was obtained through 1vR voting.

Both implementations were used out of curiosity to compare performances between the 1vR setting and the setting when all classes are handled simultaneously.

5.4.5.3 Logistic regression

As was the case for LDA and QDA, logistic regression is easily generalizable to handle multiple classes at once. In this case, the classification method is referred to as multinomial regression (MNR). However, we only present the binary case.

Logistic regression is part of a larger class of models called generalized linear models, for which the theory is well known. The best suited model within this larger framework depends on the assumed distribution of data. In logistic regression the response variable $\mathcal{Y}$ is assumed to be Bernoulli distributed

$$\mathcal{Y}_i | X_{i,:} \sim Bernouilli(p_i), \quad \mathcal{Y} \in \{0, 1\}.$$

We wish to model the probability p_i as a function of the explanatory variables $X_{i,:}$ such that we can make predictions about an unseen response based on $X_{i,:}$. Since probabilities are restricted to lie in $[0, 1]$, the model of p_i must use a function with range $[0, 1]$. The logistic function $g(x) = \frac{e^x}{1+e^x}$ satisfies this. The model of p_i, where $p_i = \mathbb{P}(\mathcal{Y}_i = 1 | X_{i,:})$ is then

$$p_i(\boldsymbol{\beta}) = \frac{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^{m} \boldsymbol{\beta}_j X_{i,j})}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^{m} \boldsymbol{\beta}_j X_{i,j})}.$$

The estimation problem lies in finding the estimates of $\boldsymbol{\beta}_j, j = 0, 1, \ldots, m$. We do this by optimizing the likelihood-function. That is, we find the values of $\boldsymbol{\beta}_j, j = 0, 1, \ldots, m$ for which the observed data is most likely. The likelihood function is defined as the product of the probability densities of all observations (5.6).

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(\boldsymbol{y}_i | \boldsymbol{\beta}), \tag{5.6}$$

where $\boldsymbol{y}$ is the vector of observations. The likelihood function is

$$f(\boldsymbol{y}_i | \boldsymbol{\beta}) = \prod_{i=1}^{n} p_i(\boldsymbol{\beta})^{\boldsymbol{y}_i} (1 - p_i(\boldsymbol{\beta}))^{1-\boldsymbol{y}_i},$$

and the estimate of $\boldsymbol{\beta}$

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(\boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1 - \boldsymbol{y}_i) \log(1 - p_i(\boldsymbol{\beta})) \right). \tag{5.7}$$

We weight each observation by the inverse of the proportion of the class it is from, as described in subsection 5.4.3. More details on logistic regression are given in appendix D.

Forward selection of parameters It is likely that only some features are relevant in a classification problem. Only including some features also increases interpretability of a model and reduce risk of over fitting. In logistic regression, this can be done by starting with a model with no explanatory variables, and then adding those variables that seem to explain variability in data best. This is called forward selection [62, p. 221]. Two common alternatives are backward and stepwise selection. In backward selection, the starting point is a model that includes all variables. The variables that explain least variability are then removed iteratively. In stepwise selection, variables can both be added and removed in each iteration. Since these two alternatives are more computationally demanding, we chose to use forward selection.

Deviance is a measure of the degree to which a model agrees with data [58, p. 23]. When building a model using feature selection, we can use this measure to determine which features to include. The deviance of a large model minus the deviance of a smaller model, nested within the larger model, is used to test whether the simple model is significantly different from the larger model. Under the null hypothesis, the deviance approximately follows the χ^2-distribution with degrees of freedom equal to the difference in parameters between the larger and the simpler model [62, p. 299]. In forward selection, we attempt to add one variable at a time. This causes the deviance to follow the χ^2-distribution with one degree of freedom. Using the 5% significance level, we add the concerned variable if the test statistic is greater than $\chi^2_{0.05}(1) = 3.841$. This corresponds to a p-value less than 0.05, which means that the model with the concerned variable gives a significantly better fit to data than the model without that variable at the 5% significance level. The deviance is defined as (5.8) [58, p.23], although an alternative definition, given in appendix D, is sometimes used in the literature.

$$D_{abs}(y, \boldsymbol{\beta}) = -2 \log(f(\boldsymbol{y}_i | \boldsymbol{\beta})) \tag{5.8}$$

To test whether a model M_1 with coefficient estimates $\boldsymbol{\beta}_1$ nested within a larger model M_1 with coefficient estimates $\boldsymbol{\beta}_0$ significantly differs from M_0, we calculate $D_{abs}(y, \boldsymbol{\beta}_0) - D_{abs}(y, \boldsymbol{\beta}_1)$. If this is larger than the critical value 3.84, we have evidence that the larger model M_0 is a significantly better explanation of the observed data than M_1.

L1-regularized logistic regression Another way to limit the number of explanatory variables in a model is to penalize the cost likelihood function by the L1 norm of the coefficient vector. This also decreases the risk that coefficients of correlated explanatory features are estimated at high values in such a way that the effects of the variables cancel each other out in the model. We give a small example to make this scenario clearer. Assume x_1 and x_2 are highly correlated explanatory variables and that neither has any substantial influence on the response variable. Then a model in which x_1 has a very large coefficient $\boldsymbol{\beta}_1$ and the coefficient of x_2 is nearly the same with opposite sign $\boldsymbol{\beta}_2 \approx -\boldsymbol{\beta}_1$ will be nearly equivalent to a model with small $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$. The first model will give a wrong idea of the relation between explanatory variables and the response, might cause numerical instabilities if the coefficient estimates are of very high magnitudes, and high uncertainty will be associated with predictions. For these reasons a model with small or zero coefficients for x_1 and x_2 would be preferable.

Using the L1 norm of the coefficient vector as penalty, we maximize (5.7) plus the penalty. The L1 norm of a vector is the sum of absolute values of the values in the vector. Both terms are weighted to reflect the degree to which we want a good fit relative to our desire for a simple model. We also take weights of observations into account as shown in (5.9).

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \boldsymbol{w}_i \left(\lambda \left[\boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1 - \boldsymbol{y}_i) \log(1 - p_i(\boldsymbol{\beta})) \right] + (1 - \lambda) \sum_{j=1}^{m} |\boldsymbol{\beta}_j| \right), \tag{5.9}$$

where $\boldsymbol{w}$ is the vector of weights for observations. The parameter λ must lie between zero and one and determines the degree to which the coefficient vector should be simple relative to the ability of the model to fit data well. Higher values of λ cause the model to fit training data better at the cost of more non-zero estimates of the coefficients. The best value for λ can be found by cross-validation.

We use the optimization method steepest gradient descent [63] to estimate the

model parameters. For more details, see appendix D. Pseudo-code for this estimation is shown in algorithm 1.

Convexity The minimization problem in logistic regression is convex, both with and without L1-regularization. Convexity is desirable property because it ensures the uniqueness of a solution. Also, research into optimization of convex functions has a long history and easily implementable algorithms with linear and quadratic convergence are well known. We show the convexity of logistic regression in appendix D.

Uncertainty on coefficient estimates Since logistic regression is part of the large theory on generalized linear models, uncertainty on estimates in the model are also well known [58]. We use the Pearson estimate of estimate uncertainty, shown in (5.10)

$$\sigma^2_{Pearson} = \frac{\sum_{i=1}^{n} \boldsymbol{w}_i(\boldsymbol{y}_i - p_i(\boldsymbol{\beta}))^2/[p_i(\boldsymbol{\beta})(1-p_i(\boldsymbol{\beta}))]}{n-m-1}, \tag{5.10}$$

where m is the number of features, and thus the number of coefficients estimated. An extra degree of freedom is subtracted due to the intercept estimate. The Pearson estimate of variance approximately follows the χ^2 distribution, $\sigma^2_{Pearson} \sim \chi^2(n-m-1)$ [58]. The calculations in MatLab used to evaluate this estimate are given in appendix D.2.

Implementation An iterative algorithm called iterative re-weighted least squares [58, 62] efficiently estimates the logistic regression model. Hence logistic regression is more computationally intensive than LDA and QDA, but less so than decision trees and support vector machines.

Five different versions of logistic regression were used, and thus five different implementations.

First, we used the function *glmfit* in MatLab, which fits generalized linear models. Logistic regression is obtained by specifying the binomial distribution for observations. The 1v1 voting scheme was employed to handle multiple classes.

Second, we used *glmfit* as above, but limiting the feature set to those chosen with forward selection, implemented in the function *sequentialfs* from the Stats toolbox. Again, the 1v1 scheme was employed to handle multiple classes.

Algorithm 1 Numerical minimization for L1-regularized logistic regression to find coefficient estimates

$\boldsymbol{\beta} \leftarrow \mathcal{N}_{m+1}(\mathbf{0}_{m+1}, I_{m+1})$ {initialize coefficient estimates to values from the standard normal distribution}
max_eval $\leftarrow 100$ {stopping criterion; maximum number of iterations}
iter $\leftarrow 0$
while iter $\leq$ max_eval $\wedge$ not converged **do**
  $J \leftarrow$ gradient of non-penalized cost function, evaluated at $\boldsymbol{\beta}$
  $J\_penal_{i,j} \leftarrow$ gradient of penalty, evaluated at $\boldsymbol{\beta}$
  {execute non-penalized updates:}
  for $j = 0 \rightarrow m$ **do**
    $\boldsymbol{\beta}_j^{new\_temporary} \leftarrow \boldsymbol{\beta}_j - \mu \cdot \sum_{i=0}^{n-1} J_{i,j}$
  end for
  {execute penalized updates, which the intercept estimate is not affected by:}
  for $j = 1 \rightarrow m$ **do**
    $\boldsymbol{\beta}_j^{new\_penalized\_temporary} \leftarrow \boldsymbol{\beta}_j^{new\_temporary} - \mu \cdot \sum_{i=0}^{n-1} J\_penal_{i,j}$
  end for
  {set estimate to zero if zero was crossed in the penalty update}
  for $j = 1 \rightarrow m$ **do**
    if $\text{sgn}(\boldsymbol{\beta}_j^{new\_temporary}) \neq \text{sgn}(\boldsymbol{\beta}_j^{new\_penalized\_temporary})$ **then**
      $\boldsymbol{\beta}_j^{new} \leftarrow 0$
    else
      $\boldsymbol{\beta}_j^{new} \leftarrow \boldsymbol{\beta}_j^{new\_penalized\_temporary}$
    end if
  end for
  $\boldsymbol{\beta}_0^{new} \leftarrow \boldsymbol{\beta}_0^{new\_temporary}$
  if new estimates decrease cost **then**
    update estimates
    $\mu \leftarrow 1.5\mu$
  else
    $\mu \leftarrow 0.5\mu$
  end if
  iter $\leftarrow$ iter+1
end while

Third, we used the implementation of MNR in the Stats toolbox, *mnrfit*.

Fourth, we used *mnrfit* with forward selection. Forward selection was again carried out with *sequentialfs*.

Finally, we implemented a binary version of L1-regularized logistic regression. L1-regularized logistic regression for several classes was achieved through 1v1 voting with the implementation of the binary L1-regularized logistic regression. The parameter λ was found by cross-validation over the values $2^{-10}, 2^{-9}, \ldots, 2^{-1}, 1$.

5.4.5.4 Decision trees

Decision trees probably constitute the classification method whose results are most easily interpretable. Decision trees can both be used in the prediction of categorical and continuous response variables. When used for categorical response variables, the tree is usually referred to as a classification tree. We used the implementation of classification trees in MatLab. Since this implementation only considers binary decisions, our presentation of decision trees is limited to binary classification trees.

Informal description A new observation is classified by going down the tree, starting at the root node, until a leaf is encountered. Each leaf represents a class. An observation is then assigned to the class of the leaf that it ends up at.

All nodes that are not leaves will be referred to as internal nodes. Since we only consider binary trees, each internal node has two child nodes.

Each node is associated with a function involving one explanatory variable, whose output is boolean. If the function evaluates to true, the left branch is traversed to the left child node, and the left child node takes the place of the current node. Otherwise, the right branch is chosen. The final classification is obtained by iterating this traversal until a leaf node is reached. The observation is assigned to the class represented by that leaf node.

Construction of a classification tree Classification trees are constructed by determining the optimal binary function f_i at each node v_i, usually using a greedy algorithm that optimizes an objective function locally. If a greedy algorithm or other heuristic is not used, the problem becomes intractable since the number of possible trees grows exponentially with the number of variables.

Denote the set of training data by A. The construction algorithm then starts at the root node and finds the function f_0 that maximizes the gain Δ [55]. The function f_0 splits the set A into two sets A_1 and A_2 such that $A_1 \cup A_2 = A$ and $A_1 \cap A_2 = \emptyset$. Two child nodes of the root are then added to the tree, and the set A_1 passed down to child node v_1, while A_2 is passed to the other child node, v_2. At each of these nodes, the best split is found using only the training data that was passed down. At v_1, A_1 is split into two new sets, and these sets are passed down to the two child nodes of v_1, which are added to the tree. This is also done at v_2. The constant splits of data cause the individual problems at each node to diminish in size, while more problems must be solved at each level of the tree. This process continues until a pre-determined stopping criterion is reached.

The gain Δ is defined as

$$\Delta = I(\text{current node}) - \left(\frac{N(\text{child node 1})}{N(\text{current node})} I(\text{child node 1}) + \frac{N(\text{child node 2})}{N(\text{current node})} I(\text{child node 2}) \right),$$

where I is a measure of the impurity of a node. We use the Gini index $gini(f)$, described below, for the impurity measure. A lower impurity is achieved when a greater proportion of observations with the same label are sent to the same child node. The integer function $N(v_i)$ gives the total number of observations at node v_i.

The restriction on f to involve only one explanatory variable x and be binary amounts to restricting f to be of one of the forms

If x is categorical with values $a_1, a_2, \ldots, a_m$:

$$x == a_1,\ x == a_2,\ \ldots, \text{ or } x == a_m$$

with complement functions $x \neq a_1,\ x \neq a_2,\ \ldots, x \neq a_m$

If x is numerical with range $[a, b]$:

$$\text{for } a \leq c \leq b\text{: } x < c, x > c, x \leq c, \text{ or } x \geq c$$

with complement functions $x \geq x, x \leq c, x > c, x < c$.

Stopping criteria in tree construction Different criteria for when to stop the tree construction exist. We used default settings in *classregtree* in MatLab to obtain large trees. The default settings cause impure nodes to be split as long as ten or more observations in the training data are considered at the node. This

is the effective stopping rule since the default minimum number of observations per leaf node is one. These settings yield very large trees, which are most likely over fitted to training data. This makes it necessary to prune the large tree.

The pruning process collapses those nodes in the tree that cause the least increase in the sum of Gini indices over the remaining nodes. Larger trees provide better fits to training data, while smaller trees generalize to new data better. If trees are too small, though, they may not capture actual patterns in the analyzed problem [57, p. 308]. Thus it is necessary to find the best trade-off between training error and tree size. The built-in cross-validation procedure in *classregtree* was used to find the optimal pruning level, i.e. trade-off. The best subtree at this pruning level was then found using *classregtree* and used as the final classification tree.

The Gini index The Gini index is a measure of the impurity of a split, i.e. how well observations with different responses are separated. Gini indices closer to one indicate higher impurities, meaning that the split does not separate observations with different responses well. Thus we wish to minimize this objective function, $gini(f)$, where f denotes the function that gives the split and $gini(f)$ is the impurity measure of the split f. Let B_1 denote the set of observations from the training data that are put into one side of the split f, and B_2 the set of training observations that are put into the other side of the split. Also, let $\hat{\pi}_k$ denote the fraction of observations from class k in B_1. Then the Gini index [57, p. 309] is defined as

$$gini(f) = \sum_{k=1}^{K} \hat{\pi}_k(1 - \hat{\pi}_k).$$

The minimum value of the Gini index is zero, achieved when $\hat{\pi}_k$ is one or zero, and the maximum value is $1/4$, achieved when $\hat{\pi}_k = 1/2$.

Uncertainty on coefficient estimates Since decision trees are made from locally optimal decisions at each step, there is no guarantee that the solution is optimal. For the same reason, a small change in data may change the final decision tree substantially. For these reasons there is little sense in analyzing the importance of features, nor is it possible analytically. Since no coefficients are estimated, there is no sense in talking of uncertainty estimates either.

A way to gain an idea of the importance of features is to first split the training data into several subsets, as in cross-validation. One feature at a time could

then be left out, and test errors over each fold collected. These test errors would then give an idea of the effect on performance of each feature. This is computationally intensive though, and other more advanced uses of decision trees such as random forests, boosting, or bagging [57] would probably be better choices than optimizing features for the simple decision tree.

Implementation Decision trees are, in their most general formulation, computationally infeasible. With the restrictions described here imposed, they become computationally feasible. However, it is still a computationally intensive method.

The MatLab function *classregtree* was used with default settings except for a supplied cost matrix to construct large trees. The cost matrix reflects the class weighting described in subsection 5.4.3. The built-in cross-validation in *classregtree* was used to find the optimal pruning level, at which the large tree was then pruned to give the final tree.

5.4.5.5 Support vector machines

Support vector machines (SVM) are a generalization of optimal separating hyper planes (OSH). We present the binary cases of these methods. In classification through optimal separating hyper planes, the observations to be classified are assumed to be perfectly linearly separable. This means that the two classes can be separated by a linear function. SVMs, on the other hand, do not require the classes to be perfectly linearly separable.

For both OSHs and SVMs classes are assumed to have the labels -1 and 1. The classification of an observation $\boldsymbol{x}$ is then $\text{sgn}(\sum_{j=1}^{m} \boldsymbol{x}_j + \boldsymbol{\beta}_0)$. We describe OSH before explaining the differences between OSHs and SVMs that make SVMs preferable.

Optimal separating hyper planes Optimal separating hyper plane classification is performed by maximizing the minimal distance between points from each class. A decision boundary is defined between the two classes such that the distance from the decision boundary to the nearest observation from either class is equal. This distance is denoted by M. The width of the margin is then $2M$, and no observations lie within the margin. We scale the model parameters such that $y_i(\sum_{j=1}^{m} \boldsymbol{x}_{i,j} + \boldsymbol{\beta}_0) = 1$ for the observations $\boldsymbol{x}_i$ that exactly lie on the hyper planes that define the margin. The constraint that no observations lie

between the hyper planes $\sum_{j=1}^{m} \boldsymbol{x}_j + \boldsymbol{\beta}_0 = -1$ and $\sum_{j=1}^{m} \boldsymbol{x}_j + \boldsymbol{\beta}_0 = 1$ can then be expressed as

$$y_i(\sum_{j=1}^{m} \boldsymbol{x}_{i,j} + \boldsymbol{\beta}_0) \geq 1 \quad \forall i.$$

To minimize generalization error, the margin should be as wide as possible. The decision boundary $\sum_{j=1}^{m} \boldsymbol{x}_j + \boldsymbol{\beta}_0 = 0$ that maximizes $M = 1/\|\boldsymbol{\beta}\|_2$ is found by minimizing the objective function [55, p. 262]

$$\frac{1}{2}\|\boldsymbol{\beta}\|_2^2 - \sum_{i=1}^{n} \lambda_i(y_i(\boldsymbol{\beta}\boldsymbol{x}_i + \boldsymbol{\beta}_0) - 1),$$

where the constraint that no training observations can lie in the margin ($y_i(\boldsymbol{\beta}\boldsymbol{x}_i + \boldsymbol{\beta}_0) \geq 1$) is incorporated in the penalty term $\sum_{i=1}^{n} \lambda_i(y_i(\boldsymbol{\beta}\boldsymbol{x}_i + \boldsymbol{\beta}_0) - 1)$, and $\lambda_i \geq 0 \forall i$. A violation of the constraint results in a positive contribution from the penalty term such that the objective function can be minimized as long as the constraint is violated. The dual of this minimization problem is the quadratic programming problem of maximizing

$$\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j X_{i,:} X_{j,:}. \tag{5.11}$$

Quadratic programming problems are optimization problems in which the sum of a number of squared variables plus a sum of the same variables not squared, and a constant term are sought minimized or maximized subject to linear equality and inequality constraints. Since research concerning optimization of such problems is vast, it is desirable to pose optimization problems in this form.

In the transformation to the dual problem, the inequality constraints $y_i(\boldsymbol{\beta}\boldsymbol{x}_i + \boldsymbol{\beta}_0) \geq 1$ are substituted by

$$\begin{aligned} &\lambda_i \geq 0 \\ &\lambda_i \left[y_i(\boldsymbol{\beta} X_{i,:} + \boldsymbol{\beta}_0) - 1\right] = 0. \end{aligned}$$

Hence λ_i can only be non-zero for observations $\boldsymbol{x}$ that lie exactly on one of the edges of the margin, since $y_i(\boldsymbol{\beta} X_{i,:} + \boldsymbol{\beta}_0) = 1$ in this case. The observations on

these edges, i.e. those that have corresponding non-zero values of λ_i are called the support vectors. The support vectors and their corresponding λ_i define the model.

More details can be found in most textbooks on machine learning, for example [55, 6, 57].

Generalization to support vector machines By allowing some observations to lie within the margin, the assumption of perfect separability may be relaxed. This is exactly what is done to obtain SVMs from OSHs. The distances of the misclassified observations, i.e. those within the margin, are constrained by specifying a penalty C on each misclassified observation. This leads to the same quadratic programming problem as above in (5.11), except for the constraints on λ_i, which are $0 \leq \lambda_i \leq C \forall i$.

The kernel trick [57] is often used with SVMs since the kernel trick enables the construction of decision boundaries in infinite space. This typically allows for better separability of classes. Here, we used radial basis function kernel, $\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = exp(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2/(2\sigma^2))$. C and σ are chosen through cross-validation.

Uncertainty on coefficient estimates The SVM model becomes more difficult to interpret when the kernel trick is used since the model is expressed in terms of the kernel functions. This makes it difficult to derive the effects of the different variables on class assignment. Due to these complexities, we have deemed investigations of feature importance for SVMs to be out of the scope of this project.

Implementation Support vector machines constitute the most computationally intensive classification method tested here. SVMs are computationally difficult even if the parameters C and σ have been determined beforehand. Since they must be found by cross-validation, SVMs by far outrank the other methods in computational difficulty of model estimation.

The SVM implementation that we used in this project was the variant from BCILab called *svmlight*. This implementation is meant for binary problems. To handle multiple classes, 1v1 voting is used.

To determine the values of C and σ, we performed a cross-validation over the values $10^{-2}, 10^{-1}, \ldots, 10^{5}$ for C and $10^{-4}, 10^{-1}, \ldots, 10^{3}$ for σ, since values in

these ranges have previously achieved a good performance in the IC classification task [56].

5.4.5.6 ADJUST

The algorithm ADJUST attempts to detect ICs that fall into one of a number of classes of artifacts. This is accomplished by finding appropriate thresholds of certain features using Gaussian mixed models, based on data. The ICs that exceed these thresholds for certain features are classified as one of the possible types of artifacts, depending on which thresholds were exceeded.

The features used in ADJUST were also used here. These are features 1 to 11.

More details on ADJUST can be found in [41].

Uncertainty on coefficient estimates ADJUST does not use any sort of feature coefficients.

Implementation ADJUST runs quite fast, with the feature calculations taking the longest time to compute. Compared to the other methods we used, ADJUST is faster than all since the feature calculations are faster.

The code for the ADJUST plug-in to EEGLab was used, albeit the implementations of features were altered slightly as described in section 5.3.

5.4.5.7 BBCI

The BBCI algorithm aims to distinguish neural from artifactual components. It uses SVM to classify ICs based on a number of features. All but one of these were also used for the project described in this report. The features that are used in BBCI are features 12, 13, 14, 15, 18,19, 20, and 21. Additionally, one feature, the "current density norm", is used in the BBCI algorithm, but not here. Details on this feature, and why we could not incorporate it, are given in appendix H.

To find the best parameters for SVM, cross-validation was used. This search was performed over powers of 10, seeming to range from -3 to 4. The exact search range is not mentioned in the paper describing the algorithm [56], but

all values reported as being best lie in this range. In our implementation, we searched over the same values as in our use of SVM, namely $10^{-2}, 10^{-1}, \ldots, 10^{5}$ for C and $10^{-4}, 10^{-1}, \ldots, 10^{3}$ for σ.

More details, in particular explanations of features, are given in [56].

Uncertainty on coefficient estimates BBCI is based on an SVM classifier, making the remarks for SVM apply here as well.

Implementation Since BBCI uses SVM to make classifications, the running time for BBCI is high, just as was the case for SVM.

Our implementation of the method does not include all the features that are described in [56]. There is one feature suggested in the BBCI paper that is not used in out implementation of the proposed algorithm. This feature is the current density norm, which requires an unrestricted dipole fit to be meaningful. If the dipole fit is restricted to contain only a certain number of dipoles, then the current density norm will be almost identical for all fits. The dipole fit in EEGLab has exactly this restriction, making this feature irrelevant. The current density norm is described in appendix H.

5.4.6 Qualitative comparison of classification methods

The assumptions underlying models, the estimation of models, and the forms all come together to result in different advantages and disadvantages of each model. We now describe such advantages and disadvantages of the models used in this project.

Interpretability Interpretability of models is advantageous for a couple of reasons. Firstly, an easily interpretable model affords insight into the problem by showing which parts of an observation are most important for classification. Secondly, it is possible to check that the model makes intuitive sense based on preliminary knowledge of the classification problem. This is a down-to-earth guard against over fitting.

Decision trees constitute the model that is easiest to interpret since the visualization of decision trees is very intuitive. At each node, it is clear why an observation is sent down a branch, ending up at the final classification.

In LDA, QDA, and logistic regression (in all its variants), it is also easy to see the effect of changes in feature values since these methods are based on linear (and quadratic for QDA) combinations of features.

On the other hand, SVMs obscure the effect of features since the classification is based on complicated functions of the features.

Model estimation The difficulty of estimating a model based on data is also different for different methods. For some models, the estimation method results in unique solutions that do not change much when data is changed to a small degree, which is desirable. The speed by which these solutions are found also differs between models.

LDA and QDA for example are easy to estimate since this estimation solely consists of estimating covariance matrices, and then calculating matrix products with the mean vectors. Hence the estimates of these models are unique for each set of data.

In logistic regression, the model is found by numerical minimization of the negative log-likelihood. This is a convex problem, which means that the solution is unique. Also, several algorithms of varying complexity and speed exist to solve convex problems. All such algorithms require several iterations, obviously implying that it is slower to estimate the logistic regression model than it is to find the LDA and QDA models. When regularized logistic regression is used, the parameter that controls the degree of regularization must be determined. Cross-validation is often used for this purpose.

The cost function that must be minimized in SVM is also convex. In SVM, two parameters control the model estimation. To find the best values of these, nested cross-validation is often used. This of course slows down the model estimation.

Decision trees are constructed based on a number of locally optimal decisions. Since a small change in data may cause a change in one of those locally optimal decision, and thus affect all future decisions, a small change in data can lead to a large change in the final tree.

Assumptions All of logistic regression, LDA, and QDA use the ratio of the probability that an observation $\boldsymbol{x}$ is one class versus the probability that an observation is in the other class [62, p. 358]

$$\frac{P(\mathcal{Y}=1|\boldsymbol{x})}{P(\mathcal{Y}=0|\boldsymbol{x})}.$$

The difference between the methods lies in the assumptions. LDA and QDA assume that both populations are normal, and LDA further assumes equal covariances. Logistic regression makes no such distributional assumptions. When assumptions of normality are fulfilled, LDA and QDA are likely to obtain better performance than logistic regression since they exploit the assumptions. Often, however, it may be difficult to ascertain whether assumptions are valid. Hence logistic regression is often a better choice.

Neither support vector machines nor decision trees make any assumptions on the distribution of data.

5.4.7 Determining the best model

We split the entire data set into ten different partitions to obtain several estimates of performance for each model, as well as several coefficient estimates for each model. By training on nine of the ten partitions, and testing on the remaining tenth, we got ten estimates of the performance of each model. We made sure to use the same partitions for each model. This enables direct comparisons of the models since the training and test data sets are the same for each model.

During training of the models, we minimized a misclassification measure related to the misclassification rate as described in 5.4.3. Since we wish to use the model that is best over all classes, and not just able to detect ICs from the largest class, we continue to use the same measure in this final comparison of models. Hence we assign higher ranks to models with lower misclassification measures, defining the misclassification measure mcm as

$$mcm = \sum_{k=1}^{K} \frac{n}{n_k} mcr_k,$$

where K is the number of classes and mcr_k is the relative misclassification rate for class k. The relative misclassification rate for class k is one minus the proportion of correctly classified samples from class k.

5.4.7.1 Comparing two models

To test whether two models are significantly different, we performed the sign test, which is the non-parametric equivalent of a paired t-test, on the ten misclassification measures. The sign test is performed by first finding the differences between misclassification measures of the two models for each test data set, resulting in ten differences. The number of negative differences is then counted. This number is compared to the binomial distribution to test whether the data is in accordance with the hypothesis that the two models show similar performance. Formally, the sign-test tests whether the medians of two populations are significantly different. Let $\mathcal{X}$ be a random variable denoting the number of negative differences. Under the null hypothesis that the two models show similar performance, and thus have the same median, each difference has equal probability of being positive or negative. Hence the number of negative differences is distributed according to the binomial distribution with ten trials and probability of success 0.5, $\mathcal{X} \sim Bin(10, 0.5)$. We then wish to find the probability of observing the observed data, or a more extreme outcome, under the null hypothesis. Assume the number of negative differences x is less than or equal to five, the median of the null-distribution of differences. We then find (5.12)

$$p = \mathbb{P}(\mathcal{X} \leq x). \tag{5.12}$$

If x is not less than or equal to five, we instead find $p = \mathbb{P}(\mathcal{X} \leq 10-x)$ to find the probability of the observed, or more extreme data. Since we wish to assess the difference between the two models, we are also interested in the other extreme outcome, namely that there are more negative differences than expected. Since the binomial distribution is symmetrical, we find this probability by multiplying p by two. Thus the final p-value is $2p$.

We denote the total number of models compared by $nmodels$. The number of binary comparisons is then $\sum_{i=1}^{nmodels-1} i) = \frac{nmodels(nmodels-1)}{2}$. Two models are said to perform equally well if $2p$ is greater than

$$\alpha_{bonf} = 2\frac{\alpha}{nmodels(nmodels-1)}$$

The significance level α_{bonf} is the Bonferroni correction to obtain an overall significance level of α, which we set to 0.05.

5.4.7.2 Determining the best model

We are interested in ranking the models from best to worst. For this to make sense, it is necessary that the performance of models is transitive. That is, if model A is better than model B, is better than model C, this should imply that model A is better than model C.

First, we compare each pair of models by counting the number of cross-validation folds in which one model outperforms the other. The model that outperforms the other the most often is ranked as the best of the two. These binary rankings can then be put together to obtain a ranking of all the models from best to worst. By performing the binary rankings first, we can use the sign-test to determine whether differences in performance are statistically significant. The final ranking of all models will then include indications of whether the performance of two adjacent models in the ranking is statistically significant or not.

We check whether a ranking of models is transitive by drawing a directed graph in Maple with edges going from worse models to better models. If the directed graph is acyclic, the ranking of the models is transitive. Otherwise, it is not. When the ranking containing all models is not transitive, models cannot be sorted in order of performance. For the classification settings in which rankings turn out to be transitive, we give these rankings. Otherwise, we do not sort the models.

We also use box plots of the misclassification measures for each model in each classification scenario to assess classification performance. Additionally, we will look at the mean of the misclassification measure and the standard deviation of this for each model over the ten cross-validation folds.

Chapter 6

Seizure detection

The pipeline to construct a seizure detection model is shown in figure 6.1. Figure 6.1 also shows the relation of seizure detection to the rest of the project.

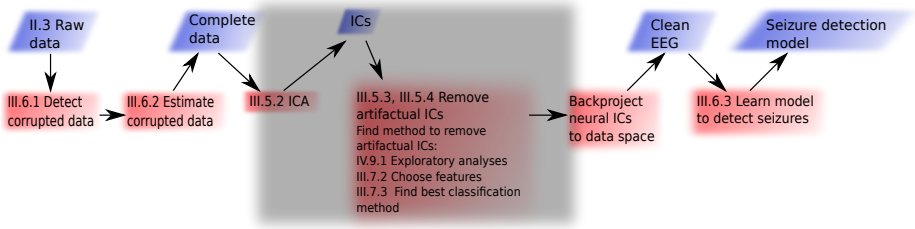

Figure 6.1: Flow of data to obtain a seizure detection model. The parts that are not grayed out are the ones that data are passed through. The grayed out box represents the part of the project that aims to construct an automatic classifier of ICs.

In the following sections, we describe how we detect erroneous values of EEG recordings, how we estimate them, and finally the method that we use to train a seizure detection model.

6.1 Detection of corrupted data

To detect corrupted recordings due to loose electrodes, we make use of the fact that at most a few electrodes are usually loose simultaneously. We split the data into intervals of one second. The range of values in each channel is found for each second. The mean range and standard deviation of the ranges over the channels are then found in each second. If the range of a channel deviates by more than two standard deviations from the mean range, we judge the data in that second for that channel to be corrupted.

6.2 Estimation of corrupted data

Missing data is a common problem in data analysis that arises within almost all applications. In the field of EEG, loose electrodes cause missing values in the recorded data. The HypoSafe data sets contain many intervals, about 80% of the total recorded time, during which one or more electrodes are missing. Missing data is problematic since few analysis techniques are able to handle missing values. We describe ways to handle missing data. However, a profound analysis of missing data approaches was out of the scope of this project. We state a few naive approaches before describing our solution to the problem. A good overview of approaches to handling missing data is given in [39].

The values referred to as missing in the HypoSafe data are, in reality, not missing, but extreme to an extent that indicates a loose electrode. Since the electrodes continuously record the EEG, values for all time points do exist. We will refer to values clearly recorded from a loose electrode as missing, since the desired values, i.e. the electrical activity on the scalp, is in fact missing in these cases. Intervals of loose electrodes are detected as described above.

Remove intervals of missing values The simplest way to handle missing values is to remove all intervals in which one or more channels have missing values. This would ensure that only properly recorded data is used in further analyses. If entire intervals are removed, the data from all the attached electrodes is also lost however. Thus a lot of useful data is lost in this process. Since a large part of the recordings have one or more loose electrodes, this is not a viable solution. This method is discussed in more detailed in [39], where it is referred to as complete-case analysis.

Set missing values to a numerical constant Another approach is to set all missing values to a certain constant, for example zero, or the mean of other observations of the same variable. This would avoid the problem of extremely large values unbalancing the analysis. However, any relation between the values in each channel, and the event of interest, will be obscured by this approach.

Factorization of available data to estimate missing data The signals recorded at the different electrodes are actually different linear combinations of the same brain activity. This spatial mixing of electrical signals implies that EEG recordings from different electrodes are highly correlated. Assuming that the linear mixing remains constant, it makes sense to express the value at each electrode as a linear combination of the values at the other electrodes. The assumption of non-changing linear mixing processes is sensible since the brain tissue determines the linear mixing, and the brain tissue does not change. We will estimate the missing values, and then fill the missing values with the estimates.

Due to the linear relations among channels, we model the data as a product of two matrices.

$$Y = AS.$$

The matrices A and S that give the closest values to the non-missing data that represent the best model of relations between electrodes. This model is conditional on observed data and multivariate, preserving associations amongst the channels, as recommended in [39, p. 72].

The challenge is then to find A and S such that $\|A \times S - Y\|_2$ is minimized. We take missing values into account by introducing the matrix W, whose $(i,j)^{th}$ element is zero if the $y_{i,j}$ is missing, and one otherwise. We disregard missing values while finding A and S by minimizing

$$\sum_{i,j} W_{i,j}(Y_{i,j} - \sum_{h} A_{i,h}S_{h,j})^2. \tag{6.1}$$

We optimize using gradient descent, alternately updating A and S. Details are given in appendix F.

6.3 Model

Common spatial patterns (CSPs) are spatial patterns that distinguish between EEG from two different conditions [36]. CSPs are those spatial patterns across the scalp that are common between the two conditions, but explain most variance in one condition, and least in the other. CSPs can then be said to maximize the proportion of explained variance in data in one condition relative to the other. In this way, the spatial patterns that are most discriminative between the two conditions are likely to be found. Since we suspect brain activity to differ from normal EEG during seizures this method seems to be a tool likely to identify seizures. Indeed, good performance in discriminating between normal subjects and subjects with a brain disorder is well documented in [36].

An $m \times M$ matrix W^T for m channels and M spatial patterns is sought, such that

$$S = WY,$$

where Y is the data matrix and S the activation of each spatial pattern over time. W can be found by first solving the generalized eigenvalue problem

$$Cov(Y|k = 1)\boldsymbol{w} = \lambda Cov(Y|k = 2)\boldsymbol{w},$$

where $Cov(Y|k = i)$ is the covariance of observations from population i, and $\boldsymbol{w}$ the generalized eigen vector that solves the problem. The columns of W consists of such $\boldsymbol{w}$. We can then construct two diagonal matrices Λ_1 and Λ_2, one for each class, such that the j^{th} element of Λ_i is the variance of the j^{th} pattern for the i^{th} class

$$\Lambda_i = WCov(Y|k = i)W.$$

We let $\lambda_i^{(j)}$ denote the j^{th} element of Λ_i. By scaling the columns of W, we can satisfy the equality $\lambda_1^{(j)} + \lambda_2^{(j)} = 1$ for all j. Clearly, then, a high variance in one condition implies a low variance in the other condition [48, pp. 341-342]. Classification on the transformed data is then performed by a classifier, where a simple classifier such as LDA is a common choice.

CSPs are well suited to interpretation since the identified patterns are precisely those that differ most between the two conditions. This makes it easy to relate

the results to the disorder or other difference between conditions being studied. The brain areas that show most differing activity between conditions may then be studied further, either to see if they may be the cause of the difference between conditions, or to investigate how these areas are effected by each condition.

Implementation We used the implementation of CSP in BCILab, which is described in detail in [7]. In this implementation, the data is first band-pass filtered with a lower limit of 7Hz and an upper limit of 30Hz. Thus only data with frequencies between 7Hz and 30Hz is used. Also, the logarithms of the spatial activations ($\boldsymbol{w}^T Cov(Y)\boldsymbol{w}$) are passed to the classifier instead of the raw spatial activations. The six eigen vectors that maximize the difference in variance between the two conditions are chosen to constitute W [7]. The patterns that are found to be best are passed as features to LDA for classification.

Each second in the HypoSafe data was labeled as being part of a normal period of EEG, or a seizure. We trained and tested the CSP model by using the brain activity at each label and a quarter of a second to each side as one labeled segment.

Part IV

Results and discussion

Chapter 7

Artifact removal

7.1 Exploratory data analysis

Exploratory analyses give a preliminary idea of the amount of information present in data. The final results can be compared to the preliminary results from exploratory analyses as a simple check that the final results are sensible.

In addition to serving as a check on the final results, exploratory analyses are useful for understanding a problem. Relations between features and classes can be extracted and visualized to gain an idea of the degree to which each feature is informative of class membership. Additionally, visualization of relations between features makes it possible to detect redundant features.

As described in part II, two studies with data sets containing labeled ICs were analyzed in this project. To prevent differences between the two studies from obscuring the visual analyses, we performed all exploratory analyses separately for each study.

Relations between features only depend on the explanatory variables, and not on the class assignments of observations. Hence we performed exploratory analyses of relations between features in the two cases of including all, or only labeled ICs.

Conversely, relations between features and classes do depend on the classes included for classification. Thus exploratory analyses of relations between features and targets were performed in all the four scenarios described in subsection 5.4.1.

We will now give an overview of the exploratory analyses that we performed. Next, the results from the analyses concerning relations between features are given. Then we go on to describe the results from analyses concerning relations between features and classes. Finally, we summarize these preliminary results.

7.1.1 Analyses

The details of principal component analyses and mutual information are given in section 5.2 and subsection 5.3.4. The focus here is on how we used these methods for exploratory analyses.

7.1.1.1 Principal component analysis

Principal component analysis (PCA) finds the linear combinations (principal components) of features that explain the most variance in data. PCA is explained in detail in section 5.2. These principal components can be viewed as new explanatory variables such that the first new explanatory variable explains most variance, the second variable second most variance, and so on. We plot the values of each of the first three principal components (PCs) for each observation in a 3D coordinate system. By coloring observations from each class with a specific color, this reveals whether the classes are easily linearly separable. A sign that classes are easily linearly separable is that clouds of each color appear separately from the rest in the 3D plot.

We also plot the fraction of variance explained by each PC. This allows us to see by how much different linear combinations of original features differ in their ability to explain data.

To investigate which features are most important in distinguishing between classes, we plot the weight of each original feature in the first three PCs.

7.1.1.2 Correlation

It is also interesting to see whether some of features actually describe the same quantity. This was done by investigating the correlation between each pair of features.

7.1.1.3 Mutual information

To capture any non-linear relations between features, we also looked at the mutual information between pairs of features. Finally, mutual information between each feature and the class assignment was plotted. High values of mutual information between a feature and classes implies that the feature is a good predictor of class label.

7.1.2 Relations between features and classes

First we present exploratory analyses when all classes of ICs are distinguished between, i.e. the multiclass scenario. Next, we show results from exploratory analyses in the binary setting when only the classes neural and non-neural were analyzed.

7.1.2.1 Multiclass classification

Principal component analysis Figure 7.1 shows the projections of feature vectors of ICs onto the first three principal components of the feature matrix. Face-on views of these plots, which show the projections onto just two of the components at a time, are given in appendix A in figure A.1.

Although a cloud of mostly unidentified observations seems to be distinguishable from neural components in the top right of figure 7.1, a lot of other points are hidden behind this black cloud. Since there are many unidentified observations, and the class of unlabeled ICs is added to the plot last, the illusion of separable clouds appears. Also, the light blue cloud of neural components is mixed with all the other types of components. The overlap of unlabeled ICs with the other classes is more apparent in the plot for the cue data set in which there are fewer observations. The plot including all ICs from the cue data set is shown in the top left of figure 7.1.

Projections of feature vectors from labeled ICs onto the first three PCs are shown in the bottom row of figure 7.1. The plots from both data sets show clouds of neural and muscle components with different centers, although the clouds overlap to a high degree. For the emotion data, a separate cloud of heart components can be seen below the majority of the points. Also, lateral eye movements are somewhat separated from the other classes. Thus these plots hint that it will be possible to separate at least the components that were manually labeled, although a linear separation might be difficult to achieve.

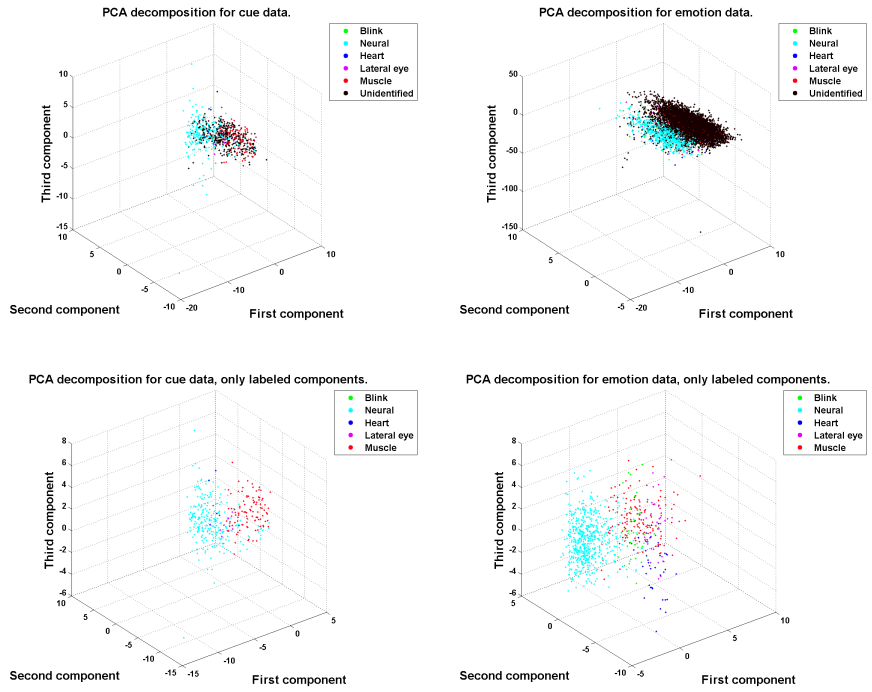

Figure 7.1: Projections of feature vectors of ICs onto first three PCs. The projections of the feature vectors are colored according to the class they are from. The top row shows projections of observations when the unlabeled ICs are included, and the bottom row shows the results when only labeled ICs are plotted. The left column shows ICs from the cue data set, and the right column shows ICs from the emotion data set, both of which are described in part II.

Mutual information Mutual information between each feature and the class labels is shown in figure 7.2 for both data sets, both including and not including the unlabeled ICs.

The last 12 features are measures that aim to detect the similarity in the time series to that of the ECG signal. It is somewhat odd that these have substantially higher MI with classes in the cue data set than in the emotion data set.

Feature 18, which is the fit error of the fitted power curve to the typical power curve from EEG, also has high MI with classes.

The first and 14th features also show high values of MI across all plots. The first feature is a measure of the discontinuity of the distribution of activity on the scalp map, while the 14th is the logarithm of the mean power in the α band. We expect the logarithm of the mean power in the α band to be high for neural components. Hence both these features should aid identification of neural components.

The logarithm of mean power in the θ band (feature 23) also has high MI in all plots. Since the θ band consists of low frequencies, and most artifacts have high frequencies, this is yet another feature that is likely to help detect neural ICs.

The consistency of the patterns over all four cases indicates that the results are not random, but reflect real structures in data.

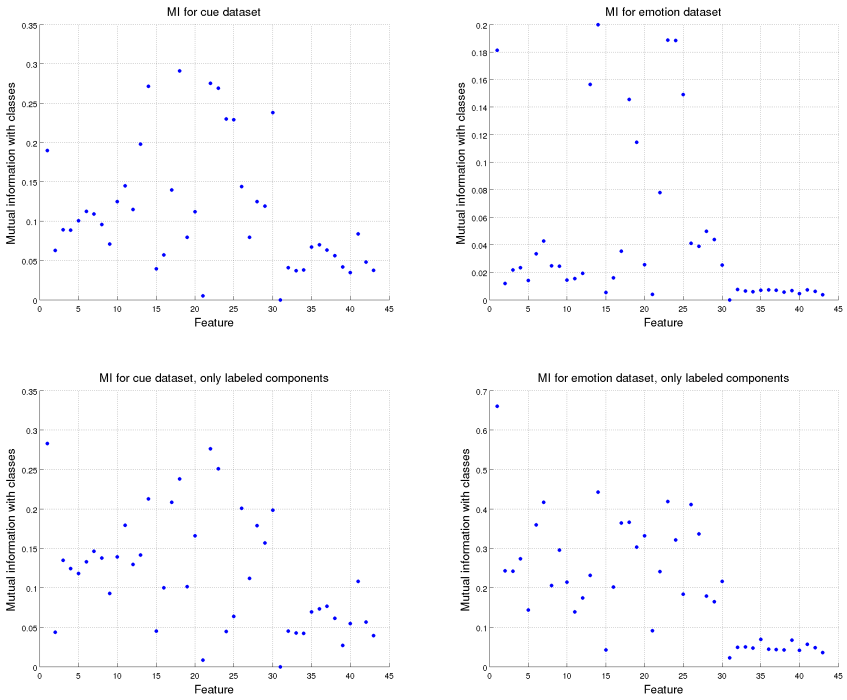

Figure 7.2: Mutual information between each feature and class assignment into eye blink, neural, heart, lateral eye movement, muscle, or mixed IC. The class of mixed ICs was only included in the top row. The left column holds results for the cue data set, and the right column for the emotion data set. The plots in the top row were obtained using all components, while the bottom row contains plots calculated solely from manually labeled components.

7.1.2.2 Binary classification

Principal component analysis Figure 7.3 shows the projections of components onto the first three principal components, color coded to indicate class membership. Easy linear separation seems achievable in the emotion data set, in both the case including unlabeled ICs, and in the case that only includes labeled ICs (top and bottom plot of right column in figure 7.3, respectively). However, linear separation does not seem possible for either case in the cue data set since the clouds of colored points overlap to a large extent. This might be due to the much smaller number of components in the cue data set, since enough information might not be available to obtain appropriate principal components.

Mutual information Figure 7.4 shows the mutual information between each feature and the class assignment, for each data set and for both cases of either including or not including unlabeled ICs.

As in the multiclass case, feature one, the measure of generic discontinuity from ADJUST, has a high MI with classes in all four cases in figure 7.4.

Feature 14, the logarithm of the mean power in the α band, also has high MI in all cases, again in correspondence with the multiclass case.

In general, the patterns in the plots for the binary case are similar to the patterns in the multiclass case. This similarity may indicate that the difficulty in solving the binary and multiclass problems are similar.

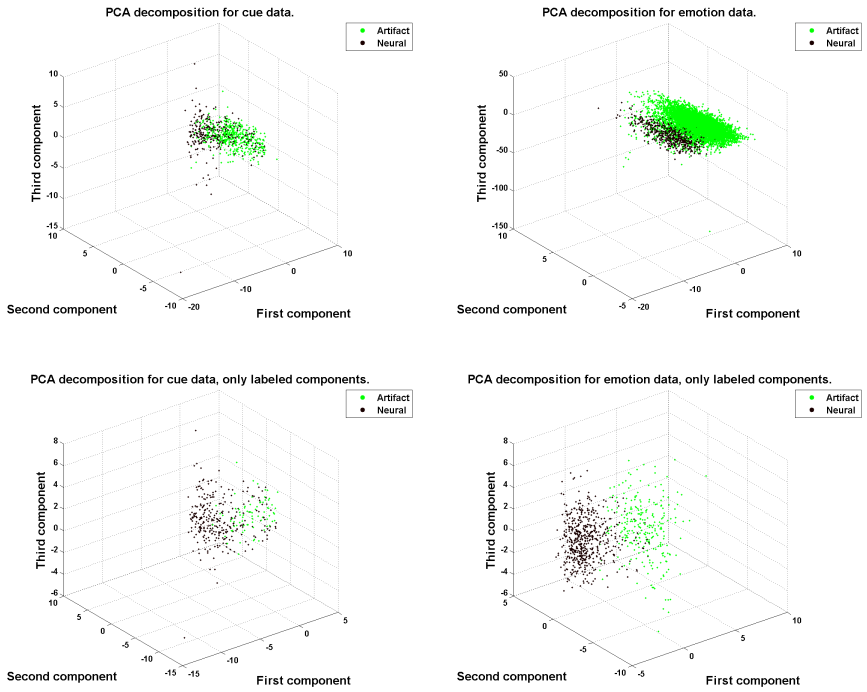

Figure 7.3: This plot shows the projections of feature vectors of ICs onto the three principal components of the feature matrix. The colors of observations indicate which class they are from. Only the ICs that were manually labeled as neural components are assigned to the class of neural ICs. All other ICs are assigned to the class of artifactual components. The top row shows projections of all ICs, including those without labels. The bottom row only includes projections of ICs that were given manual labels. The left column shows projections of ICs from the cue data set, while the right column shows projections of ICs from the emotion data set.

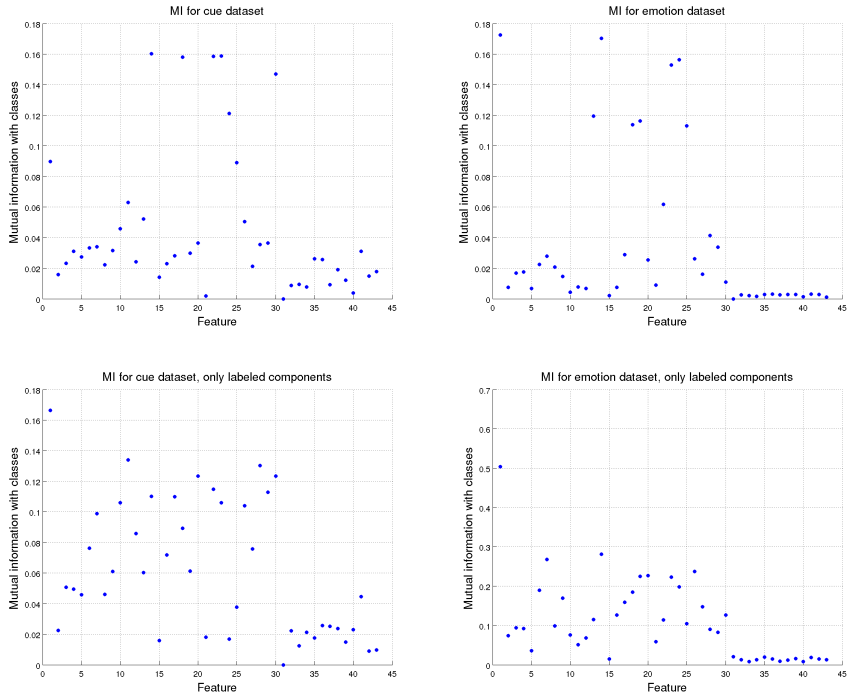

Figure 7.4: Mutual information between each feature and class assignment as neural or artifactual. The left column holds results for the cue data set, and the right column for the emotion data set. The plots in the top row were obtained using all components, while the bottom row contains plots calculated solely from manually labeled components.

7.1.3 Relations between features

Variance explained Figure 7.5 shows the fraction of variance explained by each principal component. The figure contains plots for both data sets, both including and excluding the independent components that were not manually labeled. As is often the case, the last principal components account for nearly none of the variance in the feature vectors. That the first PCs explain a large part of the variance implies that the dimension of the feature space is smaller than the number of features. This is true for both data sets, both when non-unidentified components are included and when they are not.

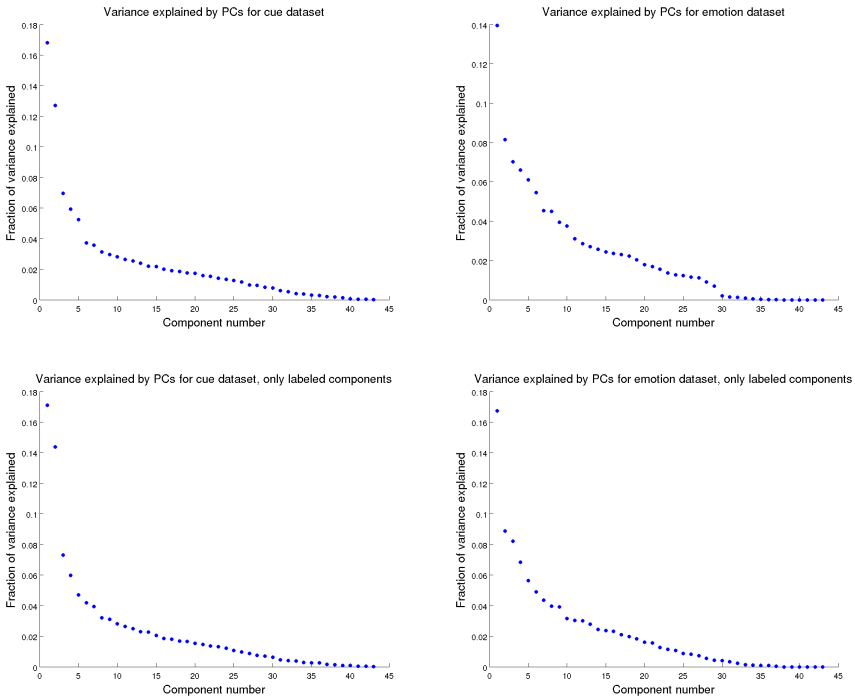

Figure 7.5: This graph shows the fraction of variance accounted for by each principal component of the feature values. The top row holds the fraction of variance explained by the PCs when all ICs were included. The bottom row shows the fraction of variance explained by PCs when only features from manually labeled ICs were included. The left column is based on the cue data set, while the right column is calculated from the emotion data set.

Loadings of PCs The contributions of features to each of the first three PCs are shown in figure 7.6. The loading is the term given to the weight of a feature in a principal component. By taking the absolute values of these, we can visualize the degree of contribution to each principal component by each feature. Some patterns are repeated in all the four plots.

Feature one, which is the measure of generic discontinuity of the scalp map, contributes a lot to one of the first PCs in both the cue and emotion data sets, both when the unlabeled ICs are included and when they are not.

Also, features 23 and 24, which are the logarithms of mean power in the θ and β bands, are given high weights in the first PC in all cases.

Feature 14, the logarithm of the mean power in the α band, is also weighted highly consistently in the first principal component.

The second and third principal components do not exhibit such patterns consistently across all four plots.

The ECG detection features for ICs from the emotion data set, are all weighted highly in either the second or the third PCs. The reason that such heart PCs were not found in the cue data set may be that there were not enough examples of heart ICs in the cue data.

Feature 17, the z-coordinate of the dipole fit, also has a high loading in one of the first PCs in all plots.

Mutual information Figure 7.7 shows the mutual information between each pair of features calculated from the emotion and cue data sets, both with and without non-labeled components. The mutual information between a feature and itself is of course high, implying that the highest values can be found in the diagonal. Most pairs of features have little mutual information, as evidenced by the mainly blue color in the four plots, which signifies the value 0. We discuss the pairs of features that have non-zero MI here.

Some patterns are repeated on all the four plots. Feature 14, for example, has non-zero MI with features 22-24. Features 14, 22, 23 and, 24 are the logs of the mean power in the α-, δ-, θ, and β-bands, respectively. Since the α-, δ-, and θ-bands are all low frequent, it is to be expected that non-zero activity in one of these bands is related to non-zero activity in the other bands.

Furthermore, feature eight has non-zero MI with features three, four, and five.

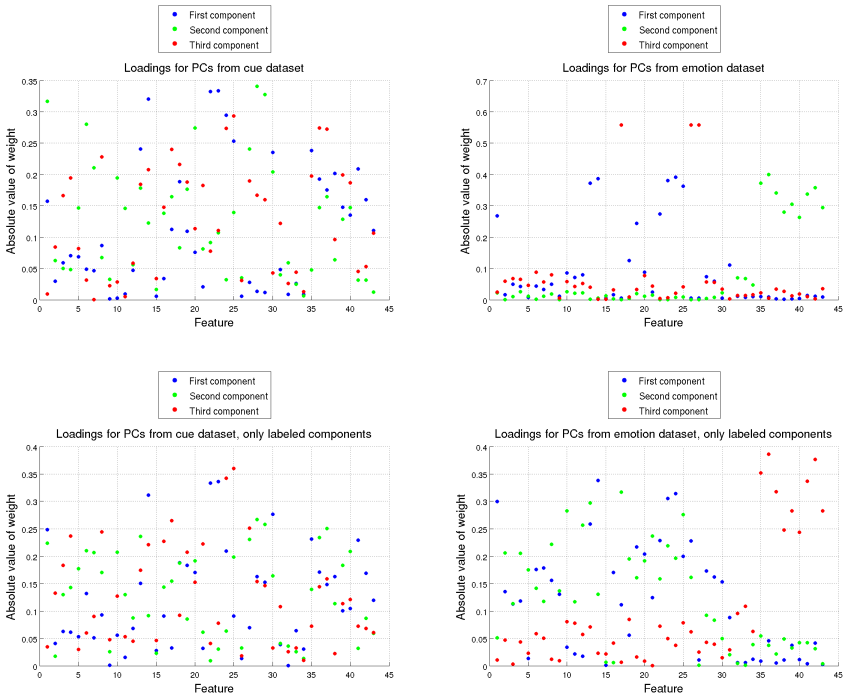

Figure 7.6: Absolute values of loadings of the first three principal components. The top row shows the loadings for the PCs of the feature matrices when all ICs were included, and the bottom row shows the loadings of PCs when only labeled ICs were included. The left column shows the results for the cue data set, while the right concerns the emotion data set.

Feature eight is the mean weight of the frontal electrodes. Features three and four are the mean of the weights of sensors on the left and the right, respectively, and feature five is the spatial average difference of frontal and posterior electrodes. The mutual information for these three pairs ((feature eight, feature three), (feature eight, feature four), and (feature eight, feature five)) are highest in the two plots based only on labeled components.

Other pairs of features that have non-zero MI in all plots are (22, 23), (24, 25), and (28, 29). Non-zero MI for these pairs is to be expected since features 22-25 are the logarithms of the mean power in the bands δ, θ, β, and γ, respectively. Features 28 and 29 are the first and second estimates of the Hurst exponent from *wfbmesti*.

Additionally, the last twelve features have non-zero pairwise MI for several pairs. Since the last twelve features are all aimed at detection of the ECG, this is to be expected.

These patterns are the same in all plots, which is to be expected since the features, and hence the relations between them, are the same in all cases.

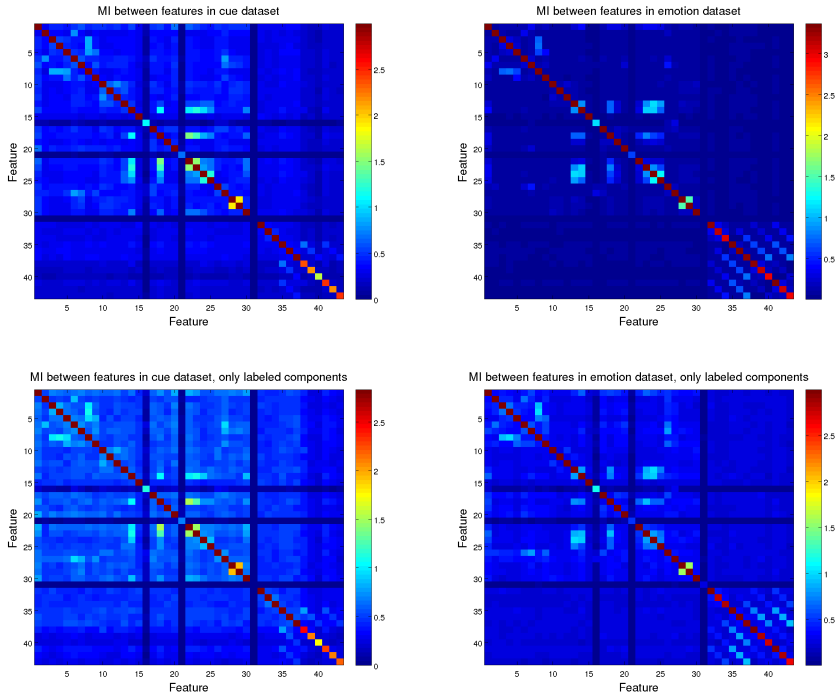

Figure 7.7: Mutual information between each pair of features. The left column holds results for the cue data set, and the right column for the emotion data set. The plots in the top row were obtained using all components, while the bottom row contains plots calculated solely from manually labeled components.

7.1.3.1 Correlation

Correlations between pairs of features are shown in figure 7.8. Again, the diagonal shows the correlation between a feature and itself. Hence the diagonal contains all ones. As was the case for mutual information, we also see that the correlation between most pairs of features is zero or close to zero. Some patterns

of non-zero values, however, do appear consistently across the four plots.

Features 28 and 29 have a correlation of one in all plots. Since these are the first and second estimates of the Hurst exponent, this is not odd. It is strange, however, that the third estimate (feature 30) is negatively correlated with the first two estimates.

Likewise, features 22 and 23 are also highly correlated in all four plots. These two features also had high MI. Since these features are the logarithms of the mean band power in bands δ and θ, this is not surprising.

Finally, note the lower right corner of all four plots in figure 7.8. Features 32 to 43 measure the success with which QRS complexes were detected at regular intervals in the activation time series of the IC. That some of these features are negatively correlated is somewhat puzzling, while the positive correlation is to be expected.

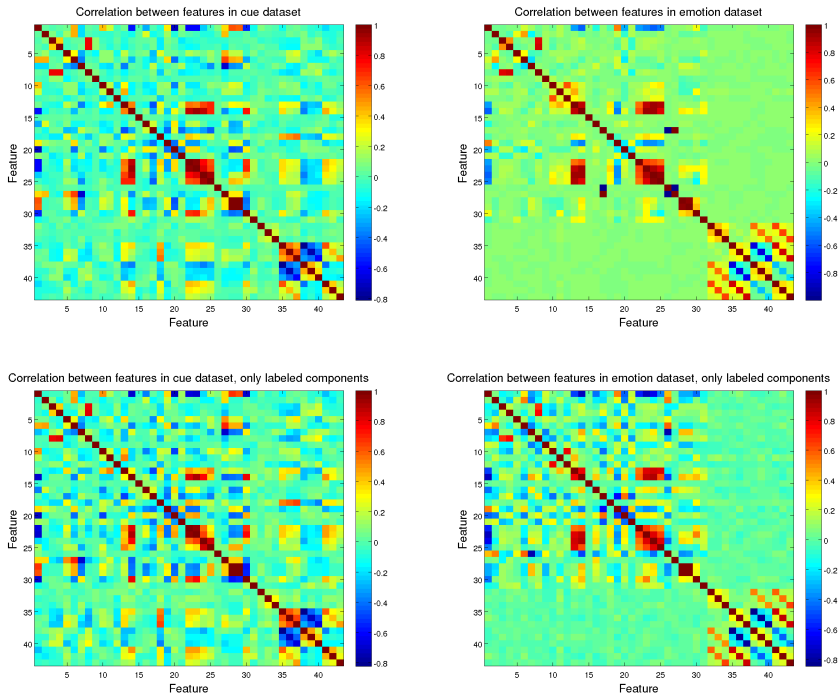

Figure 7.8: Correlation between each pair of features. The left column holds results for the cue data set, and the right column for the emotion data set. The plots in the top row were obtained using all components, while the bottom row contains plots calculated solely from manually labeled components.

7.1.4 Summary

Based on the plots of projections of IC features onto the first three PCs, it seems that a linear separation is achievable, both in the binary and the multiclass setting.

The similar results for the MI between features and class assignment in the binary and multiclass settings indicate that the difficulty in solving the binary and multiclass problems may be similar. It may also be an indication that neural components are easy to identify in general, since neural ICs are assigned to one class in both the binary and multiclass problem.

The low dimensionality of the feature space that seems evident from these analyses (figure 7.5) can be increased by introducing non-linear functions of one or more features. The increase in dimension will hopefully aid in distinguishing between independent components of different types.

Thus classification might be easier using non-linear classifiers, but should be possible with linear methods. Also, features differ substantially in how much information related to class assignments they carry. The features that carry most information are the same both when unlabeled ICs are included and when they are not, and in the binary and multiclass problems.

7.2 Performance of classification methods

We investigated classification within four paradigms determined by two binary variables. One binary variable was whether to include all ICs, or just the ICs with labels. The second binary variable determined whether or not the classification would be binary. With binary classification, we distinguish between neural and non-neural components. Otherwise we try to classify into each class, such that different types of artifacts are distinguished between.

Classification methods were tested in each of the four scenarios, both using all features and using only those chosen by the MI-criterion. Using only features chosen by the MI-criterion did not improve performance. Evidence for this is given in appendix J. Hence we only give the performance results of classification methods using all features. More detailed performance measures in the forms of confusion matrices can be seen in appendix J.1.

We compared 12 classification algorithms as well as the BBCI method proposed

in [56] and the algorithm ADJUST, proposed in [41].

As previously mentioned, some of the classification methods are used in two different ways. QDA and LDA are evaluated both as voting schemes between binary models, and taking all classes into consideration at once 5.4.5. In the binary classification paradigms, we would expect the voting scheme versions of a model to give identical results to the version taking all classes into account at once. In the multiclass problems, we would expected the performance of the voting scheme version of a method to be a little lower than for the multiclass version of the method.

Also, we expect the standard version and the BCILab versions of LDA, which are equal as shown in appendix B, to perform equally well both in binary classification and when used for multiclass problems through voting schemes.

First, we show box plots of the misclassification measure *mcm*, described in subsection 5.4.7, for each model from the ten cross-validation folds. We then discuss the means and standard deviations of the means of *mcm* for each model in each scenario. Next, we give rankings of models in the cases where the ranking is transitive. These rankings are based on the number of times each model outperforms another model, as described in subsection 5.4.7. Finally, we relate the performance of the classification methods to the exploratory analyses.

7.2.1 Box Plots

Figure 7.9 shows box plots of the misclassification measures for each of the classification methods. The box plots in figure 7.9 were made based on the misclassification measures from the ten cross-validation folds. Misclassification measures cannot be compared across scenarios, but only across models in the same scenario. The top row of figure 7.9 shows results from the scenarios in which only manually labeled ICs were considered. The bottom row shows results from using all ICs. The left column shows results from classification into multiple classes, while the right column concerns binary classification. Red lines in figure 7.9 indicate median values, and the boxes around them show 95% confidence intervals for the medians.

7.2.1.1 Multiclass classification using only labeled ICs

Most models show similar performance in the multiclass scenario using only labeled ICs. The BBCI method and QDA, however, have substantially higher

misclassification measures than the other methods. Since QDA is a more complex version of LDA, which obtained the best performance, it seems that the QDA model has been over fit. Such an over fit occurs readily in the data we had available since some classes are very small. Small classes make the estimation of class specific covariance matrices, necessary in QDA, unstable.

Strangely, the binary standard and BCILab versions of LDA perform differently. As expected, the multiclass version of LDA outperforms the binary LDA models with voting schemes.

Contrary to expectations, binary QDA in a voting scheme outperformed the multiclass version of QDA. As mentioned above, this oddity may be attributed to sparsity of data in some classes.

Multiclass LDA was the best model in this scenario.

7.2.1.2 Binary classification using only labeled ICs

When performing binary classification of labeled ICs, logistic regression and logistic regression with forward selection perform substantially worse than all the other models. In particular, MNR and MNR with forward selection perform a lot better, at about the same level of the other models. In the binary case, MNR reduces to logistic regression. Hence these performance differences are very strange.

The medians of misclassification measures of the other models do not differ significantly at the 5% level, as seen from the boxes around the medians. SVM obtains the lowest median, and thus best performance.

7.2.1.3 Multiclass classification using all ICs

The performance of models differs to a large degree when all ICs are used in the multiclass setting . The variance of misclassification measures over cross-validation folds is also large for most models, and all models have at least one outlier. The greater variation in misclassification measures indicates that the classification problem is substantially more difficult to solve when unlabeled ICs are included as a separate class.

Logistic regression with forward selection in the 1v1 voting scheme achieved the best performance, also outperforming MNR with forward selection.

7.2.1.4 Binary classification using all ICs

To compare our results to the state of the art, we classified the components in each of the available data sets using ADJUST [41], which detects different types of artifacts. Only a few types of artifacts are detected both by ADJUST and the methods we investigated. It was not possible to acquire the data used in the development of ADJUST so we resorted to binary classification to allow comparisons between ADJUST and the other models. Since ADJUST goes through an entire data set and picks out the artifactual ICs, the fairest setting for comparing ADJUST to our methods was the binary classification setting involving all ICs. The confusion matrix resulting from this classification is shown in the lower right of figure J.4. The comparison of ADJUST to the other methods is not as clean as the comparisons over the other methods. Since ADJUST works by processing an entire data set at time, and finding artifactual ICs, it was not possible to pass the same cross-validation folds consisting of single ICs to ADJUST as were passed to the other methods for evaluation. We attempted to project out unlabeled ICs to enable comparison in the scenario using only labeled ICs. Unfortunately, something went wrong in the ADJUST algorithm when the data set with unlabeled ICs projected out of data was passed to ADJUST.

Since ADJUST detects artifacts, all components not identified as artifactual were put into the neural component category. Most components were not classified as artifacts, implying that most were classified as neural components. Thus it is not surprising that almost all neural components were classified correctly (lower right square of lower right image in figure J.4). Conversely, only 50% of artifactual components are classified correctly.

Both the voting scheme and multiclass versions of QDA performed substantially worse than all other methods. SVM, MNR, MNR with forward selection, decision trees, LDA, and BBCI performed a little worse than the models with the lowest medians of misclassification measures. These models, that performed very well, and similarly, were the BCILab and standard versions of LDA, logistic regression with forward selection L1-regularized logistic regression, standard logistic regression, and ADJUST. Out of these methods, ADJUST had the largest range of misclassification measures, as well as high outliers. Also, ADJUST was unable to detect about 50% of artifactual components. The best of the other models was L1-regularized logistic regression. L1-regularized logistic regression only missed 10% of the artifactual components. However, this comparison may be unfair to ADJUST since ADJUST solely aims at detecting pure artifacts. In the artifact class that we investigated, mixed components (unlabeled ICs) are also included.

7.2.2 Means and standard deviations of misclassification measures

Table 7.1 shows the mean misclassification measures and the standard deviations of these means for each model tested in each classification scenario. The misclassification measures can only be compared over models within the same classification setting, and not between scenarios.

	Only labeled ICs, multiclass		Only labeled ICs, binary		All ICs, multiclass		All ICs, bi
	Mean	SD	Mean	SD	Mean	SD	Mean
LDA(bcilab), 1vR	19.5	3.06	0.433	0.0757	334	32.1	0.958
LDA(standard), 1vR	21.9	12.4	0.433	0.0757	276	102	0.958
SVM, 1v1	20.8	11.8	0.256	0.036	464	80.8	2.27
QDA, 1vR	15.2	4	0.424	0.0643	582	93.3	9.18
Logistic(forward), 1v1	10.4	1.68	3.24	0.174	153	66	1.11
Logistic(L1), 1v1	19.3	5.51	0.482	0.0386	484	90.4	0.794
Logistic, 1v1	12.7	3.03	3.24	0.174	330	69.6	1.11
MNR	14.2	3.46	0.558	0.0427	450	85.9	3.8
MNR(forward)	10.8	2.53	0.497	0.0508	213	45.9	2.97
Decision tree	20.6	3.69	0.467	0.0434	200	43.2	2.34
LDA	6.51	2.27	0.394	0.0445	303	72.9	4.07
QDA	43.5	10.6	0.381	0.0477	580	93.6	9.05
BBCI	46.8	12.3	0.635	0.101	347	83.4	2.8
ADJUST	-	-	-	-	-	-	1.39

Table 7.1: Means and standard deviations of the misclassification measure *mcm*, described in subsection 5.4.7, for each model over the ten cross-validation folds, in each classification setting. ADJUST could only be tested in the binary setting using all ICs. Thus means and standard deviations of the misclassification measure are not reported for ADJUST in the other classification scenarios. The misclassification measures can only be compared over models within the same classification setting, and not between settings.

The results from table 7.1 correspond very well to the results based on the box plots above. Also, the methods that showed identical performance in the box plots also have identical statistics here, another indication that those methods obtained identical misclassification measures in each cross-validation fold. Using the normal approximation, approximate 95% confidence intervals for the means can be obtained by adding and subtracting two times the standard deviation to the mean.

As seen previously, the BCILab and standard formulations of binary LDA perform identically in binary settings, but differently in the multiclass scenarios.

The multiclass version of LDA outperforms both the binary version when only labeled ICs are used, but is outperformed by both when all ICs are used in the binary setting. Also, the interval calculated from the mean performance of multiclass LDA plus and minus two standard deviations of this mean does not overlap with the performance of the binary LDA methods in these three scenarios. In the multiclass setting using all ICs, though, the interval does contain the mean misclassification measures of both binary LDA versions, indicating no significant difference.

When only labeled ICs are used, the strange phenomenon of binary QDA outperforming multiclass QDA in the multiclass setting, and vice versa in the binary setting, occurs again. The performance of the binary and multiclass versions do not differ significantly when all ICs are used, neither in the multiclass or the binary setting, though.

Logistic regression outperforms multinomial regression, both with and without forward selection, in both multiclass classification scenarios. Conversely, multinomial regression outperforms logistic regression, again both with and without forward selection, in the binary setting using only labeled ICs.

Since BBCI is a SVM with only a subset of the features used in our SVM, we expect the BBCI algorithm to perform worse than SVM consistently. This, however, is not the case in the multiclass scenario using all ICs. The large standard deviations, though, imply that the performances do not differ significantly in this case.

7.2.3 Ranking of models

In the two scenarios using all ICs, a transitive ranking of models was possible. These rankings are based on the number of times a model outperforms the model at the previous rank. If a model outperforms another a significant number of times, we use the inequality symbol in the list of rankings. If a model outperforms the previously model at chance level, under the hypothesis that they perform equally, then we use equality. Since ADJUST was not tested on the same cross-validation folds as the other methods, it does not make sense to compare ADJUST in these rankings. We show the rankings here, from best to worst:

- All ICs, non-binary classification
  Logistic(forward), 1v1 = Decision tree = MNR(forward) = LDA(standard),

1vR = Logistic, 1v1 = LDA = BBCI = LDA(bcilab), 1vR = MNR = SVM, 1v1 = Logistic(L1), 1v1 > QDA = QDA, 1vR

- All ICs, binary classification:
  Logistic(L1), 1v1 = LDA(standard), 1vR > LDA(bcilab), 1vR = Logistic, 1v1 > Logistic(forward), 1v1 > SVM, 1v1 = Decision tree = MNR(forward) = BBCI > MNR = LDA > QDA > QDA, 1vR

These rankings, based on the number of times a model outperforms another, are in agreement with the box plots. However, only few models show significantly different performances, especially in the binary case. This is probably due to the conservative Bonferroni correction, since the p-value used in each binary test of two models was only

$$2\frac{0.05}{12 \cdot 13} \approx 6.41 \cdot 10^{-4}.$$

The p-value of 0.05 is divided by the number of comparisons, which is $\frac{12*13}{2} = 78$, since we compare 13 different models.

7.2.4 Relations to exploratory results

Based on the exploratory results, we expected linear separation of classes to be possible in all four paradigms, albeit easiest with binary classification. It is surprising that decision trees and SVMs, which are both able to catch non-linear effects, were the best methods in the binary setting. When all features were used, though, the performance of LDA was not significantly different. The superiority of SVMs when only some features were used may be because the absence of some features destroys the linear separability.

In the multiclass problem, we found that LDA and QDA performed best. LDA was best when only manually labeled components were used, whereas QDA was best when all ICs were used. This is interesting for several reasons.

Firstly, LDA does not explain non-linear effects, and QDA can only account for non-linear effects that are quadratic. Hence the multiclass problem is more linear than the binary problem, contradicting the hypotheses from the exploratory analyses.

Secondly, the need for quadratic effects in distinguishing between classes when all ICs are used is in line with the exploratory results. The small step in increased

complexity between LDA and QDA in going from only manually labeled ICs to all ICs is evidence that the two problems are really quite similar. This means that the ICs without labels do have characteristics in common by which they can be distinguished. Hence further work should make it possible to detect neural components, artifacts, and noisy components. The noisy components which do not belong to any class cleanly should then be kept since they probably contain at least some neural activity which may be extracted in more advanced analyses.

7.3 Summary

We now put forth some general considerations on the classification performance results.

Firstly, results concerning model performances relative to each other are consistent across the different evaluation methods. This is reassuring since we compared models based on several different measures, namely means and medians of misclassification measures, as well as the number of data partitions in which a model achieved lower misclassification measures than the other models. The consistency over all these measures indicates that the performance differences are not due to a few outliers. If only means had been used, for example, one very high mean misclassification measure in one cross-validation fold for some model would decrease the overall evaluation of that model substantially. By evaluating using medians as well, we avoid this problem. That the results over all evaluation measures agree is then a sign that the models maintained consistent performances over the different cross-validation folds.

Since the BBCI method was proposed as a binary classifier to distinguish artifacts from neural components, a poor performance in the multiclass scenarios was to be expected. However, BBCI outperformed SVM in the multiclass setting using all ICs. The variance of the misclassification measure of both BBCI and SVM were large in this case, though, meaning that the difference in performances might be due to chance.

Also, as already mentioned, the evaluation of ADJUST is probably unfair since ADJUST aims to detect purely artifactual ICs. When evaluating ADJUST, we interpreted all ICs not classified as artifacts as being classified as neural components by ADJUST. A fairer comparison would only look at whether all artifacts were detected, and compare this result to the other methods, which should then be changed to also just detect artifacts, thereby not necessarily classifying all ICs. When tested, ADJUST obtained one of the better, although

not the best, performances compared to the other models.

Classification when not including unlabeled ICs was much better than when unlabeled ICs were included. Since labeled ICs have characteristic features, it is not surprising that classification of only labeled ICs is easier.

Some strange results occurred for logistic regression and QDA. In the multiclass settings, the binary versions of these methods would often outperform the multiclass version, and vice versa.

Table 7.2 shows the best model in each of the four scenarios investigated. These models obtained both the lowest median and mean misclassification measures, as shown in figure 7.9 and table 7.1, respectively.

	Inclusion of labeled ICs only	Inclusion of all ICs
Binary classification	SVM	L1-regularized logistic regression
Multiclass classification	Multiclass LDA	Logistic regression with forward selection in 1v1 voting scheme

Table 7.2: Best model for each of the four classification scenarios.

The classification methods were trained on ICs that were obtained through different algorithms. Since good performance was achieved even though ICs from different algorithms were used, we do not expect that the algorithm used to obtain ICs will influence the classification performance on future data.

7.4 Feature analyses

Variance of the estimated coefficients of features, and the features that were chosen for inclusion in classification are described in this section.

We used some classification methods with built-in feature choosing strategies. These were L1-regularized logistic regression and logistic and multinomial regression with forward selection. We compare the features chosen by these methods to those chosen by the MI criterion, described in subsection 5.3.4.

Next, we compare analytical estimates of coefficient variance to the variance of

coefficient estimates over cross-validation folds. However, we only have analytical estimates of variance for logistic and multinomial regression, and LDA. To ease the analyses, we focus on the binary classification problems.

7.4.1 Best features

In this subsection, we describe the features chosen by each of L1-regularized logistic regression, logistic and multinomial regression with forward selection, and by the MI criterion. The results are split into the four classification scenarios described in subsection 5.4.1.

We quantify selection of features through the fraction of times that each feature was chosen in the ten cross-validation folds. Plots of these fractions and their standard deviations are shown in plots for each of the four methods.

If a feature is chosen for inclusion in the classification method in r of the cross-validation folds, the fraction of times the feature was chosen is obviously $r/10$. We assume that the cross-validation folds are independent of each other. This implies that the number of times a feature is selected is binomially distributed. The variance of the fraction $r/10$ is then

$$\frac{(r/10)(1-r/10)}{10},$$

and the standard deviation is

$$\hat{\sigma} = \sqrt{\frac{(r/10)(1-r/10)}{10}}.$$

The fraction $r/10$ is shown as filled blue circles. We show the uncertainty on $r/10$ by plotting red asterisks at $r/10 - \hat{\sigma}$ and $r/10 + \hat{\sigma}$.

7.4.1.1 Multiclass classification with all independent components

The features chosen by the MI criterion and by methods with feature selection strategies are shown in figure 7.10. Figure 7.10 is constructed from multiclass classification including all ICs.

All features are chosen by L1-regularized logistic regression in all ten cross-validation folds, indicating that a high value for the regularization-parameter λ had been chosen, which implies that all features were needed for good classification. Also, all features except feature 15 (measure of similarity between power curve of IC and that of typical EEG) were chosen in all ten folds by the MI criterion.

None of the ECG detection features were chosen in all ten cross-validation folds by either of logistic or multinomial regression with forward selection.

Fewer features were chosen by MNR than by logistic regression, probably because the advantage of taking all classes into account simultaneously yields classification information such that some features are redundant.

7.4.1.2 Multiclass classification solely with labeled independent components

The features chosen by the MI criterion and by methods with feature selection strategies are shown in figure 7.11.

That L1-regularized logistic regression included all features except feature 43 in all cross-validation folds indicates that a high value was chosen for the regularization parameter λ. A high value for λ, in turn, indicates that the best performance during training was achieved when most features were used to discriminate between classes.

Except for some ECG detection features and feature 15 (measure of similarity between fitted power curve of IC and that of typical EEG), the MI criterion selected all features in all folds.

The results from logistic and multinomial regression with forward selection are similar, as could be expected since logistic regression is a special case of multinomial regression. The difference between the two methods is that multinomial regression takes all classes into account simultaneously, while a voting scheme is used to generalize logistic regression to multiple classes.

Conversely, logistic and multinomial regression with forward selection only selected a few features in all folds. The features that were selected in all folds by both logistic and multinomial regression were features 1 (measure of generic discontinuity), 2 (spatial eye difference), 5 (spatial average difference between frontal and posterior electrodes), 11 (maximal variance), 14 (logarithm of mean power in the α band), 16 (number of Talairach areas assigned to the dipole fit

coordinates of an IC), 17 (z-coordinate of dipole fit), 25 (logarithm of mean power in the θ band), and 26 (x-coordinate of dipole fit).

Feature 7 (the variance of weights on posterior electrodes) was also chosen by logistic regression with forward selection in all folds. Feature 22 (the logarithm of mean power in the δ band) was also chosen by logistic regression. Multinomial regression chose feature 23 (the logarithm of mean power in the θ band), but not 22. Since 22 and 23 were seen to be highly correlated in the exploratory analyses, it is likely that these two features serve the same purpose in logistic and multinomial regression. It seems odd that feature 22 was consistently chosen in logistic regression, while feature 23 was chosen in MNR.

7.4.1.3 Binary classification with all independent components

Figure 7.12 shows the fractions of runs in which each feature was chosen by each of the classification methods with built-in feature choosing strategies, and by the MI-criterion when all ICs are taken into account and only the classes of neural and non-neural ICs are distinguished between.

Almost all features were chosen by the MI-criterion and L1-regularized logistic regression in all ten cross-validation folds. The only two features not chosen by the MI-criterion in all cases were features 15 (measure of similarity between power curve of IC and typical EEG) and 31 (number of fitted dipoles). The only feature not chosen by L1-regularized logistic regression in all folds was the last ECG detection feature.

The plots for logistic and multinomial regression are identical, as they should be since multinomial regression reduces to logistic regression in the binary case. The features chosen were features 1 (spatial kurtosis), 8 (mean of weights of frontal electrodes), 10 (temporal kurtosis), 13 (logarithm of range of time series), 14 (logarithm of mean power in the α-band), 16 (number of anatomical brain areas of dipole fit), 19 (2D DFT of electrode weights), 23 (logarithm of mean power in the θ-band), and 29 (estimate 2 of Hurst exponent).

7.4.1.4 Binary classification solely with labeled independent components

The features chosen by the MI criterion and by methods with feature selection strategies are shown in figure 7.13.

The first thing to notice is that the plots for logistic and multinomial regression are identical. This is reassuring since multinomial regression in the binary case reduces to logistic regression. Since these plots are identical, we refer to both methods as logistic regression with forward selection in the analysis of figure 7.13.

Again, all features except feature 43 were chosen by L1-regularized logistic regression.

The MI criterion did not choose the ECG detection features in all folds, and logistic regression with forward selection did not choose any of the ECG features in any fold. This is in accordance with the exploratory analyses, in which MI between ECG features and class labels was low.

Apart from the ECG detection features, the measure of similarity between the fitted power curve and the typical EEG power curves (feature 15), and the number of fitted dipoles (feature 31), the MI criterion selected all features in all folds.

Conversely, logistic regression with forward selection only selected a few features in all folds.

7.4.2 Variance of feature values

For most of the models, analytical estimates of the variance of coefficient and bias estimates are non-trivial. To estimate the variances, we used the estimates from the ten cross-validation folds.

However, the influence of features cannot be easily derived for QDA and SVM. In QDA, coefficients for each feature as well as for each pair of features, i.e. the quadratic terms, are used. The number of quadratic terms is the square of the number of linear terms, so investigating the variance of all coefficient estimates is impractical. Investigating only linear terms would be meaningless since this would ignore the feature effects from the quadratic terms. In SVM, the support vectors define the model in an infinite dimensional space. Since SVM models are defined by support vectors, i.e. specific training samples, it does not make sense to talk of the variance of coefficient estimates in SVM models. Likewise, coefficients are not estimated in decision trees. Hence it does not make sense to estimate variance of coefficient estimates for decision trees either.

Empirical variance estimates from the coefficient estimates in the ten cross-validation folds for all other models than QDA, SVM, and decision trees are

given in appendix K. For all models, the standard deviations are very small.

To simplify the presentation, we only look at the case of binary classification using labeled ICs.

For logistic regression and LDA, we did have analytical expressions for the variance of coefficient estimates. We compare the analytical results to the empirical variance estimates.

7.4.2.1 Linear discriminant analysis

Since we were only able to derive an analytical expression for an estimate of the variance of the intercept and coefficient in a univariate LDA model, we calculated the LDA model using only one explanatory variable. We used our own implementation of the standard formulation of LDA within the framework of BCILab.

We chose the one variable to include in the model to be feature one since previous analyses showed that feature one is a good predictor of class label.

We used the same ten cross-validation folds that we have used throughout all analyses to calculate the binary LDA model ten times. The variance of the intercept and coefficient estimates over the ten cross-validation folds were $2.2881 \cdot 10^{-4}$ and $1.3479 \cdot 10^{-3}$, respectively. Compare these to the corresponding analytical estimates of $2.1023 \cdot 10^{-3}$ and $5.7121 \cdot 10^{-3}$, respectively. These estimates are surprisingly close. Thus the rule of error propagation is a simple method to estimate the uncertainty of coefficient and intercept estimates for univariate LDA. Details on the analytical variance estimates are given in appendix B. A complete analytical exposition of variance of estimates from LDA is presented in [9]. It seems that the variance estimates presented in [9] are computationally complex, so the simple approximation we present may be better suited for practical use.

7.4.2.2 Logistic regression

Figure 7.14 shows the analytical variance estimates of coefficient estimates. These are incredibly large, in particular compared to the empirical variance estimates. The large variance estimates are most likely due to several fitted probability values close to zero and one. The product of the fitted values and one minus the fitted values is present in the denominator in one of the terms in

the analytical expression for variance estimates. Since fitted values close to one or zero will cause this denominator to be close to zero, the variance estimates blow up.

7.4.3 Summary

Based on the analyses of chosen features, it seems that feature 15 (similarity of IC power curve and that of typical EEG) is the least informative. Additionally, feature 1 (generic discontinuity measured based on scalp map) seems to be very informative. Since neural components tend to exhibit scalp maps with activation over large areas, it makes sense that feature 1 is informative. Another feature that was chosen often was feature 14 (logarithm of mean power in α-band). Feature 14 is also typically high for neural components. In general, the choices of features are in accordance with the exploratory analyses.

The most surprising result is that almost all features were consistently chosen by the MI criterion and by L1-regularized logistic regression.

The best agreement between the feature choosing strategies occurred in the binary case for the ECG features, which were only chosen rarely.

Furthermore, the variances of the feature coefficient estimates over the ten cross-validation folds turned out to be small, indicating consistent model estimates over the ten data partitions.

Analytical estimates of the variance of feature coefficient estimates in logistic regression were off by a magnitude of 13. Conversely, the analytical expression for an analytical estimate of the intercept and coefficient estimates in the univariate LDA model were in very good accordance with the empirical variance.

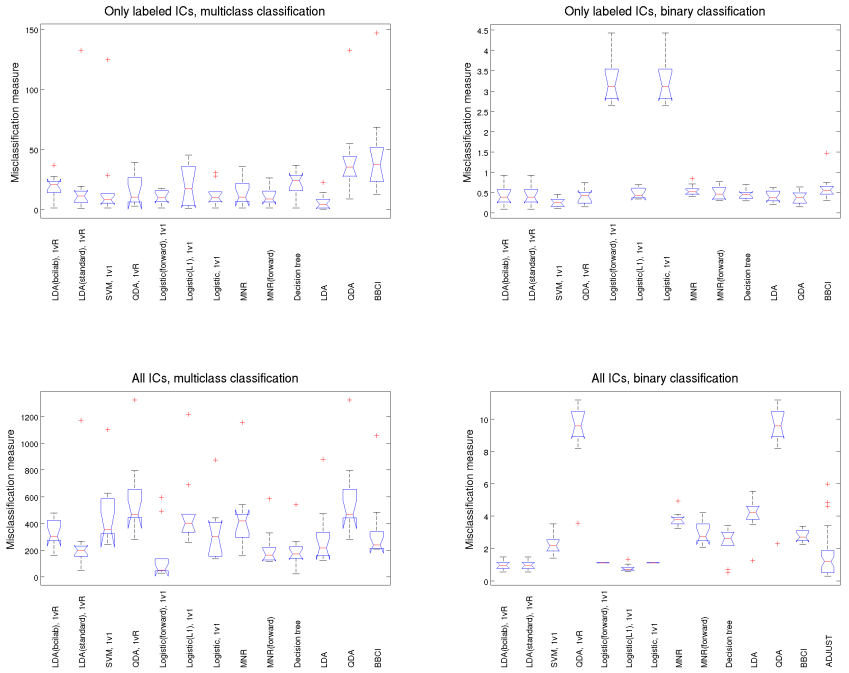

Figure 7.9: Misclassification measure of the investigated classification methods under each scenario described in subsection 5.4.1. The box plots are based on the test error from the ten cross-validation folds. Red lines indicate medians, and the notches (trapezoids to each side of the red lines) represent the 95% confidence intervals. Blue dashed lines, and the horizontal lines at the ends of the blue dashed lines, indicate the range of values. Red asterisks represent outliers. The misclassification measures can only be compared across models within each scenario, and not between scenarios. The top row shows classification performances when only manually labeled ICs were used. The bottom row shows results from classification taking all ICs into account. The left column shows results from classification into multiple classes while the right column shows results from binary classification.

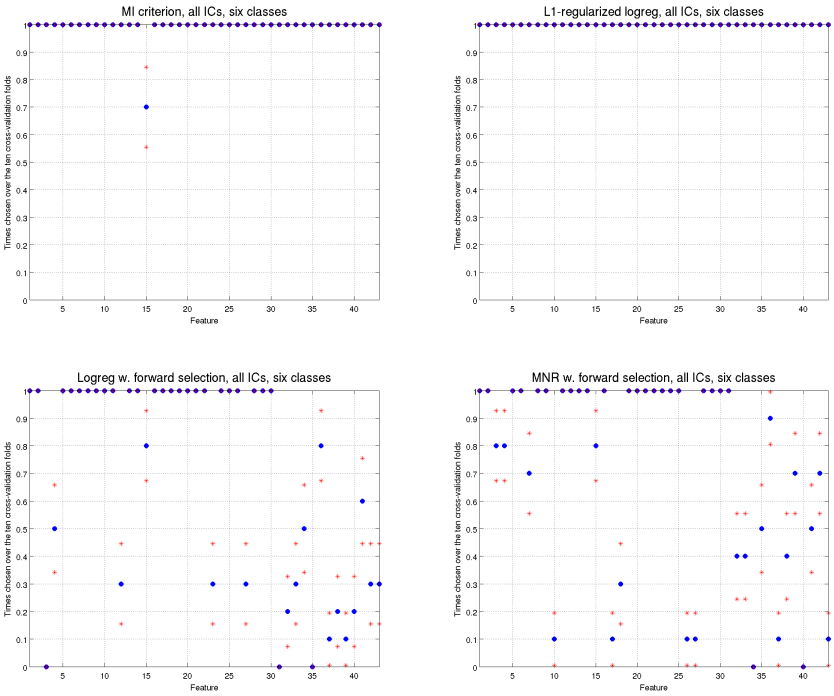

Figure 7.10: Fraction of times that features were chosen in the ten cross-validation folds by the MI criterion, L1-regularized logistic regression, and logistic and multinomial regression with forward selection. All classes were considered, and all ICs were used. Blue circles represent the fraction of times that a feature was chosen. Red asterisks represent the fraction plus and minus the standard deviation of the fraction.

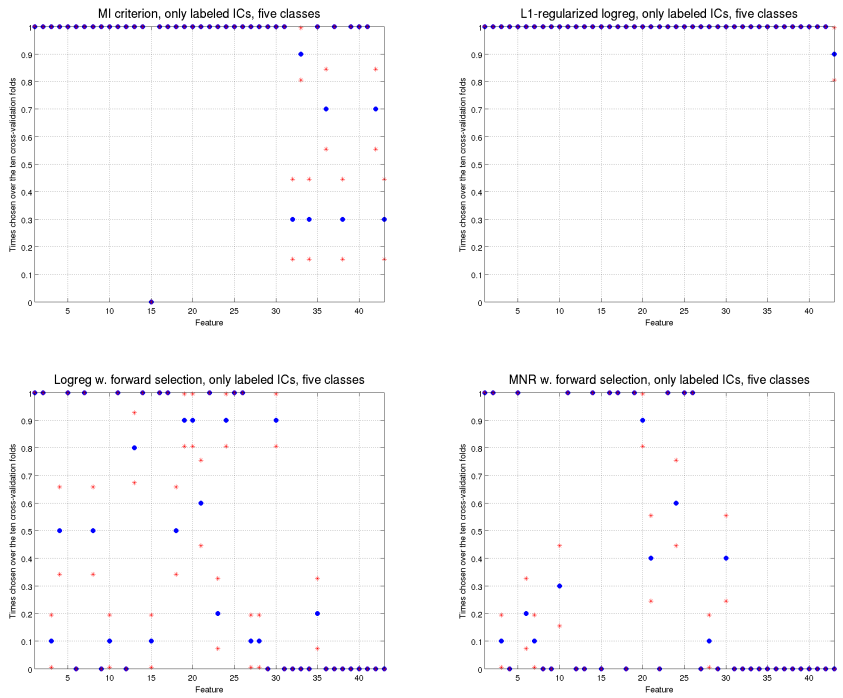

Figure 7.11: Fraction of times that features were chosen in the ten cross-validation folds by the MI criterion, L1-regularized logistic regression, and logistic and multinomial regression with forward selection. All classes were considered, and only labeled ICs were used. Blue circles represent the fraction of times that a feature was chosen. Red asterisks represent the fraction plus and minus the standard deviation of the fraction.

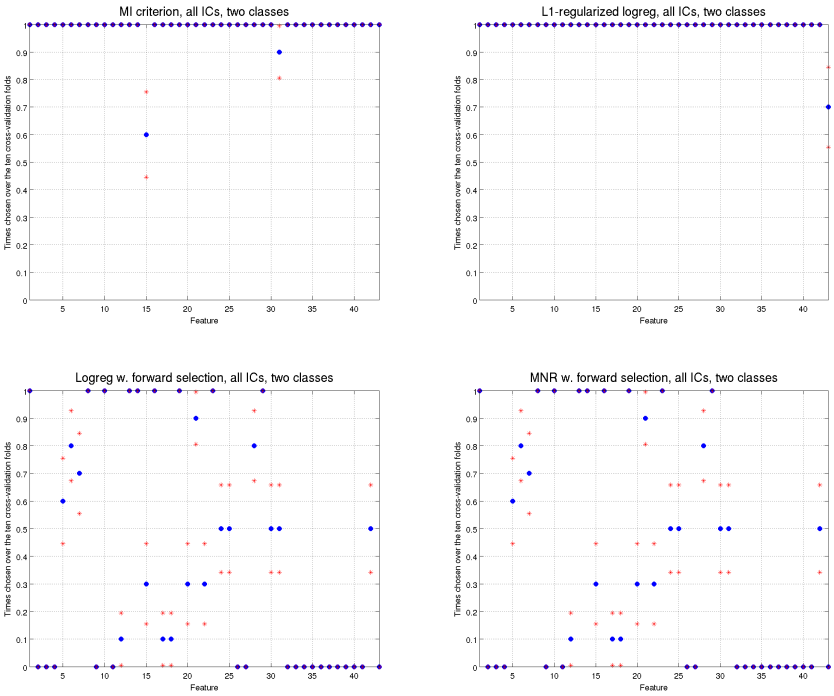

Figure 7.12: Fraction of times that features were chosen in the ten cross-validation folds by the MI criterion, L1-regularized logistic regression, and logistic and multinomial regression with forward selection. Only the classes neural and non-neural were distinguished between, and all ICs were used. Blue circles represent the fraction of times that a feature was chosen. Red asterisks represent the fraction plus and minus the standard deviation of the fraction.

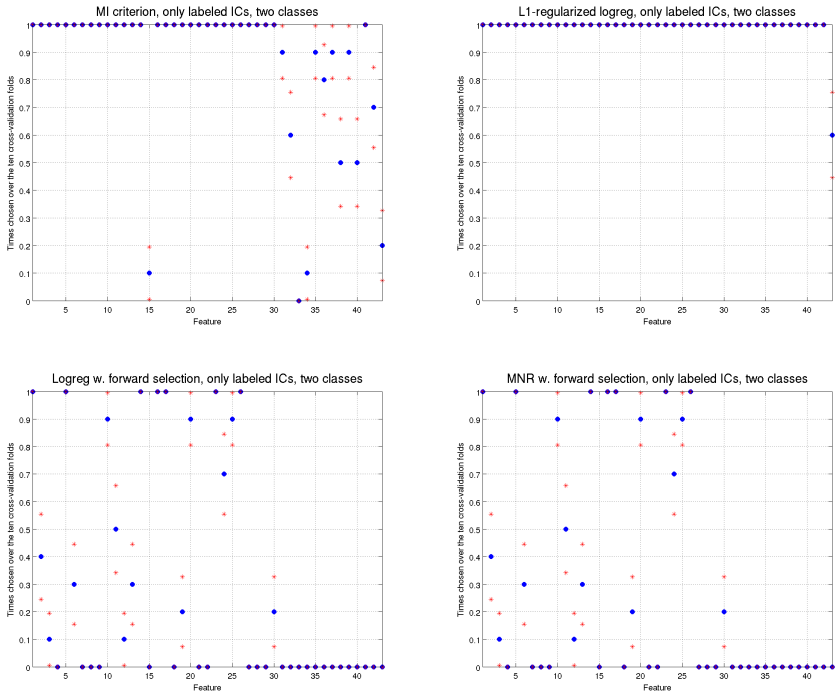

Figure 7.13: Fraction of times that features were chosen in the ten cross-validation folds by the MI criterion, L1-regularized logistic regression, and logistic and multinomial regression with forward selection. Only the classes neural vs. non-neural were considered, and only labeled ICs were used. Blue circles represent the fraction of times that a feature was chosen. Red asterisks represent the fraction plus and minus the standard deviation of the fraction.

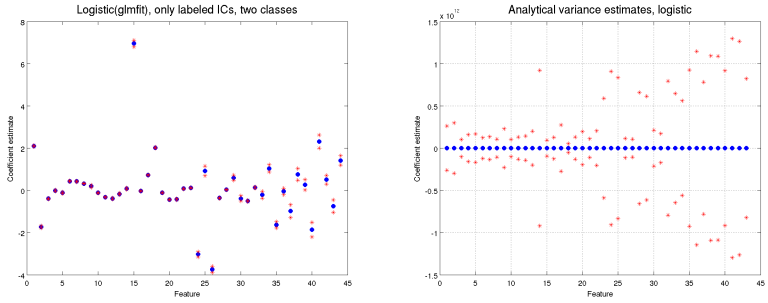

Figure 7.14: Coefficient estimates and analytical variance estimates of the coefficient estimates.

Chapter 8

Seizure detection

We will now describe the results relating to detection of seizures in the EEG of type I diabetics. Section 8.1 gives the results from data reconstruction. Next, we describe the performance of the seizure detection model in section 8.2.

8.1 Estimation of corrupted data

Empirical results showed successful reconstruction of missing data. We used a data set without missing values, from which we removed 7% of the data since this is about the proportion missing from the seizure data. We then reconstructed these missing values. The result of such a reconstruction is shown in figure 8.1.

To quantify the performance of the factorization, we compare the test error

$$test\_error = \frac{(1-W)\|A*S-Y\|_2^2}{\|1-W\|_1}$$

to the training error

$$training\_error = \frac{W\|A * S - Y\|_2^2}{\|W\|_1}$$

on a randomly chosen data set with no missing values. The average test error per value was 65.8959 while the average training error per value was 0.0670. This is a difference of magnitude 100, indicating that the underlying structure of data is not well described by this factorization model. It is likely that improvements can be made by taking temporal patterns into account when reconstructing data in addition to the spatial relations being used by the current model. This is an obvious focus area for future work.

8.2 Performance of seizure detection model

As mentioned previously, we use the developed IC classification method to investigate the influence of artifacts on seizure detection. First we show results from a detection model trained on non-denoised data, and applied to raw test data. Next we show results from the detection model trained on denoised data, and applied to both raw and denoised data.

Originally, we wanted to use the best classification model from the binary classification scenario using all ICs. However, this model did not classify any of the ICs from the HypoSafe data sets as artifactual. Since we wanted to use one of the models trained on all ICs, including the unlabeled ICs, we used the model from the multiclass setting instead.

We first compare the scalp map patterns found by the seizure detection model trained on raw data and that trained on denoised data. Next, we compare seizure detection performance.

8.2.1 Scalp maps

Figure 8.2 shows the scalp patterns found by the seizure detection models trained on raw and denoised data, respectively. Clearly, the scalp patterns are different, indicating that the denoised data is different from the raw data, and that this difference influences the final model. The first scalp pattern for the model trained on raw data (left part of figure 8.2) resembles an IC for eye lateral movements. Likewise, the second scalp pattern resembles the activity seen for ICs representing lateral eye movements. Patterns four, five, and six also look

similar to scalp maps for muscle artifacts. Conversely, only patterns one, five, and six resemble scalp maps from artifactual ICs in the right figure, which shows the scalp patterns used by the model trained on denoised data.

Hence it seems that the model trained on raw data exploits artifacts in seizure detection.

8.2.2 Seizure detection performance

Figure 8.3 shows confusion matrices of the training errors for the model trained on raw data, and that trained on denoised data. Since the two confusion matrices are identical, it seems that non-artifactual patterns that indicate seizures are used by the model to detect seizures.

Figure 8.4 shows confusion matrices for test errors from the model trained on raw data and applied to raw test data, and the model trained on denoised data applied to both raw and denoised test data. The confusion matrices showing test errors from the model trained on denoised data and applied to raw and denoised test data (middle and right) are identical. In both of these, no seizures were detected.

The left part of figure 8.4 shows the test errors resulting from applying the model trained on raw data and tested on raw test data. In this case, 20% of the seizures were detected, and 80% missed. False seizure detections were issued in 15% of the normal state intervals.

The very low performance on test data of the model trained on denoised data hints that part of the seizure detection success of models based on raw data exploit patterns of artifacts during seizures.

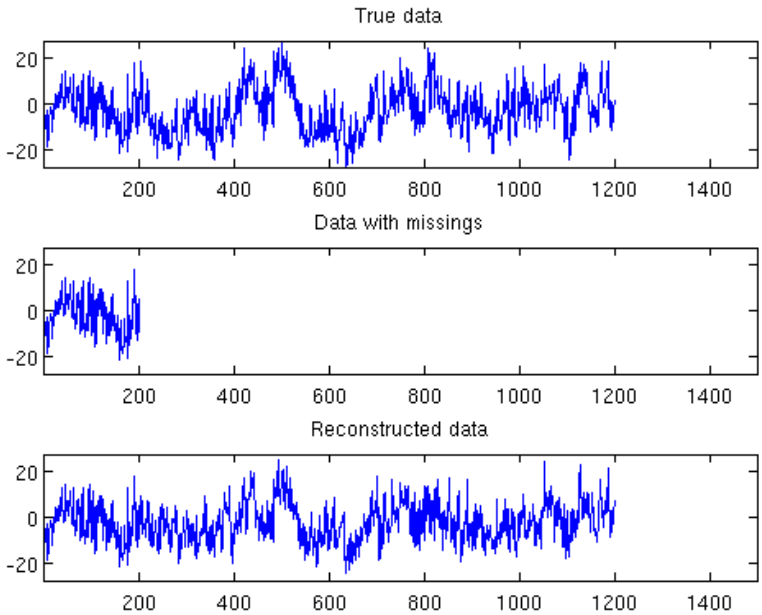

Figure 8.1: Top: original EEG data snippet from one channel. This snippet is part of a larger data matrix with 64 channels and 387200 observations. Middle: Data from top plot with some data removed. Bottom: Missing data in the middle plot has been reconstructed and appended to the non-missing data from the middle plot. The data was reconstructed based on the full data matrix, with other snippets of data removed. About 7% of data was removed in total, since this is approximately the proportion of corrupted data in the HypoSafe data. All data that was removed was removed as entire intervals, resembling patterns of corrupted data due to loose electrodes. The bottom plot is amazingly similar to the top plot containing the true data.

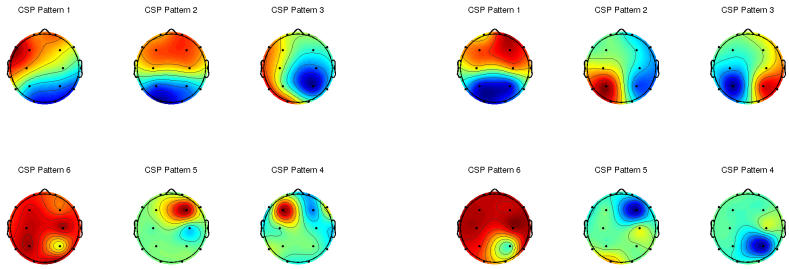

Figure 8.2: The left plot shows the scalp patterns used in classification by the seizure detection model trained on raw data, and the right plot shows the scalp patterns used for seizure detection by the model trained on denoised data.

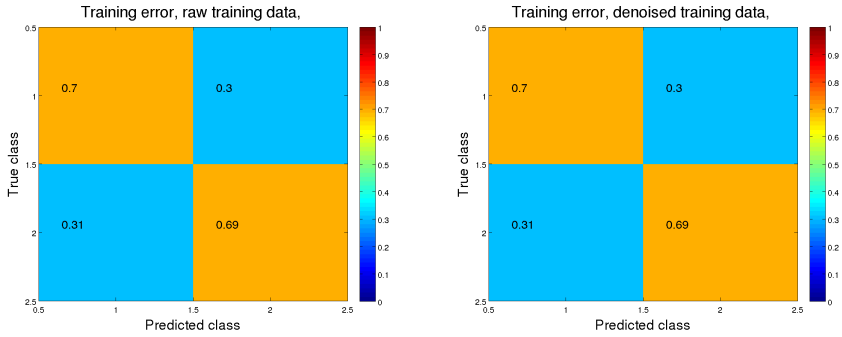

Figure 8.3: Confusion matrices showing seizure detection rates on training data. The left plot shows the performance of the model trained on raw data on the same raw training data. The right plot shows the training errors when denoised training data is used. Class 1 represents states of normal blood sugar and class two represents seizures.

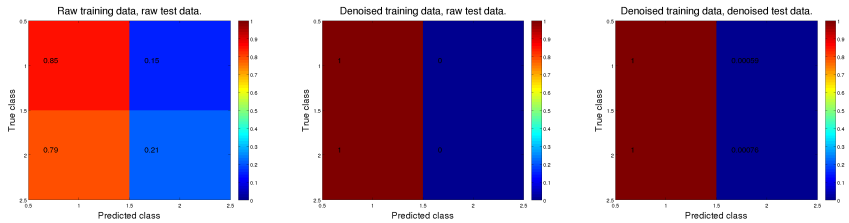

Figure 8.4: Confusion matrices showing seizure detection rates on test data. The left plot shows the performance of the model trained on raw data on raw test data. The middle and right plots show the performance of the model trained on denoised data on raw and denoised test data, respectively. Class 1 represents states of normal blood sugar and class two represents seizures.

Chapter 9

Conclusion

We used the results from the IC classification investigations to compare performance of seizure detection models trained on raw and on data cleaned by removing the ICs classified as artifactual by the best IC classification method. The model trained on raw data was tested on raw data, and the model trained on cleaned data was tested on both raw and clean data. The model trained on clean data showed identical performance on the raw and clean data, detecting no seizures in either case. Conversely, the model trained on raw data correctly identified 20% of the seizures. Although 15% of normal intervals were misclassified as seizures by the model trained on raw data, the performance by the model based on raw data must be said to outperform the model trained on clean data. The superior performance of the model trained on raw data may have been caused by the removal of too much data during the cleaning process, which would be the case if neural components were wrongly classified as artifactual ICs. However, a reliance on characteristic patterns of artifacts during seizures may also explain why the model trained on raw data obtained the best results.

Our two questions relating to the first aim of investigating the effect of artifacts on seizure detection may then be answered as follows. Training on data cleaned by removing artifactual ICs does not improve seizure detection performance, at least not with the classification performances that can currently be obtained. Secondly, yes, it does seem that the presence of artifacts may help seizure detection.

We obtained classification performance of independent components (ICs) surpassing the current state of the art methods, which was one of our aims. However, the presence of mixed ICs, including both neural and artifactual activity, complicates the classification problem. When mixed ICs are not included in classification, very high performance of more than 85% correct classifications in all the classes of eye blinks, neural, heart, lateral eye movement, and muscle ICs is obtained. When mixed ICs are not included, and neural ICs are distinguished from artifactual ICs, 96% of neural ICs are classified correctly, while 91% of artifactual ICs are correctly classified. When including mixed ICs and distinguishing neural from all other ICs, we obtained 90% correct classifications of neural ICs, and 88% correct classifications of non-neural ICs. The best performance previously reported, as far as we know, is 90.5% in the binary problem of distinguishing neural from non-neural ICs [56].

We quantified classification performance using a misclassification measure that caused classes of different sizes to be weighted equally. When including independent components (ICs) containing both neural and artifactual activity in the classification problem, logistic regression with L1-regularization and forward selection of features performed best. Without mixed ICs, LDA and SVM were best, although most of the compared methods performed similarly. The high performance of simple methods such as LDA and logistic regression indicates that it should be possible to achieve higher performance.

The MI-criterion that we used to select features chose almost all features. This may be due to the conservative Bonferroni correction applied to take the multiple hypothesis tests into account. However, it may also be due to a real presence of class information in almost all features. The fact that L1-regularized logistic regression also chose almost all features supports the second reason. Thus we conclude that almost all features (except feature 15) carry class-related information.

Our analytical estimate of the uncertainty of estimates in the univariate LDA model were in good accordance with the empirical estimates. Extending the analytical estimate to the multivariate case presents a way to estimate uncertainty of coefficient estimates even when little data is available.

Also, the empirical variance of coefficient estimates over cross-validation folds was very small for all features in all models. The small variance indicates that the model estimates are robust and reflect real structures in data.

These feature analyses fulfill our third aim of investigating the importance of features on classification performance.

9.1 Future work

Several issues are still open for future investigations. Firstly, the problem of mixed ICs must be dealt with if a practical classifier of ICs is to be constructed. We propose to use probabilities of class membership for this purpose, such that an ICs is classified as a mixed IC if there is no class for which the probability of membership is higher than some threshold. The probabilities may be obtained from a model trained only on labeled ICs, since such models performed much better than when trained on all ICs in our analyses. The problem of mixed ICs may also be circumvented by denoising data based on the ICs from dimension-reduced data, since mixed ICs will probably not appear in such data. This solution was suggested in [64].

The effect of the reference with which EEG was recorded has not been investigated, but may also influence classification performance. It is necessary to take any such effects into account in a generally applicable classifier.

Our results so far indicate that artifacts in EEG recordings may be used to detect seizures, and that it is difficult to detect seizures based solely on neural activity.

Appendix A

Exploratory analyses

Figures A.1, A.2, A.3, and A.4 show the projections of feature vectors from ICs onto 2D coordinate systems. Each 2D coordinate system is spanned by a pair of two of the first three PCs. Colors indicate the class of the IC from which each feature vector was calculated.

These projections of feature vectors are easier to view than the 3D figures 7.1 and7.3 in chapter 7.1 since only two dimensions are shown at a time in figures A.1, A.2, A.3, and A.4.

Figure A.1 shows the projections of feature vectors of ICs onto 2D coordinate systems spanned by two of the first three PCs. Figure A.1 shows projections of all ICs, including the unlabeled ICs. The PCs in figure A.1 are based on the standardized matrices of features from all ICs. As explained in part II, the emotion data set comprises substantially more ICs than the cue data set. The large number of ICs in the emotion data is evident from figure A.1, since the clouds of observations in the left column, which relates to the emotion data, are denser than the clouds in the right column.

One observation is clearly an outlier along the third PC in the emotion data. The outlier makes it very difficult, if not impossible, to discern patterns along the other PC when the third PC is included in the plot. Along the first and second PCs, though, a cloud of neural and on of unlabeled ICs are clear, although

overlapping. Clouds representing the other classes cannot be seen. Thus the right column of figure A.1 indicates that neural and mixed ICs might be separable, but separation of artifactual ICs is likely to be difficult.

Projections of IC feature vectors from the cue data are shown in the right column of figure A.1. Again, the cloud of projections of neural ICs only overlaps to a small degree with the cloud of projections of feature vectors from mixed ICs, in all three plots in the right column of figure A.1. Thus it seems that neural and mixed ICs are separable, but that artifactual ICs will be difficult to distinguish from mixed and neural ICs.

Figure A.2 shows projections of feature vectors from labeled ICs onto each pair of the first three PCs derived from the standardized matrix of feature vectors from labeled ICs for each of the cue and emotion data sets. Plots relating to the emotion data are shown in the left column of figure A.2 while the right column relates to ICs from the cue data.

For both the cue and the emotion data, the first PC separates the class of neural ICs from artifactual classes well. In the emotion data, the third PC separates the heart ICs from the other classes. However, the clouds of different artifacts overlap to a great extent in both the cue and emotion data, for all the first three PCs. This overlap indicates that it might be difficult to distinguish between types of artifacts.

Figures A.3 and A.4 show the same projections as figures A.1 and A.2, but projections are only colored according to the classes neural and non-neural. Figure A.3 includes all ICs, while figure A.4 only includes projections of feature vectors from labeled ICs. Again, the left columns show projections of feature vectors from ICs from the emotion data, while the right columns show feature vectors of ICs from the cue data.

The first PC separates neural from non-neural feature vectors well, both when unlabeled ICs are included and when they are not, and for both data sets.

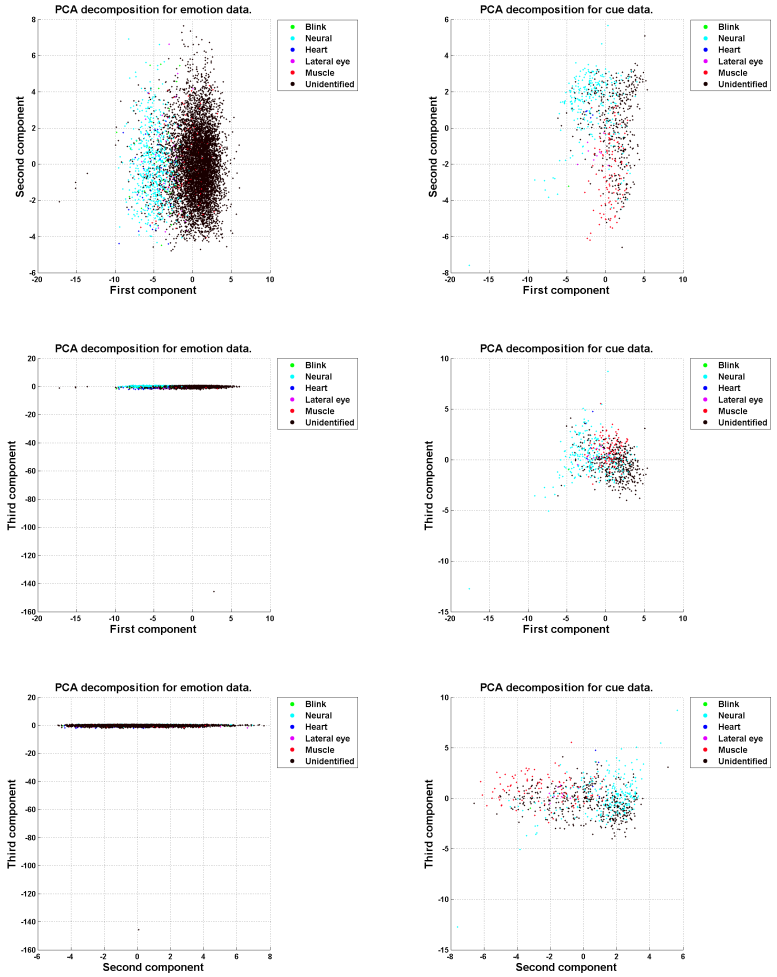

Figure A.1: Face-on views of two principal components at a time. These components are the first three PCs that result from a PCA of the cue and emotion data sets when unlabeled ICs are included. The PCAs were performed on the standardized feature matrices of the ICs for each of the cue and emotion data sets. Projections of feature vectors of ICs are colored according to the class of the IC, either eye blink, neural, heart, lateral eye movement, muscle, or unlabeled

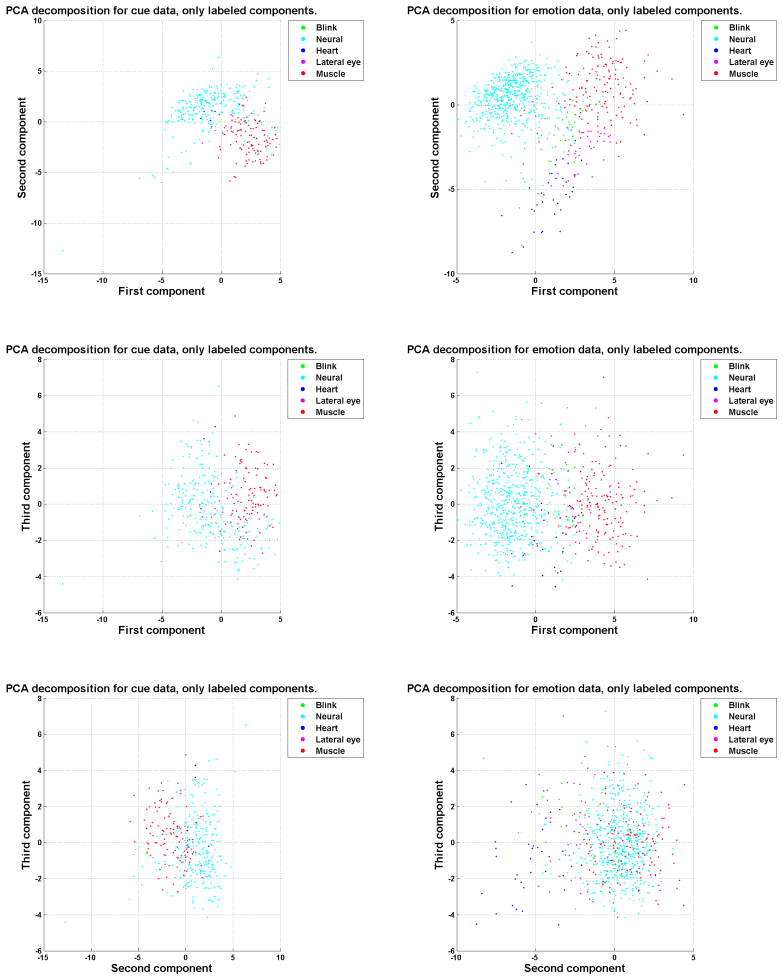

Figure A.2: Face-on views of two components at a time. The three first PCs shown here were calculated based only on the labeled ICs for each of the cue and emotion data sets. The PCAs were performed on the standardized feature matrices. Projections of feature vectors of ICs are colored according to the class of the IC, either eye blink, neural, heart, lateral eye movement, or muscle.

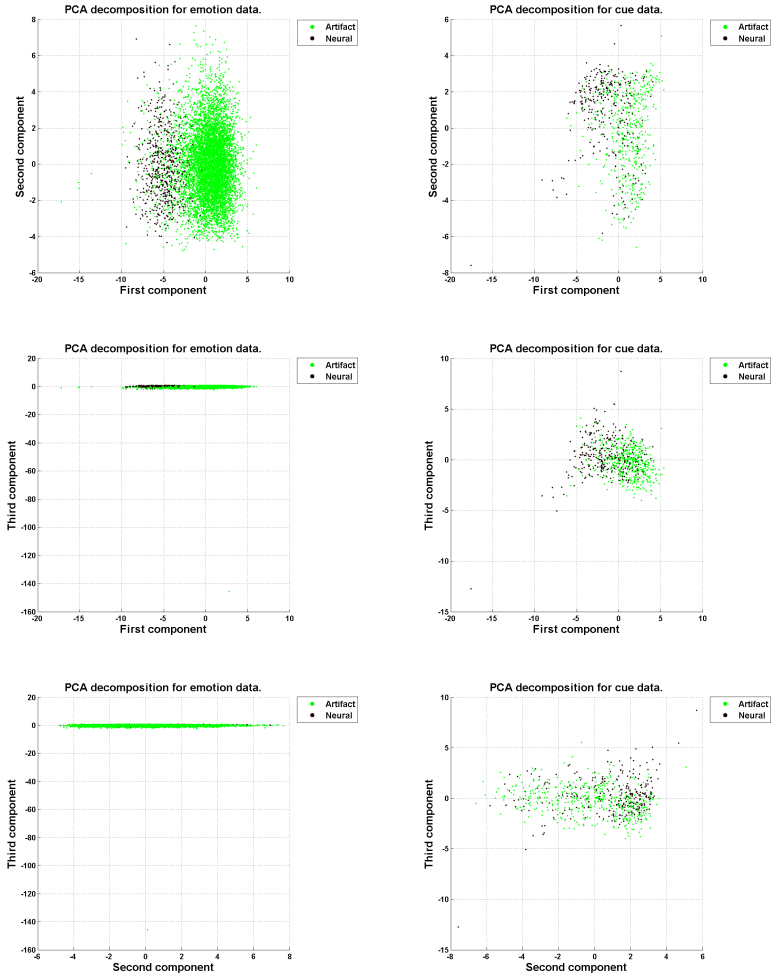

Figure A.3: Face-on views of two principal components at a time. These components are the first three PCs that result from a PCA of the cue and emotion data sets when unlabeled ICs are included. The PCAs were performed on the standardized feature matrices of the ICs for each of the cue and emotion data sets. Projections of feature vectors of ICs are colored according to the class of the IC, either neural or non-neural.

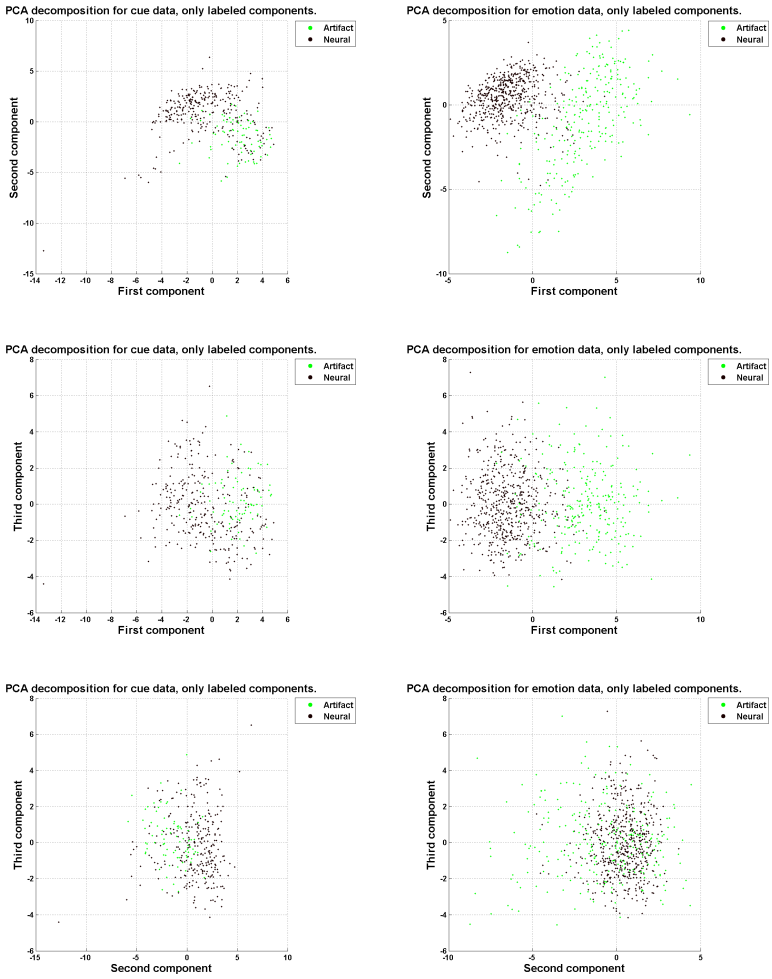

Figure A.4: Face-on views of two components at a time. The three first PCs shown here were calculated based only on the labeled ICs for each of the cue and emotion data sets. The PCAs were performed on the standardized feature matrices. Projections of feature vectors of ICs are colored according to the class of the IC, either neural or non-neural.

Appendix B

Linear Discriminant Analysis

Assumptions of normally distributed data and a common covariance matrix for all classes are exploited by the classifier "linear discriminant analysis" (LDA). For simplicity, we only present the case of binary classification here, such that the number of classes K is equal to two. However, we use general terms whenever a more general presentation does not introduce extra complexity.

LDA classifies an observation to the class that results in the highest likelihood of the observation. Following [57, pp. 20-21], we minimize the expected prediction error by classifying an observation $\boldsymbol{x}$ into class k as follows

$$k = \underset{k \in \{1,2,\ldots,K\}}{\arg\min} \sum_{i=1}^{K} \mathbb{P}(\mathcal{Y} = i|\boldsymbol{x}) \cdot loss(i,k),$$

where $\mathcal{Y}$ is a random variable denoting the class of the observation $\boldsymbol{x}$. The function $loss(i,k)$ is the loss imposed by classifying an observation from class i into class k. When i is equal to k, the loss is zero. If the loss is set equal to one whenever i is not equal to k, then the observation $\boldsymbol{x}$ will be classified into the class i that the observation $\boldsymbol{x}$ is most likely to belong to, i.e. the class i that maximizes $\mathbb{P}(\mathcal{Y} = i|\boldsymbol{x})$.

Let f_k denote the probability density function for class k and π_k the prior probability that an observation is from class k. We use the proportion of samples from each class as prior probabilities, such that $\pi_k = \frac{n_k}{n}$ where n is the total number of observations and n_k is the number of observations from class k. Using Baye's rule, we find the probability that an observation $\boldsymbol{x}$ belongs to class k to be

$$\mathbb{P}(\mathcal{Y} = k|\boldsymbol{x}) = \frac{f_k(\boldsymbol{x})\pi_k}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2}.$$

We assume that observations from both classes are normally distributed with the same covariance matrix, as shown in (B.1)

$$\begin{aligned} \boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma) \text{ for } \boldsymbol{x} \text{ from class 1} \\ \boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma) \text{ for } \boldsymbol{x} \text{ from class 2.} \end{aligned} \tag{B.1}$$

Thus we have that

$$f_k(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}^m \sqrt{|\Sigma|}} \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)),$$

where m is the number of variables and thus dimension of $\boldsymbol{\mu}_k$.

Instead of using the zero-one loss function described above, we use the strategy described in subsection 5.4.3. Thus we set $loss(i, k)$ equal to zero if i is equal to k, but use the loss $\frac{n_i}{n}$ if i is not equal to k. Writing this out, the objective function that we wish to minimize becomes

$$\begin{aligned} \sum_{i=1}^{K} \mathbb{P}(\mathcal{Y} = i|\boldsymbol{x}) \cdot loss(i, k) &= \sum_{i=1}^{K} \frac{f_i(\boldsymbol{x})\pi_i}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2} \cdot \frac{n}{n_i} \cdot \delta_{i,k} \\ &= \sum_{i=1}^{K} \frac{f_i(\boldsymbol{x})\pi_i}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2} \cdot \frac{1}{\pi_i} \cdot \delta_{i,k} \\ &= \sum_{i=1}^{K} \frac{f_i(\boldsymbol{x})}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2} \cdot \delta_{i,k}, \end{aligned}$$

where $\delta_{i,k}$ is the Kronecker delta function which is zero if i and k are equal and one otherwise. To minimize the objective function, we need to classify into the class for which $\frac{f_i(\boldsymbol{x})}{f_1(\boldsymbol{x})\pi_1+f_2(\boldsymbol{x})\pi_2}$ is highest. Hence we classify into class one when

$$\left(\frac{f_1(\boldsymbol{x})}{f_1(\boldsymbol{x})\pi_1+f_2(\boldsymbol{x})\pi_2}\right) / \left(\frac{f_2(\boldsymbol{x})}{f_1(\boldsymbol{x})\pi_1+f_2(\boldsymbol{x})\pi_2}\right) = \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq 1 \tag{B.2}$$

We manipulate the above requirement to get a linear expression in the observation $\boldsymbol{x}$.

$$\begin{aligned}
&\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq 1 \Leftrightarrow \log\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) \geq 0 \Leftrightarrow \\
&\log\left(\frac{\frac{1}{\sqrt{2\pi}^k\sqrt{|\Sigma|}}\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1))}{\frac{1}{\sqrt{2\pi}^k\sqrt{|\Sigma|}}\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)))}\right) \geq 0 \Leftrightarrow \\
&\log\left(\frac{\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1))}{\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)))}\right) \geq 0 \Leftrightarrow \\
&-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) - \left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right) \geq 0 \Leftrightarrow \\
&-\frac{1}{2}(\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1) - \\
&(-\frac{1}{2}(\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x} - \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{x} + \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2)) \geq 0 \Leftrightarrow \\
&\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 \geq 0 \Leftrightarrow \\
&\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 \geq 0.
\end{aligned} \tag{B.3}$$

We conclude that those observations, $\boldsymbol{x}$, for which $\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2$ is greater than zero should be classified as class one.

B.0.1 BCILab version of LDA

The implementation of LDA in BCILab differs from the standard version. We wanted to make sure that the two formulations were equivalent, both analytically

and empirically. We show the analytical equivalence here, and compare the results from the two versions in chapter IV.

First we introduce the row vectors $\boldsymbol{w}_1^T$ and $\boldsymbol{w}_2^T$.

$$
\begin{aligned}
\boldsymbol{w}_1^T &= (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} \\
\boldsymbol{w}_{\mathbf{2}}^T &= \boldsymbol{w}_1^T (\boldsymbol{\mu}_2^T \boldsymbol{w}_1 - \frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2} \boldsymbol{w}_1)^{-1} \\
&= (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} \left(\boldsymbol{\mu}_2^T \Sigma^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - \frac{(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^T}{2} \Sigma^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \right)^{-1} \\
&\text{in the above we used the symmetry of } \Sigma \text{ to get } (\Sigma^{-1})^T = (\Sigma^T)^{-1} = \Sigma^{-1} \\
&= (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} \left(\boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_1 - \frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 \right)^{-1} \\
&= (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} \left(\frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_1 \right)^{-1} \\
&= (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} \left(\frac{1}{2} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \right)^{-1} = \\
&= \frac{2}{a} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} \\
&\text{where } a = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1).
\end{aligned}
$$

The model can now be expressed as

$$
\boldsymbol{x}^T \boldsymbol{w}_{\mathbf{2}} - \frac{1}{2} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^T \boldsymbol{w}_{\mathbf{2}}, \tag{B.4}
$$

where we classify to class one when B.4 is less than zero.

$$\begin{aligned}
\boldsymbol{x}^T\boldsymbol{w_2} - \frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\boldsymbol{w_2} &= \boldsymbol{x}^T\frac{2}{a}\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - \frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\frac{2}{a}\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\\
&= \frac{2}{a}\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - \frac{1}{a}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\\
&= \frac{1}{a}(2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1))\\
&= \frac{1}{a}(2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\Sigma^{-1}\\
&\text{Use that } \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_1 = \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_2 \text{ to reduce}\\
&= \frac{1}{a}(2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1))\\
&= \frac{1}{a}(2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1)\\
&\text{Use that } a = (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) \Rightarrow a + 2\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_2 - 2\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2\\
&\qquad = \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 - 2\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_2 + 2\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_2 - 2\boldsymbol{\mu}_2^T\Sigma\\
&\qquad = \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 \text{ to get}\\
&= \frac{1}{a}(2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) + a + 2\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_2 - 2\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2)\\
&= \frac{1}{a}(2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) + 2(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^T\Sigma^{-1}\boldsymbol{\mu}_2) + 1\\
&= \frac{2}{a}(\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^T\Sigma^{-1}\boldsymbol{\mu}_2) + 1\\
&= \frac{2}{a}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) + 1.
\end{aligned}$$

By classifying to class one when the above expression is less than or equal to zero, we get the same model as before, derived in (B.3). This is what is done in BCILab.

$$\begin{aligned}
\frac{2}{a}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) + 1 \leq 0 &\Leftrightarrow 2(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) \leq -a\\
&\Leftrightarrow 2(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) \leq -(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\\
&\Leftrightarrow 2(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) \leq 2\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\Sigma\\
&\Leftrightarrow 2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - 2(\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_1)\\
&\leq 2\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1\\
&\Leftrightarrow 2\boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - \boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 \leq 0\\
&\Leftrightarrow 0 \leq \boldsymbol{x}^T\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1.
\end{aligned}$$

This shows the analytical equality of the standard formulation of LDA and that used in BCILab.

B.0.2 Uncertainty of coefficient estimates, univariate case

We use the rule of error propagation E to estimate the variance of the coefficient estimates. The case of a single variable is considerably simpler than the multivariate case, so we first present the univariate case. The coefficient estimate is then

$$x\sigma^{-1}(\mu_1 - \mu_2),$$

where σ is the common standard deviation of the two populations, and μ_1 and μ_2 the respective means. The intercept estimate is

$$\frac{1}{2}\mu_2^T\sigma^{-1}\mu_2 - \frac{1}{2}\mu_1^T\sigma^{-1}\mu_1.$$

Since the true values σ, μ_1, and μ_2 are unknown they must be estimated from data. Let n_1 denote the number of observations from population one, n_2 the number of observations from population two, and set $n = n_1 + n_2$. Since data is assumed normally distributed we have the following distributions for the respective estimators s, $\bar{X}_1$, and $\bar{X}_2$

$$\begin{aligned}
&\bar{X}_i \sim \mathcal{N}(\mu_i, \frac{\sigma^2}{n_k}) \\
&\frac{(n-1)s^2}{\sigma^2} \sim \chi^2(n-1) \Leftrightarrow s^2 = \frac{\sigma^2}{n-1}\mathcal{Z},
\end{aligned}$$

where $\mathcal{Z} \sim \chi^2(n-1)$.

The expected values and variances of each of these quantities are as follows

$$\begin{aligned}
&E(\bar{X}_i) = \mu_i \quad Var(X_i) = \frac{\sigma^2}{n_k} \\
&E(s^2) = \sigma^2 \quad Var(s^2) = Var\left(\frac{\sigma^2}{n-1}Z\right) = \frac{\sigma^4}{(n-1)^2} \cdot 2(n-1) = \frac{2\sigma^4}{(n-1)},
\end{aligned}$$

where we implicitly assume that the variance is estimated based on all data at once, and not an aggregate estimate from the two variance estimates from the separate populations.

Due to the assumption of normality, we have independence relations between the estimators of mean values and of variance. Since there is no relation between the two populations, the two estimators of mean values are also independent

$$\bar{X}_1 \perp s^2, \quad \bar{X}_2 \perp s^2, \quad \bar{X}_1 \perp \bar{X}_2.$$

This allows us to use the simple rule of error propagation, since independence implies zero correlation. First we define the function that defines the coefficient estimates (f_1) and the function that defines the intercept estimate (f_2).

$$\begin{aligned} f_1(\bar{X}_1, \bar{X}_2, s) &= \frac{1}{s^2}(\bar{X}_1 - \bar{X}_2) \\ f_2(\bar{X}_1, \bar{X}_2, s) &= \frac{1}{2}\bar{X}_2^2\frac{1}{s^2} - \frac{1}{2}\bar{X}_1^2\frac{1}{s^2} = \frac{1}{2s^2}(\bar{X}_2^2 - \bar{X}_1^2). \end{aligned}$$

Then we take the derivative of each of these functions with respect to each coordinate.

$$\begin{aligned} \frac{\partial f_1(\bar{X}_1, \bar{X}_2, s)}{\partial \bar{X}_1} &= \frac{1}{s^2} \\ \frac{\partial f_1(\bar{X}_1, \bar{X}_2, s)}{\partial \bar{X}_2} &= -\frac{1}{s^2} \\ \frac{\partial f_1(\bar{X}_1, \bar{X}_2, s)}{\partial s^2} &= \frac{1}{s^4}(\bar{X}_2 - \bar{X}_1) \end{aligned}$$

$$\begin{aligned} \frac{\partial f_2(\bar{X}_1, \bar{X}_2, s)}{\partial \bar{X}_1} &= -\frac{\bar{X}_1}{s^2} \\ \frac{\partial f_2(\bar{X}_1, \bar{X}_2, s)}{\partial \bar{X}_2} &= \frac{\bar{X}_2}{s^2} \\ \frac{\partial f_2(\bar{X}_1, \bar{X}_2, s)}{\partial s^2} &= \frac{1}{2s^4}(\bar{X}_1^2 - \bar{X}_2^2). \end{aligned}$$

We can now plug these calculations into the rule of error propagation to get the desired variance of the coefficient estimate.

$$\begin{aligned} Var(f_1(\bar{X}_1, \bar{X}_2, s)) &\approx Var(\bar{X}_1)\frac{1}{s^4} + Var(\bar{X}_2)\frac{1}{s^4} + Var(s^2)\left(\frac{1}{s^4}(\bar{X}_2 - \bar{X}_1)\right)^2 \\ &= \frac{\sigma^2}{n_1}\frac{1}{s^4} + \frac{\sigma^2}{n_2}\frac{1}{s^4} + \frac{2\sigma^4}{(n-1)}\left(\frac{1}{s^4}(\bar{X}_2 - \bar{X}_1)\right)^2. \end{aligned}$$

Since we still do not know the true value σ, we use s instead. This allows simplification of the expression

$$\begin{aligned} &\frac{\sigma^2}{n_1}\frac{1}{s^4} + \frac{\sigma^2}{n_2}\frac{1}{s^4} + \frac{2\sigma^4}{(n-1)}\left(\frac{1}{s^4}(\bar{X}_2 - \bar{X}_1)\right)^2 \\ &\approx \frac{s^2}{n_1}\frac{1}{s^4} + \frac{s^2}{n_2}\frac{1}{s^4} + \frac{2s^4}{(n-1)}\left(\frac{1}{s^4}(\bar{X}_2 - \bar{X}_1)\right)^2 \\ &= \frac{1}{n_1 s^2} + \frac{1}{n_2 s^2} + \frac{2(\bar{X}_2 - \bar{X}_1)^2}{(n-1)s^4}. \end{aligned}$$

We now find the variance of the intercept estimate in the same way

$$\begin{aligned} Var(f_2(\bar{X}_1, \bar{X}_2, s)) &\approx Var(\bar{X}_1)\frac{\bar{X}_1^2}{s^4} + Var(\bar{X}_2)\frac{\bar{X}_2^2}{s^4} + Var(s^2)\left(\frac{1}{2s^4}(\bar{X}_1^2 - \bar{X}_2^2)\right)^2 \\ &= \frac{\sigma^2}{n_1}\frac{\bar{X}_1^2}{s^4} + \frac{\sigma^2}{n_2}\frac{\bar{X}_2^2}{s^4} + \frac{2\sigma^4}{(n-1)}\left(\frac{1}{2s^4}(\bar{X}_1^2 - \bar{X}_2^2)\right)^2 \\ &\approx \frac{s^2}{n_1}\frac{\bar{X}_1^2}{s^4} + \frac{s^2}{n_2}\frac{\bar{X}_2^2}{s^4} + \frac{2s^4}{(n-1)}\left(\frac{1}{2s^4}(\bar{X}_1^2 - \bar{X}_2^2)\right)^2 \\ &= \frac{\bar{X}_1^2}{n_1 s^2} + \frac{\bar{X}_2^2}{n_2 s^2} + \frac{1}{2(n-1)s^4}\left(\bar{X}_1^2 - \bar{X}_2^2\right)^2. \end{aligned}$$

B.0.3 Uncertainty of coefficient estimates, multivariate case

The multivariate case is more complex. In this case, we have the following distributional assumptions

$$\mathcal{X}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma), \quad \bar{\mathcal{X}_1} \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma/(n_1))$$
$$\mathcal{X}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma), \quad \bar{\mathcal{X}_2} \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma/(n_2))$$
$$\hat{\Sigma} \sim W_p(\Sigma, n-1),$$

where W_m denotes the m^{th} dimensional Wishart distribution and $n-1$ is the number of degrees of freedom for the covariance matrix estimate $\hat{Sigma}$. As above, we assume that the covariance matrix is estimated based on all observations. The estimate of the inverse covariance matrix, $\Psi = \Sigma^{-1}$ then follows the m^{th} dimensional inverse Wishart distribution,

$$\hat{\Psi} \sim IW(\Sigma^{-1}, n-1).$$

Since $\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = \Psi(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ gives the vector of coefficients, the j^{th} coefficient β_j is

$$\beta_j = \sum_{h=1}^{m} \psi_{j,h}(\boldsymbol{\mu}_{1,h} - \boldsymbol{\mu}_{2,h}),$$

where $\psi_{j,h}$ is the entry in the j^{th} row and h^{th} column of Ψ. I have not been able to find a reference describing the correlation structure of elements of Ψ. Hence I cannot apply the chain rule as desired, and my analytical treatment of the variance of coefficients in the multivariate case stops here.

A complete analysis of estimates from LDA is presented in [9].

Appendix C

Quadratic Discriminant Analysis

As in linear discriminant analysis, we start out with the ratio of the densities of the two distributions into which we wish to classify observations. We classify into class one if this ratio is larger than or equal to one.

$$\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq 1.$$

Both f_1 and f_2 are assumed to be multivariate normal. No assumptions are made about their means or covariance structures. We take logarithms and simplify to get the classification criterion.

$$
\begin{aligned}
\log\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) &= \log(f_1(\boldsymbol{x})) - \log(f_2(\boldsymbol{x})) \\
&= \log\left(\frac{1}{\sqrt{2\pi}^k\sqrt{|\Sigma_1|}}\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1))\right) \\
&- \log\left(\frac{1}{\sqrt{2\pi}^k\sqrt{|\Sigma_2|}}\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2))\right) \\
&= \log((2\pi)^{-k/2}) + \log(|\Sigma_1|^{-1/2})\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right) \\
&- \left[\log((2\pi)^{-k/2}) + \log(|\Sigma_2|^{-1/2})\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right)\right] \\
&= \frac{1}{2}\log\left(|\Sigma_2|^{-1/2}\right)(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) - \frac{1}{2}\log\left(|\Sigma_1|^{-1/2}\right)(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) \\
&= \frac{1}{2}\Big[\log\left(|\Sigma_2|^{-1/2}\right)\left\{\boldsymbol{x}^T\Sigma_2^{-1}\boldsymbol{x} + \boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{\mu}_2 - 2\boldsymbol{x}^T\Sigma_2^{-1}\boldsymbol{\mu}_2\right\} \\
&- \Big[\log\left(|\Sigma_1|^{-1/2}\right)\left\{\boldsymbol{x}^T\Sigma_1^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu}_1 - 2\boldsymbol{x}^T\Sigma_1^{-1}\boldsymbol{\mu}_1\right\}\Big]\Big] \\
&= \frac{1}{2}\Big[\boldsymbol{x}^T\left(\log\left(|\Sigma_2|^{-1/2}\right)\Sigma_2^{-1} - \log\left(|\Sigma_1|^{-1/2}\right)\Sigma_1^{-1}\right)\boldsymbol{x}\Big] - \boldsymbol{x}^T\left(\Sigma_1^{-1}\boldsymbol{\mu}_1 - \Sigma_2^{-1}\boldsymbol{\mu}_2\right) \\
&+ \frac{1}{2}\Big[\boldsymbol{\mu}_2^T\log\left(|\Sigma_2|^{-1/2}\right)\Sigma_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\log\left(|\Sigma_1|^{-1/2}\right)\Sigma_1^{-1}\boldsymbol{\mu}_1\Big].
\end{aligned}
$$

Thus we classify to class one when

$$
\begin{aligned}
&\frac{1}{2}\Big[\boldsymbol{x}^T\left(\log\left(|\Sigma_2|^{-1/2}\right)\Sigma_2^{-1} - \log\left(|\Sigma_1|^{-1/2}\right)\Sigma_1^{-1}\right)\boldsymbol{x}\Big] - \boldsymbol{x}^T\left(\Sigma_1^{-1}\boldsymbol{\mu}_1 - \Sigma_2^{-1}\boldsymbol{\mu}_2\right) \\
&+ \frac{1}{2}\Big[\boldsymbol{\mu}_2^T\log\left(|\Sigma_2|^{-1/2}\right)\Sigma_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\log\left(|\Sigma_1|^{-1/2}\right)\Sigma_1^{-1}\boldsymbol{\mu}_1\Big] \geq 0,
\end{aligned}
$$

and to class two otherwise.

Appendix D

Logistic regression

D.1 Logistic regression with all the gory details

In logistic regression each data point is assumed to be Bernoulli distributed

$$\mathcal{Y}_i | X_{i,:} \sim Bernouilli(p_i), \quad \mathcal{Y}_i \in \{0, 1\}.$$

We wish to model the probability p_i as a function of the explanatory variables $X_{i,:}$ such that we can make predictions about an unseen response based on $X_{i,:}$. Since probabilities are restricted to lie in $[0, 1]$, the model of p_i must use a function with range $[0, 1]$. The logistic function $g(x) = \frac{e^x}{1+e^x}$ satisfies this. The model of p_i, where $p_i = \mathbb{P}(\dagger_i = 1 | X_{i,:})$ is then

$$p_i(\boldsymbol{\beta}) = \frac{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^{m} \boldsymbol{\beta}_j X_{i,j})}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^{m} \boldsymbol{\beta}_j X_{i,j})}.$$

The estimation problem lies in finding the estimates of $\boldsymbol{\beta}_j, j = 0, 1, \ldots, k$. We do this by optimizing the likelihood-function. That is, we find the values of $\boldsymbol{\beta}_j, j =$

$0, 1, \ldots, k$ for which the observed data is most likely. The likelihood function is defined as the product of the probability densities of all observations (D.1).

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} f(\boldsymbol{y}_i|\boldsymbol{\beta}), \tag{D.1}$$

where $\boldsymbol{y}$ is the vector of observations. Since

$$\begin{aligned}
&\mathbb{P}(\mathcal{Y}_i = 1|X_i = x_i) = p_i \\
&\mathbb{P}(\mathcal{Y}_i = 0|X_i = x_i) = 1 - p_i \Rightarrow \\
&\qquad f(\boldsymbol{y}_i|\boldsymbol{\beta}) = p_i(\boldsymbol{\beta})^{\boldsymbol{y}_i}(1 - p_i(\boldsymbol{\beta}))^{1-\boldsymbol{y}_i},
\end{aligned}$$

the likelihood function becomes

$$\prod_{i=1}^{n} p_i(\boldsymbol{\beta})^{\boldsymbol{y}_i}(1 - p_i(\boldsymbol{\beta}))^{1-\boldsymbol{y}_i}.$$

The estimate of $\boldsymbol{\beta}$ is

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} p_i(\boldsymbol{\beta})^{\boldsymbol{y}_i}(1 - p_i(\boldsymbol{\beta}))^{1-\boldsymbol{y}_i} \\
&= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \log\left((p_i(\boldsymbol{\beta}))^{\boldsymbol{y}_i}(1 - p_i(\boldsymbol{\beta}))^{1-\boldsymbol{y}_i}\right) \\
&= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(\boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1 - \boldsymbol{y}_i)\log(1 - p_i(\boldsymbol{\beta}))\right).
\end{aligned} \tag{D.2}$$

This must be solved numerically. We used the function *glmfit* in MatLab, which fits generalized linear models. Logistic regression is obtained by specifying the binomial distribution for observations.

The observations were given prior weights according to their class, such that observations from larger classes had lower weights as explained in 5.4.3. The weights are re-calculated in each iteration of the numerical maximization of (D.2)

such that observations with higher estimated variance ($\hat{p}_i(1-\hat{p}_i)$) are weighted higher. The weight for the i^{th} observation in the s^{th} iteration when prior weights are not used is $(\hat{p}_i(1-\hat{p}_i))^s$. When prior weights $\boldsymbol{w}_i$ are used, the weight is $\boldsymbol{w}_i(\hat{p}_i(1-\hat{p}_i))^s$. The vector $\boldsymbol{w}$ contains the prior weights.

Another way to obtain weighted logistic regression is by using (D.3) instead of (D.2).

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \boldsymbol{w}_i \left(\boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1-\boldsymbol{y}_i) \log(1-p_i(\boldsymbol{\beta})) \right). \tag{D.3}$$

where $\boldsymbol{w}_i$ is the weight assigned to the i^{th} observation. This is a more direct way of taking weights into account. This was implemented by using the implementation of l1-regularized logistic regression described below, with λ set to 1.

D.1.1 Forward selection of parameters

It is likely that only some features are relevant in a classification problem. To increase interpretability of a model and reduce risk of over fitting, it may be wise to include only some features. In logistic regression, this can be done by starting with a model with not explanatory variables, and then adding those variables that seem to explain variability in data best. This is called forward selection. Two common alternatives are backward and stepwise selection. In backward selection, the starting point is a model that includes all variables. The variables that explain least variability are then removed iteratively. In stepwise selection, variables can both be added and removed in each iteration. Since these two alternatives are more computationally demanding, we chose to use forward selection.

Deviance is a measure of the degree to which a model agrees with data [58, p. 23]. The test statistic used to test whether a simpler model is significantly different from a larger model is the deviance of the larger model minus the deviance of the smaller model, which is assumed to be nested within the larger model. Under the null hypothesis, the deviance approximately follows the χ^2-distribution with degrees of freedom equal to the difference in parameters between the larger and the simpler model [62, p. 299]. Since we use forward selection, we attempt to add one variable at a time. This causes the deviance to follow the χ^2-distribution with one degree of freedom. Using the 5% significance level, we add the concerned variable if the test statistic is greater than $\chi^2_{0.05}(1) = 3.841$.

This corresponds to a p-value less than 0.05, which means that the deviance of the model with the concerned variable gives a significantly better fit to data than the model without the concerned variable, at the 5% significance level. The deviance is defined as (D.4)[58, p.23].

$$\begin{aligned} D(y, \boldsymbol{\beta}) &= -2\log(f(\boldsymbol{y}_i|\boldsymbol{\beta})) = -2\log\left((\prod_{i=1}^{n} p_i(\boldsymbol{\beta})^{\boldsymbol{y}_i}(1-p_i(\boldsymbol{\beta}))^{1-\boldsymbol{y}_i}\right) \\ &= -2\left((\sum_{i=1}^{n} \boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1-\boldsymbol{y}_i)\log((1-p_i(\boldsymbol{\beta})))\right) \\ &= 2\left((\sum_{i=1}^{n} \boldsymbol{y}_i \log(\frac{1}{p_i(\boldsymbol{\beta})}) + (1-\boldsymbol{y}_i)\log(\frac{1}{1-p_i(\boldsymbol{\beta}}))\right) \end{aligned} \tag{D.4}$$

Other authors [62, p. 299] define the deviance as the test statistic used to test whether a certain model is the true model (the null hypothesis), or the true model is the saturated model. This definition is shown in (D.5)

$$D_{rel}(Y, \boldsymbol{\beta}) = 2(\log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{saturated})) - \log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{model}))), \tag{D.5}$$

where $\boldsymbol{\beta}_{saturated}$ is the vector of parameters for the saturated model and $\boldsymbol{\beta}_{model}$ the vector of parameters for the model being tested. The saturated model is the model that includes all main and interaction effects between features [62, p.298]. By this definition, the deviance follows a χ^2 distribution with degrees of freedom equal to the number of parameters in the saturated model minus the number of parameters in the smaller model.

When testing a model against the saturated model these definitions of deviance give identical results. When comparing two arbitrary nested models, M_0 and M_1 defined above, by finding the differences in deviance according to definition (D.5), we get

$$\begin{aligned} &D_{rel}(Y, \boldsymbol{\beta}_{saturated}) - D_{rel}(Y, \boldsymbol{\beta}_{model}) = \\ &2(\log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{saturated})) - \log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{saturated}))) - (2(\log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{saturated})) - \log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{model})))) = \\ &2\left(\log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{model})) - \log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{saturated}))\right) = -2\log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{saturated})) - [-2\log(f(\boldsymbol{y}_i|\boldsymbol{\beta}_{model}))] = \\ &D_{abs}(Y, \boldsymbol{\beta}_{saturated}) - D_{abs}(Y, \boldsymbol{\beta}_{model}) \end{aligned}$$

Thus the test statistic to compare two models is the same for the two definitions. However, the interpretation of the measure for a single model differs. Whereas (D.5) can always be understood as the difference in explanatory ability between the model under consideration and the saturated model, the definition in (5.8) has no such interpretation.

D.1.2 L1-regularized logistic regression

Another way to limit the number of explanatory variables in a model is to penalize the cost likelihood function by the L1 norm of the coefficient vector. This also decreases the risk that coefficients of correlated explanatory features are estimated at high values in such a way that the effects of the variables cancel each other out in the model. We give a small example to make this scenario clearer. Assume x_1 and x_2 are highly correlated explanatory variables and that neither has any substantial influence on the response variable. Then a model in which x_1 has a very large coefficient $\boldsymbol{\beta}_1$ and the coefficient of x_2 is nearly the same with opposite sign $\boldsymbol{\beta}_2 \approx -\boldsymbol{\beta}_1$ will be nearly equivalent to a model with small $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$. The first model will give a wrong idea of the relation between explanatory variables and the response, might cause numerical instabilities if the coefficient estimates are of very high magnitudes, and high uncertainty will be associated with predictions. For these reasons a model with small or zero coefficients for x_1 and x_2 would be preferable.

Using the L1 norm of the coefficient vector as penalty, we need to maximize (D.2) plus the penalty, weighting both terms to reflect the degree to which we want a good fit relative to our desire for a simple model. We also take weights of observations into account as shown in (D.6).

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \boldsymbol{w}_i \left(\lambda \boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1 - \boldsymbol{y}_i) \log(1 - p_i(\boldsymbol{\beta})) + (1 - \lambda) \sum_{j=1}^{m} |\boldsymbol{\beta}_j| \right) \tag{D.6}$$

The parameter λ must lie between zero and one and determines the degree to which the coefficient vector should be simple relative to the ability of the model to fit data well. Higher values of λ cause the model to fit training data better at the cost of more non-zero estimates of the coefficients. The best value for λ can be found by cross-validation.

To find the coefficient estimates, we use a simple numerical optimization algorithm in which the first derivative of (D.6) with respect to the coefficients is

needed.

D.1.2.1 Derivatives of the log-likelihood

Using the chain rule for differentiation, we find the derivative (D.7). We set

$$f_i^{logistic\_log-likelihood} = \boldsymbol{y}_i \log(p_i(\boldsymbol{\beta})) + (1 - \boldsymbol{y}_i) \log(1 - p_i(\boldsymbol{\beta}))$$

and use that

$$\frac{\partial f_i^{logistic\_log-likelihood}}{\partial p_i(\boldsymbol{\beta})} = \boldsymbol{y}_i \frac{1}{p_i(\boldsymbol{\beta})} + (1-\boldsymbol{y}_i) \frac{-1}{1 - p_i(\boldsymbol{\beta})} = \boldsymbol{y}_i \frac{1}{p_i(\boldsymbol{\beta})} + (\boldsymbol{y}_i - 1) \frac{1}{1 - p_i(\boldsymbol{\beta})}.$$

The derivatives of the log-likelihood function with respect to the parameters are then

Derivative w.r.t. intercept:

$$\frac{\partial \sum_{i=1}^{n} \boldsymbol{w}_i \left(\lambda f_i^{logistic\_log-likelihood} - (1 - \lambda) \sum_{j=1}^{m} |\boldsymbol{\beta}_j| \right)}{\partial \boldsymbol{\beta}_0} = \sum_{i=1}^{n} \boldsymbol{w}_i \lambda \frac{\partial p_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_0} \left(\boldsymbol{y}_i \frac{1}{p_i(\boldsymbol{\beta})} + (\boldsymbol{y}_i - 1) \frac{1}{1 - p_i(\boldsymbol{\beta})} \right)$$

Derivative w.r.t. coefficients:

$$(h \in \{1, 2, \ldots, k\}) \qquad \frac{\partial \sum_{i=1}^{n} \boldsymbol{w}_i \left(\lambda f_i^{logistic\_log-likelihood} - (1 - \lambda) \sum_{j=1}^{m} |\boldsymbol{\beta}_j| \right)}{\partial \boldsymbol{\beta}_h} = \sum_{i=1}^{n} \boldsymbol{w}_i \left[\lambda \frac{\partial p_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_h} \left(\boldsymbol{y}_i \frac{1}{p_i(\boldsymbol{\beta})} + (\boldsymbol{y}_i - 1) \frac{1}{1 - p_i(\boldsymbol{\beta})} \right) - (1 - \lambda) \operatorname{sgn}(\boldsymbol{\beta}_h) \right], \tag{D.7}$$

and the derivatives of $p_i(\boldsymbol{\beta})$ are

$$\frac{\partial p_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_0} = \frac{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}{\left(1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})\right)^2} = \frac{p_i(\boldsymbol{\beta})}{(1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}$$
$$\frac{\partial p_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_h} = \frac{X_{i,h}\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}{\left(1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})\right)^2} = \frac{X_{i,h} p_i(\boldsymbol{\beta})}{(1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}.$$

Using that

$$\begin{aligned}
1 - p_i(\boldsymbol{\beta}) =& 1 - \frac{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} = \\
& \frac{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} - \frac{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} = \\
& \frac{1}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} = \frac{p_i(\boldsymbol{\beta})}{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})},
\end{aligned}$$

(D.7) can be simplified as follows

$$\begin{aligned}
&\sum_{i=1}^n \boldsymbol{w}_i \lambda \frac{\partial p_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_0} \left(\boldsymbol{y}_i \frac{1}{p_i(\boldsymbol{\beta})} + (\boldsymbol{y}_i - 1) \frac{1}{1 - p_i(\boldsymbol{\beta})} \right) = \\
&\sum_{i=1}^n \boldsymbol{w}_i \lambda \frac{p_i(\boldsymbol{\beta})}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} \left(\frac{\boldsymbol{y}_i}{p_i(\boldsymbol{\beta})} + (\boldsymbol{y}_i - 1) \frac{\exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})}{p_i(\boldsymbol{\beta})} \right) = \\
&\sum_{i=1}^n \boldsymbol{w}_i \lambda \frac{1}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} \left(\boldsymbol{y}_i + (\boldsymbol{y}_i - 1) \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j}) \right) = \\
&\sum_{i=1}^n \boldsymbol{w}_i \lambda \left(\frac{\boldsymbol{y}_i}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} + (\boldsymbol{y}_i - 1) p_i(\boldsymbol{\beta}) \right) = \\
&\sum_{i=1}^n \boldsymbol{w}_i \lambda \left(\frac{\boldsymbol{y}_i (1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j}))}{1 + \exp(\boldsymbol{\beta}_0 + \sum_{j=1}^m \boldsymbol{\beta}_j X_{i,j})} - p_i(\boldsymbol{\beta}) \right) = \\
&\sum_{i=1}^n \boldsymbol{w}_i \lambda \left(\boldsymbol{y}_i - p_i(\boldsymbol{\beta}) \right).
\end{aligned}$$

Thus we get the final expressions for the derivatives as shown in (D.8)

$$\begin{aligned}
&\frac{\partial \sum_{i=1}^{n} \boldsymbol{w}_i \left(\lambda f_i^{logistic_log-likelihood} - (1-\lambda)\sum_{j=1}^{m} |\boldsymbol{\beta}_j|\right)}{\partial \boldsymbol{\beta}_0} = \\
&\sum_{i=1}^{n} \boldsymbol{w}_i \lambda \left(\boldsymbol{y}_i - p_i(\boldsymbol{\beta})\right) \\
&\frac{\partial \sum_{i=1}^{n} \boldsymbol{w}_i \left(\lambda f_i^{logistic_log-likelihood} - (1-\lambda)\sum_{j=1}^{m} |\boldsymbol{\beta}_j|\right)}{\partial \boldsymbol{\beta}_h} = \\
&\sum_{i=1}^{n} \boldsymbol{w}_i \left[\lambda X_{i,h} \left(\boldsymbol{y}_i - p_i(\boldsymbol{\beta})\right) - (1-\lambda)\,\mathrm{sgn}(\boldsymbol{\beta}_h)\right].
\end{aligned} \tag{D.8}$$

D.1.2.2 Finding the parameter estimates

We take advantage of the fact that maximizing the log-likelihood (D.6) is equivalent to minimizing the negative log-likelihood (D.9)

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \boldsymbol{w}_i \left[\lambda f_i^{logistic_log-likelihood} - (1-\lambda) \sum_{j=1}^{m} |\boldsymbol{\beta}_j| \right] \right\} \\
&= \arg\min_{\boldsymbol{\beta}} \left\{ - \sum_{i=1}^{n} \boldsymbol{w}_i \left[\lambda f_i^{logistic_log-likelihood} - (1-\lambda) \sum_{j=1}^{m} |\boldsymbol{\beta}_j| \right] \right\}.
\end{aligned} \tag{D.9}$$

To find the coefficient estimates, we start with an initial random guess. We then change this guess by moving a step size μ in the direction opposite to the gradient (D.10).

$$\boldsymbol{\beta}_h^{new\_simple} = \boldsymbol{\beta}_h - \mu \cdot \sum_{i=1}^{n} -\boldsymbol{w}_i \left[\lambda X_{i,h} \left(\boldsymbol{y}_i - p_i(\boldsymbol{\beta})\right) - (1-\lambda)\,\mathrm{sgn}(\boldsymbol{\beta}_h)\right], \tag{D.10}$$

where we use the gradient that is negative of that shown in (D.8) since (D.8) is the gradient relevant to the maximization problem (D.6), but we wish to solve the minimization problem (D.9). We set $X_{i,0}$ to one for all i, and $\text{sgn}(\boldsymbol{\beta}_0)$ to zero.

To avoid taking too large steps, thereby risking missing the minimum, or taking too small steps, which would increase the time to convergence, we adapt the step size μ in each iteration. Also, we only take a step if the step causes the cost function to decrease. If this is not the case, we decrease the step size but do not change the coefficient and bias estimates. If the cost function does decrease, we update the estimates as shown in (D.10) and increase μ by multiplying the former value by 1.5 and using this new value for μ.

Also, instead of updating the coefficients exactly as in (D.10), we split the update into two steps. This split ensures that the coefficient estimates do not oscillate around zero. We do this by first updating according to the non-penalty part of the gradient as follows (D.11)

$$\boldsymbol{\beta}_h^{new\_temporary} = \boldsymbol{\beta}_h + \mu \cdot \sum_{i=1}^{n} \boldsymbol{w}_i \lambda X_{i,h} \left(\boldsymbol{y}_i - p_i(\boldsymbol{\beta}) \right). \tag{D.11}$$

We then update according to the penalty as shown in (D.12)

$$\boldsymbol{\beta}_h^{new\_penalized\_temporary} = \boldsymbol{\beta}_h^{new\_temporary} - \mu \cdot (1 - \lambda) \, \text{sgn}(\boldsymbol{\beta}_h), \text{for } h \in \{1, 2, \ldots, k\}. \tag{D.12}$$

The penalized update step is not performed for the bias $\boldsymbol{\beta}_0$. Now comes the step that protects from oscillation of estimates around zero

$$\boldsymbol{\beta}_h^{new\_guess} = \begin{cases} \boldsymbol{\beta}_h^{new\_penalized\_temporary} & \text{sgn}(\boldsymbol{\beta}_h^{new\_penalized\_temporary}) = \text{sgn}(\boldsymbol{\beta}_h^{new\_temporary}) \\ 0 & \text{sgn}(\boldsymbol{\beta}_h^{new\_penalized\_temporary}) \neq \text{sgn}(\boldsymbol{\beta}_h^{new\_temporary}) \end{cases} \tag{D.13}$$

where $h \in \{1, 2, \ldots, k\}$. Finally, the coefficient estimates are updated if the cost function decreased. To sum up, the algorithm is shown in Algorithm 1 on page 67. For ease of reading, details have been left out. These can be found in Algorithm 2 on page 165.

D.1.3 Convexity

We now show that the minimization problem in logistic regression is convex, both with and without L1-regularization. Convexity is desirable property because it ensures the uniqueness of a solution. Also, research into optimization of convex functions has a long history and easily implementable algorithms with linear and quadratic convergence are well known.

A twice differentiable function is convex if and only if its Hessian is positive semi-definite [3, p. 198]. Thus we need to show that the Hessian of the function being minimized in logistic regression is positive semi-definite. We have already found the gradient (D.7), so we can get the Hessian by differentiating this with respect to each of the variables $\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k$.

The $(s,t)^{th}$ entry in the Hessian of a function f is the partial derivative of f with respect to the s^{th} variable, which is then differentiated with respect to the t^{th} variable. The second order partial derivatives of $f_i^{logistic\_log-likelihood}$ are given in (D.14), calculated using the chain and product rules for differentiation

$$\frac{\partial\partial\left\{-\boldsymbol{w}_i\lambda f_i^{logistic\_log-likelihood}\right\}}{\partial\boldsymbol{\beta}_s\partial\boldsymbol{\beta}_t} = \frac{\partial\left\{-\boldsymbol{w}_i\lambda\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_s}\left(\boldsymbol{y}_i\frac{1}{p_i(\boldsymbol{\beta})}+(\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)\right\}}{\partial\boldsymbol{\beta}_t} = \\ -\boldsymbol{w}_i\lambda\Big[\frac{\partial\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_s\partial\boldsymbol{\beta}_t}\left(\boldsymbol{y}_i\frac{1}{p_i(\boldsymbol{\beta})}+(\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+ \\ \frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_s}\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_t}\left(\boldsymbol{y}_i\frac{(-1)}{(p_i(\boldsymbol{\beta}))^2}+(\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\Big]. \tag{D.14}$$

To see that the Hessian is positive semi-definite, we continue evaluation of the second order partial derivatives.

$$
\begin{aligned}
&- \boldsymbol{w}_i\lambda\Big[\frac{\partial\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_s\partial\boldsymbol{\beta}_t}\left(\boldsymbol{y}_i\frac{1}{p_i(\boldsymbol{\beta})}+(\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_s}\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_t}\left(\boldsymbol{y}_i\frac{(-1)}{(p_i(\boldsymbol{\beta}))^2}+(\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\\
&- \boldsymbol{w}_i\lambda\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_t}\left[X_{i,s}\left(\boldsymbol{y}_i\frac{1}{p_i(\boldsymbol{\beta})}+(\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_s}\left(\boldsymbol{y}_i\frac{(-1)}{(p_i(\boldsymbol{\beta}))^2}+(\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right.\\
&- \boldsymbol{w}_i\lambda\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_t}X_{i,s}\left[\left(\boldsymbol{y}_i\frac{1}{p_i(\boldsymbol{\beta})}+(\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left(\boldsymbol{y}_i\frac{(-1)}{(p_i(\boldsymbol{\beta}))^2}+(\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right.\\
&- \boldsymbol{w}_i\lambda\frac{\partial p_i(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_t}X_{i,s}\left[\left((\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left((\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right]=\\
&- \boldsymbol{w}_i\lambda X_{i,s}X_{i,t}p_i(\boldsymbol{\beta})\left[\left((\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left((\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right],
\end{aligned}
$$

where $X_{i,0}=1$ for all i. Since

$$
\begin{aligned}
&\boldsymbol{y}_i=1\Rightarrow\left[\left((\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left((\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right]=0,\\
&\boldsymbol{y}_i=0\Rightarrow\left[\left((\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left((\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right]=-\frac{1}{1-p_i(\boldsymbol{\beta})}\left(1+p_i(\boldsymbol{\beta})\frac{1}{1-p_i}\right.
\end{aligned}
$$

and $-\boldsymbol{w}_i\lambda p_i(\boldsymbol{\beta})<0$, the product of $-\boldsymbol{w}_i\lambda p_i(\boldsymbol{\beta})$ and $\left[\left((\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left((\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right.\right.$ is non-negative. Thus we may set

$$
q_{i,j}=X_{i,j}\sqrt{-\boldsymbol{w}_i\lambda p_i(\boldsymbol{\beta})\left[\left((\boldsymbol{y}_i-1)\frac{1}{1-p_i(\boldsymbol{\beta})}\right)+p_i(\boldsymbol{\beta})\left((\boldsymbol{y}_i-1)\frac{1}{(1-p_i(\boldsymbol{\beta}))^2}\right)\right]},
$$

and $\boldsymbol{q_i}=(q_{i,0},q_{i,1},\ldots,q_{i,k})$. We can then express the Hessian of a single observation as

$$
H_i=\boldsymbol{q_i}\boldsymbol{q_i}^T.
$$

The Hessian for all observations is then

$$
H=\sum_{i=1}^{n}H_i=\sum_{i=1}^{n}\boldsymbol{q_i}\boldsymbol{q_i}^T,
$$

and by left- and right-multiplying an arbitrary vector $\boldsymbol{z}$ onto H, we get

$$\boldsymbol{z}^T H \boldsymbol{z} = \boldsymbol{z}^T \left(\sum_{i=1}^{n} H_i \right) \boldsymbol{z} = \sum_{i=1}^{n} \boldsymbol{z}^T \left(H_i \right) \boldsymbol{z} = \sum_{i=1}^{n} \boldsymbol{z}^T \boldsymbol{q_i} \boldsymbol{q_i}^T \boldsymbol{z} = \sum_{i=1}^{n} (\boldsymbol{z}^T \boldsymbol{q_i})^2 \geq 0,$$

Showing that the Hessian is positive semi-definite and thus that the cost function being minimized is convex.

D.2 Variance of coefficient estimates for logistic regression

```
fitted = ml_predict(feats, model);
fittedprobs=fitted{2};
fits = fittedprobs(:,1); % extract probabilities for membership of
    class 1
fits(fits > (1-eps*100)) = 1-eps*100; % cut off observations at
    extremes of support
fits(fits < (eps*100)) =   eps*100;

var_obs_temp=fits.*(1-fits);
var_obs=var_obs_temp;
var_pearson_obs=obsweights'.*((train_targets'-fits).^2)./var_obs;
var_pearson = sum(var_pearson_obs)/(length(obsweights)-44); % -43 for
    the 43 features and minus 1 for the intercept

Xmat = [ones(length(obsweights), 1) feats];

diagonal = obsweights'.*var_obs;
Sigmainv = Xmat' *diag(diagonal) *Xmat;
Sigma=inv(Sigmainv);

mean_ests = [model.model.b model.model.W];
var_ests = var_pearson^2*diag(Sigma)';
```

Algorithm 2 Numerical minimization for L1-regularized logistic regression to find coefficient estimates

$\boldsymbol{\beta} \leftarrow \mathcal{N}_{m+1}(\mathbf{0}_{m+1}, I_{m+1})$ {initialize coefficient estimates}
max_eval $\leftarrow 100$ {stopping criterion; maximum number of iterations}
$\mu \leftarrow 1$, its $\leftarrow 0$, nconverged_parameters $\leftarrow 0$ {initialize variables}
param_conv $\leftarrow 0$ {initialize parameter convergence to zero to enter loop}
tolx $\leftarrow 10^{-3}$ {amount by which all parameters must change less than to obtain convergence}
exp_term $\leftarrow \exp\left(\boldsymbol{\beta}_0 + \sum_{j=1}^{m} \boldsymbol{\beta}_j X_{i,j}\right)$
cost_current $\leftarrow \sum_{i=0}^{n-1} -\boldsymbol{w}_i \left[\lambda \left[\boldsymbol{y}_i \left(\boldsymbol{\beta}_0 + \sum_{j=1}^{m} \boldsymbol{\beta}_j X_{i,j}\right) - \log\left(1 + \text{exp\_term}\right)\right] - (1-\lambda) \sum_{j=1}^{m} |$

while its $\leq$ max_eval $\wedge$ nconverged_parameters $<$ m+1 **do**
  for $i = 0 \rightarrow n-1$ **do**
    $J_{i,1} \leftarrow -\lambda \boldsymbol{w}_i \boldsymbol{y}_i \frac{\text{exp\_term}}{1+\text{exp\_term}}$ {the i^{th} row in the first column of J contains derivatives of the non-penalized cost function with respect to the intercept, evaluated at the data point i}
    for $j = 1 \rightarrow m$ **do**
      $J_{i,j} \leftarrow -\lambda \boldsymbol{w}_i \left(\boldsymbol{y}_i X_{i,j} - X_{i,j} \frac{\text{exp\_term}}{1+\text{exp\_term}}\right)$ {the i^{th} row in the j^{th} column of J contains the derivatives of the non-penalized cost function with respect to the j^{th} variable at each data point}
    end for
  end for
  for $i = 0 \rightarrow n-1$ **do**
    for $j = 1 \rightarrow m$ **do**
      $J\_penal_{i,j} \leftarrow -(1-\lambda) \boldsymbol{w}_i \operatorname{sgn}(\boldsymbol{\beta}_{j+1})$ {the i^{th} row in the j^{th} column of J_penal contains derivatives of the penalty on the cost function with respect to the j^{th} variable. }
    end for
  end for
  J_penal $\leftarrow$ -J_penal {execute non-penalized updates}
  for $j = 0 \rightarrow m$ **do**
    $\boldsymbol{\beta}_j^{new\_temporary} \leftarrow \boldsymbol{\beta}_j - \mu \cdot \sum_{i=0}^{n-1} J_{i,j}$
  end for
  {execute penalized updates, which the intercept estimate is not affected by}
  for $j = 1 \rightarrow m$ **do**
    $\boldsymbol{\beta}_j^{new\_penalized\_temporary} \leftarrow \boldsymbol{\beta}_j^{new\_temporary} - \mu \cdot \sum_{i=0}^{n-1} J\_penal_{i,j}$
  end for
  {set estimate to zero if zero was crossed in the penalty update}
  for $j = 1 \rightarrow m$ **do**
    if $\operatorname{sgn}(\boldsymbol{\beta}_j^{new\_temporary}) \neq \operatorname{sgn}(\boldsymbol{\beta}_j^{new\_penalized\_temporary})$ **then**
      $\boldsymbol{\beta}_j^{new} \leftarrow 0$
    else
      $\boldsymbol{\beta}_j^{new} \leftarrow \boldsymbol{\beta}_j^{new\_penalized\_temporary}$
    end if
  end for
  $\boldsymbol{\beta}_0^{new} \leftarrow \boldsymbol{\beta}_0^{new\_temporary}$
  exp_term_new $\leftarrow \exp\left(\boldsymbol{\beta}_0^{new} + \sum_{j=1}^{m} \boldsymbol{\beta}_j^{new} X_{i,j}\right)$

Appendix E

Error propagation

An approximation of the variance of a function of several random variables with known variances can be achieved with the rule of error propagation [54].This rule uses the first order Taylor approximation to a function by using the variance of this approximation. First we briefly describe the first order Taylor approximation. We then proceed to find the expectation and variance of the first order Taylor approximation of a function of random variables.

Taylor approximation consists of using the first derivative of a function in approximating the value of that function at a particular value. First we describe the first order Taylor expansion.

Take a univariate function f. The first order Taylor expansion is then [45]

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + O((x - x_0)^2),$$

where x_0 is the point that we approximate around. Thus the approximation is best at points x close to x_0. We use the notation $f'(x_0)$ to mean the derivative of f evaluated at x_0. Finally, $O((x - x_0)^2)$ denotes the order of growth of error in the approximation as a function of the difference between x and x_0. If f is a multivariate function, the first order Taylor expansion becomes

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \sum_{j=1}^{p} \left(f'(\boldsymbol{x}_{0,j})(\boldsymbol{x}_j - \boldsymbol{x}_{0,j}) + O((\boldsymbol{x}_j - \boldsymbol{x}_{0,j})^2) \right),$$

where we let $f'(\boldsymbol{x}_{0,j})$ denote the derivative of f with respect to the j^{th} coordinate, evaluated at the j^{th} coordinate of $\boldsymbol{x}_0$, $\boldsymbol{x}_{0,j}$. As $\boldsymbol{x} \to \boldsymbol{x}_0$, $\sum_{j=1}^{p} O((\boldsymbol{x}_j - \boldsymbol{x}_{0,j})^2) \to 0$ i.e. the error term vanishes.

To obtain the first order Taylor approximation, we drop the error term. The first order Taylor approximation then becomes (E.1).

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \sum_{j=1}^{p} f'(\boldsymbol{x}_{0,j})(\boldsymbol{x}_j - \boldsymbol{x}_{0,j}). \tag{E.1}$$

Estimates using the first order Taylor approximation We first find the expectation of the first order Taylor approximation of a function of random variables. Let X denote a random vector. Expanding around the mean of X, μ, the first order Taylor approximation becomes

$$f(X) \approx \hat{f}(X) = f(\mu) + \sum_{j=1}^{p} f'(\mu_j)(X_j - \mu_j).$$

The expectation of the approximation to f is then

$$\begin{aligned} E(f(X)) \approx E(\hat{f}(X)) &= E\left(f(\mu) + \sum_{j=1}^{p} f'(\mu_j)(X_j - \mu_j) \right) \\ &= f(\mu) + \sum_{j=1}^{p} E\left(f'(\mu_j)(X_j - \mu_j) \right) \\ &= f(\mu) + \sum_{j=1}^{p} f'(\mu_j)(E\left(X_j\right) - \mu_j) \\ &= f(\mu) + \sum_{j=1}^{p} f'(\mu_j)(\mu_j - \mu_j) = f(\mu). \end{aligned}$$

The variance is

$$Var(f(X)) \approx Var(\hat{f}(X)) = Var\left(f(\mu) + \sum_{j=1}^{p} f'(\mu_j)(X_j - \mu_j)\right) = Var\left(\sum_{j=1}^{p} f'(\mu_j)(X_j - \mu_j)\right)$$
$$= \sum_{j=1}^{p} f'(\mu_j)^2 Var((X_j) + 2\sum_{j=1}^{p}\sum_{h=j+1}^{p} f'(\mu_j)f'(\mu_h)Cov(X_j, X_h).$$

If the coordinates of X are pairwise uncorrelated, this simplifies to

$$Var(f(X)) \approx Var(\hat{f}(X)) = \sum_{j=1}^{p} f'(\mu_j)^2 Var((X_j).$$

Appendix F

Factorization of data to estimate missing values

Steepest gradient descent to estimate factors of data matrix To reconstruct the missing values based on the non-missing values from the other channels, a factorization of the data can be used. We assume

$$Y = AS.$$

The challenge is then to find A and S such that $\|A \times S - Y\|_2$ is minimized. We take missing values into account by introducing the matrix W, whose $(i,j)^{th}$ element is zero if the $Y_{i,j}$ is missing, and one otherwise. We disregard missing values while finding A and S by minimizing

$$\sum_{i,j} W_{i,j}(Y_{i,j} - \sum_{h} A_{i,h}S_{h,j})^2.$$

We optimize using gradient descent, alternately updating A and S. Thus we must partially differentiate 6.1 with respect to $A_{q,r}$ and $S_{q,r}$. For $A_{q,r}$, we get

$$
\begin{aligned}
\frac{\partial \sum_i \sum_j W_{i,j}(Y_{i,j} - \sum_h A_{i,h}S_{h,j})^2}{\partial A_{q,r}} &= \frac{\partial \sum_i \sum_j W_{i,j}(Y_{i,j}^2 + (\sum_h A_{i,h}S_{h,j})^2 - 2Y_{i,j}\sum_h A_{i,h}S_{h,j})}{\partial A_{q,r}} \\
&= \frac{\partial \sum_{i,j} W_{i,j}((\sum_h A_{i,h}S_{h,j})^2 - 2Y_{i,j}\sum_h A_{i,h}S_{h,j})}{\partial A_{q,r}} \\
&= \frac{\partial \sum_{i,j} W_{i,j}(\sum_h A_{i,h}S_{h,j})^2}{\partial A_{q,r}} - \frac{\partial \sum_{i,j} W_{i,j}(2Y_{i,j}\sum_h A_{i,h}S_{h,j})}{\partial A_{q,r}} \\
&= \sum_{i,j} W_{i,j}\left(\frac{\partial(\sum_h A_{i,h}S_{h,j})^2}{\partial \sum_h A_{i,h}S_{h,j}} \cdot \frac{\partial \sum_h A_{i,h}S_{h,j}}{\partial A_{q,r}}\right) - 2\sum_j W_{q,j}Y_{q,j}S_{r,} \\
&= \sum_{i,j} W_{i,j}\left(2\left(\sum_h A_{i,h}S_{h,j}\right) \cdot \frac{\partial \sum_h A_{i,h}S_{h,j}}{\partial A_{q,r}}\right) - 2\sum_j W_{q,j}Y_{q,j}S_r \\
&= \sum_j W_{q,j}2\left(\sum_h A_{q,h}S_{h,j}\right)S_{r,j} - 2\sum_j W_{q,j}Y_{q,j}S_{r,j} \\
&= 2\sum_j W_{q,j}\left(\left(\sum_h A_{q,h}S_{h,j}\right)S_{r,j} - Y_{q,j}S_{r,j}\right) \\
&= 2\sum_j W_{q,j}S_{r,j}\left(\left(\sum_h A_{q,h}S_{h,j}\right) - Y_{q,j}\right).
\end{aligned}
\tag{F.1}
$$

Thus the $(q,r)^{th}$ element of the gradient of A, G^A is calculated as shown in the last line ofF.1. To find the elements of the gradient of S, we proceed similarly by differentiating with respect to $S_{q,r}$

$$
\begin{aligned}
\frac{\partial \sum_i \sum_j W_{i,j}(Y_{i,j} - \sum_h A_{i,h}S_{h,j})^2}{\partial S_{q,r}} &= \frac{\partial \sum_i \sum_j W_{i,j}(Y_{i,j}^2 + (\sum_h A_{i,h}S_{h,j})^2 - 2Y_{i,j}\sum_h A_{i,h}S_{h,j})}{\partial S_{q,r}} \\
&= \frac{\partial \sum_{i,j} W_{i,j}((\sum_h A_{i,h}S_{h,j})^2 - 2Y_{i,j}\sum_h A_{i,h}S_{h,j})}{\partial S_{q,r}} \\
&= \frac{\partial \sum_{i,j} W_{i,j}(\sum_h A_{i,h}S_{h,j})^2}{\partial S_{q,r}} - \frac{\partial \sum_{i,j} W_{i,j}(2Y_{i,j}\sum_h A_{i,h}S_{h,j})}{\partial S_{q,r}} \\
&= \sum_{i,j} W_{i,j}\left(\frac{\partial(\sum_h A_{i,h}S_{h,j})^2}{\partial \sum_h A_{i,h}S_{h,j}} \cdot \frac{\partial \sum_h A_{i,h}S_{h,j}}{\partial S_{q,r}}\right) - 2\sum_i W_{i,r}Y_{i,} \\
&= \sum_{i,j} W_{i,j}\left(2\left(\sum_h A_{i,h}S_{h,j}\right) \cdot \frac{\partial \sum_h A_{i,h}S_{h,j}}{\partial S_{q,r}}\right) - 2\sum_i W_{i,r}Y_i \\
&= \sum_j W_{i,r}2\left(\sum_h A_{q,h}S_{h,j}\right)A_{i,q} - 2\sum_i W_{i,r}Y_{i,r}A_{i,q} \\
&= 2\sum_j W_{i,r}\left(\left(\sum_h A_{q,h}S_{h,j}\right)A_{i,q} - Y_{i,r}A_{i,q}\right) \\
&= 2\sum_j W_{i,r}A_{i,q}\left(\left(\sum_h A_{q,h}S_{h,j}\right) - Y_{i,r}\right).
\end{aligned}
$$

With these gradients, steep gradient descent can be used to find the matrices A and S.

To save computations, if the estimated data is to be IC decomposed, an SVD of each of A and S can be performed. Since these matrices are smaller, it is cheaper to pre-whiten data by pre-whitening each of A and S, and then multiplying, instead of pre-whitening the product of A and S.

We now perform a singular value decomposition (SVD) on A and S such that we end up with an SVD of the original data, with reconstructed missing values. We factorize A and S as follows

$$
\begin{aligned}
A &= U^A\Sigma^A(V^A)^T \\
S &= U^S\Sigma^S(V^S)^T,
\end{aligned}
$$

where U^A, V^A, U^S, and V^S are unitary matrices and Σ^A and Σ^S are $k \times m$ and $m \times n$ diagonal matrices, respectively. Thus $U^A(U^A)^T = I_{k,k}$, $V^A(V^A)^T =$

$I_{m,m}$, $U^S(U^S)^T = I_{m,m}$, and $V^S(V^S)^T = I_{n,n}$. $m \leq n$ is the number of dimensions used to reconstruct the missing values. We now set

$$Q = \Sigma^A(V^A)^T U^S \Sigma^S,$$

and find the SVD decomposition of the $k \times n$ matrix Q

$$Q = U^Q \Sigma^Q (V^Q)^T.$$

The SVD decomposition of Y, through which we reconstruct missing values, is then

$$\begin{aligned} U &= U^A U^Q \\ V^T &= (V^Q)^T (V^S)^T \\ \Sigma &= \Sigma^Q, \end{aligned}$$

since

$$\begin{aligned} Y = AS = U^A \Sigma^A (V^A)^T U^S \Sigma^S (V^S)^T = U^A Q (V^S)^T \\ = U^A U^Q \Sigma^Q (V^Q)^T (V^S)^T = U \Sigma V^T. \end{aligned}$$

We can check that U and V are unitary, as required in an SVD decomposition

$$\begin{aligned} UU^T = U^A U^Q (U^A U^Q)^T = U^A U^Q (U^Q)^T (U^A)^T = U^A I_{k,k} (U^A)^T = I_{k,k} \\ VV^T = V^S V^Q (V^Q)^T (V^S)^T = V^S I_{n,n} (V^S)^T = I_{n,n}, \end{aligned}$$

as desired.

Appendix G

Implementation

All numerical calculations were performed in MatLab. The toolboxes and stand-alone code that was used, but implemented by others, are listed here.

- EEGLab [14]
- BCILab [16]
- immoptibox [43]
- MatLab official Stats toolbox
- MatLab official Wavelets toolbox
- ADJUST [41]. The methods that calculate features were modified to allow for differences in the number of channels and ICs. The code was also changed so feature calculations that assume several epochs in data made sense for continuous data sets, by splitting such data into smaller intervals for the purpose of calculating these features.
- Reconstruction of data, SVDMissingData.m, by Morten Mørup

Appendix H

Current density norm, the left out BBCI feature

The current density norm measures the complexity of the dipole fit. For artifacts, this complexity should be high since it should be difficult to match artifactual activity to a source in the brain. A mathematical description follows.

Let $\boldsymbol{z}$ be a vector denoting the amplitude of M dipoles at fixed locations and orientations in the brain. Let $\boldsymbol{a}_j$ be the spatial pattern of the j^{th} IC, i.e. the vector of length k that holds the activation of each of the k electrodes for component j. Finally, let F be the $k \times M$ lead-field matrix which contains information on the conductive properties of brain tissue. That is, F describes the activity caused by each dipole in $\boldsymbol{z}$ on the scalp. Thus $\boldsymbol{z}$ must satisfy the equation

$$\boldsymbol{a}_j = Fz \tag{H.1}$$

The $\boldsymbol{z}$ that minimizes the squared error $\|Fz - \boldsymbol{a}_j\|_2^2$ subject to the simultaneous minimization of $\lambda\|\Gamma\boldsymbol{z}\|_2^2$ can be expressed as

$$(F^TF + \lambda\Gamma^T\Gamma)^{-1}F^T\boldsymbol{a}_j = J_\lambda\boldsymbol{a}_j,$$

where Γ is a weight matrix that reduces the bias that would otherwise cause superficial dipoles [35] (i.e. dipoles close to the scalp) to be over estimated, implying a "tendency to reconstruct superficial currents" [35]. Γ is a diagonal matrix, where the diagonal *diag* is calculated as

$colnorms = (\|F_{.1}\|_2, \|F_{.2}\|_2, \ldots, \|F_{.M}\|_2)$ Where $F_{.h}$ indicates the h^{th} column of F

$$diag = \frac{colnorms.^{2/p} + \beta^2}{colnorms.^{1/p}},$$

where . indicates element wise operation, and the fraction is also element wise. Also, β is the 90^{th} percentile of $colnorms.^{1/p}$ divided by the signal to noise ratio(SNR).

The number

$$\log\left(J_\lambda \frac{\boldsymbol{a}_j}{\|\boldsymbol{a}_j\|_2}\right)$$

quantifies the complexity of the fit, i.e. the complexity of the $\boldsymbol{z}$ that comes closest to satisfying (H.1). A more complex $\boldsymbol{z}$ is a vector with many non-zero non-elements, implying that many dipole sources are necessary to explain the electrical activity on the scalp. For artifact components with dipoles outside the scalp, this fit will necessarily be complex, and hence $\log\left(J_\lambda \frac{\boldsymbol{a}_j}{\|\boldsymbol{a}_j\|_2}\right)$ will be large in such cases. However, the dipole fits provided by EEGLab assume that just one, or two symmetrical with respect to the midline, dipoles explain the scalp activity for a neural component. Thus only one (or two symmetrical) dipoles are fitted, making the current density norm meaningless.

Appendix I

Feature quantifying similarity to ECG time series

In previous attempts at component classification, components representing the ECG turned out to be the most frequently misclassified components. Thus we introduce a feature specifically aimed at detecting ECG components by exploiting the shape of the QRS complex. The QRS complex is a pattern that occurs once in every heart beat, and is the part of the heart's cycle with the largest amplitude. Thus components representing the pulse are likely to exhibit patterns similar to the QRS complex at regular intervals.

Discrete wavelet analysis (see section I.1 for background on wavelet analysis) was used to search for the QRS complex.

In order to detect the QRS complex, we must have an idea of the time scale that should be searched. In normal ECG, the QRS complex lasts from 0.06 to 0.10 seconds and the number of heartbeats per minute in humans is 60 to 90 [65], the average of which is 75. The frequency with which QRS complexes occur is then approximately $\frac{1}{0.10} - \frac{1}{0.06} \approx 10Hz - 17Hz$. The average duration of a heartbeat is approximately $\frac{75 \text{ heart beats}}{60\text{seconds}} \approx 1.25$ seconds. The length of the interval between two consecutive QRS complexes is then approximately 1.2 seconds, since this is

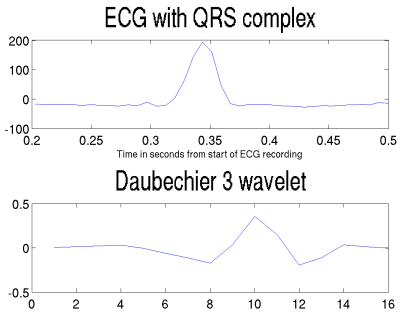

Figure I.1: A typical QRS complex and the Daubechies 3 wavelet. The Daubechies wavelet was chosen because of the similarities in shape between this wavelet and the QRS complex.

the time between two heart beats minus the average duration of a QRS complex ($1.25 - 0.08 = 1.17 \approx 1.2$).

A wavelet analysis of a time series using a wavelet resembling the QRS complex should give large reconstruction coefficients. These coefficients can then be used to find the time series that contain QRS complexes. Thus we use discrete wavelet analysis to make the largest peak in a time interval clearer, while flattening out smaller peaks.

Wavelets of the type Daubechies 3 were chosen due to the similarity between the shapes of these and that of the QRS complex I.1, and because others have previously used Daubechies 3 wavelets to detect ECG in EEG [51].

To allow identification of the QRS pattern, the wavelet must be scaled similarly. We use discrete wavelet analysis for computational efficiency, so we must choose a scale that is a power of two. To find the correct power, we solve $srate/(2^x) = 15$ for x since we expect the frequency of QRS complexes to occur with frequency approximately 15Hz. To make the subsequent analyses, based on these approximative calculations, more robust we do three parallel analyses at levels $\lfloor x \rfloor - 1$, $\lfloor x \rfloor$, and $\lfloor x \rfloor + 1$.

We then quantify the degree of ECG behavior at each of the decomposition levels $\lfloor x \rfloor - 1$, $\lfloor x \rfloor$, and $\lfloor x \rfloor + 1$. For each level, we find the approximation coefficients. These coefficients are then split into windows. Each window has length 1.1 seconds, and there is a space of 0.1 seconds between the end of one window and the beginning of the next such that the same peak is not detected in two separate windows, which would make one peak seem as two. The maximal

and minimal values in each window is found, and we store indices in the time series at which these values were found. Thus we get two vectors, one containing the indices of the local maximal values, and one containing the indices of the local minimal values, where the indices are with respect to the entire time series. We keep track of both the largest positive and negative values since ICs can only be determined up to sign, which means that the ECG component may either have large positive or large negative peaks when the QRS occurs.

Before doing further processing, we find the vector that shows the closest resemblance to the expected 1.2 second interval between QRS complexes. The interval between two maximal values is found by subtracting the previous index from each index, and dividing these differences by the sampling rate to get the interval lengths in seconds. We then find the mean, the variance, and the fifth and 95th percentiles of intervals between peaks. These four values are the features that we use to quantify the resemblance of the time series to the ECG. Since these operations are done at three scales, we get 12 features in total.

I.1 Discrete wavelet analysis

Wavelet analysis is a way to split a signal into two parts, one of which contains the large scale trends while the other gives the smaller variations around the overall trends. In general, the techniques can be applied to both 1D and 2D signals. However, we will only need the 1D case and thus focus on this. Also, wavelet analysis can be performed in both a discrete and a continuous form. Since the discrete form allows for much faster computations, we use this. Hence this discussion is limited to discrete 1D wavelet analysis.

The large scale trend in a signal can be found by averaging over a few consecutive values in a moving window. Conversely, the small scale fluctuations may be expressed as differences between consecutive values. By repeatedly taking averages and differences of values in the previous large scale trend, both trends and small fluctuations may be obtained at several levels. At higher levels, i.e. after several repetitions of the process, larger large scale trends result since averages are taken over many values of the original signal. We now make this more concrete.

Assume we have a signal $\boldsymbol{f} = (f_1, f_2, f_3, \ldots, f_n)$, where n is an even number. If the original signal does not have an even number of values, we make it even by appending zeros [1]. The moving window, in which we take averages and differences, is moved forward by two values between calculating each average (or difference). Thus the resulting trend and fluctuation each have length $\frac{n}{2}$.

We denote the trend by $\boldsymbol{a}^1 = (a_1^1, a_2^1, \ldots, a_{n/2}^2)$ and the fluctuation, or detail, by $\boldsymbol{d}^1 = (d_1^1, d_2^1, \ldots, d_{n/2}^2)$. The superscripts 1 signify that these are first level trend and detail coefficients, i.e. the moving averages and differences of the original signal. We can find the coefficient a_h^1 by taking scalar products between $\boldsymbol{f}$ and a scaling signal $\boldsymbol{v}_h^1$ to find the trend, and between $\boldsymbol{f}$ and a wavelet $\boldsymbol{w}_h^1$ to find the detail d_h^1 [1].

We used the Daubechies3 scaling signals and wavelets, named Daubechies6 by some authors [40]. These have the forms shown in (I.1)[1].

$$
\begin{aligned}
&\text{Scaling signals:}\\
\boldsymbol{v}_1^1 =&(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0)\\
\boldsymbol{v}_2^1 =&(0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0)\\
\vdots =&\vdots\\
\boldsymbol{v}_{n/2-1}^1 =&(\alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4)\\
\boldsymbol{v}_{n/2}^1 =&(\alpha_3, \alpha_4, \alpha_5, \alpha_6, 0, 0, \ldots, 0, \alpha_1, \alpha_2)\\
&\text{Wavelets:}\\
\boldsymbol{w}_1^1 =&(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0)\\
\boldsymbol{w}_2^1 =&(0, 0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0)\\
\vdots =&\vdots\\
\boldsymbol{w}_{n/2-1}^1 =&(\beta_5, \beta_6, 0, 0, \ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4)\\
\boldsymbol{w}_{n/2}^1 =&(\beta_3, \beta_4, \beta_5, \beta_6, 0, 0, \ldots, 0, \beta_1, \beta_2),
\end{aligned}
\tag{I.1}
$$

where the approximate values of the filter coefficients are

$$
\begin{aligned}
(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6) =&(0.3327, 0.8069, 0.4599, -0.1350, -0.08544, 0.035226)\\
(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6) =&(\alpha_6, -\alpha_5, \alpha_4, -\alpha_3, \alpha_2, -\alpha_1).
\end{aligned}
$$

We find the scaling and fluctuation coefficients at the first level by taking scalar products with the scaling signals and wavelets in (I.1)

$$
\begin{aligned}
a_h^1 =&\boldsymbol{v}_h^1 \cdot \boldsymbol{f}\\
d_h^1 =&\boldsymbol{w}_h^1 \cdot \boldsymbol{f}.
\end{aligned}
$$

The operation that performs this decomposition is written [1]

$$\boldsymbol{f} \overset{DB3^1}{\rightarrow} (\boldsymbol{a}^1|\boldsymbol{d}^1).$$

The second level trend and fluctuation are found by taking the scalar products of the scaling signals and wavelets (I.1) with $\boldsymbol{a^1}$. The m^{th} level decomposition is found by repeating this process m times, and written

$$\boldsymbol{f} \overset{DB3^m}{\rightarrow} (\boldsymbol{a}^m|\boldsymbol{d}^m|\boldsymbol{d}^{m-1}|\ldots||\boldsymbol{d}^1).$$

The original signal can be reconstructed as follows (I.2) [1]

$$\boldsymbol{f} = a_1^1\boldsymbol{V}_1^1 + a_2^1\boldsymbol{V}_2^1 + \ldots + a_{n/2}^1\boldsymbol{V}_{n/2}^1 + d_1^1\boldsymbol{W}_1^1 + a_2^1\boldsymbol{W}_2^1 + \ldots + d_{n/2}^1\boldsymbol{W}_{n/2}^1. \quad \text{(I.2)}$$

By setting all detail coefficients to zero, we can extract the large trends by reconstructing the signal using the calculated trend coefficients. This reconstruction results in the a denoised version of the original signal. We will refer to the values of the denoised signal as approximation coefficients. An example is shown in figure I.2.

The discrete wavelet analysis in figure I.2 was performed at the third level since the ECG signal was recorded at 128 Hz. Solving for x in $128/(2^x) = 15$, we find $x = \log_2(128/15) \approx 3.0931 \approx 3$. Further details on the choice of the level are given in the description of the QRS detection feature. The emphasis on the largest peak provided by wavelet analysis is used in the calculation of the QRS detection feature described in section 5.3.

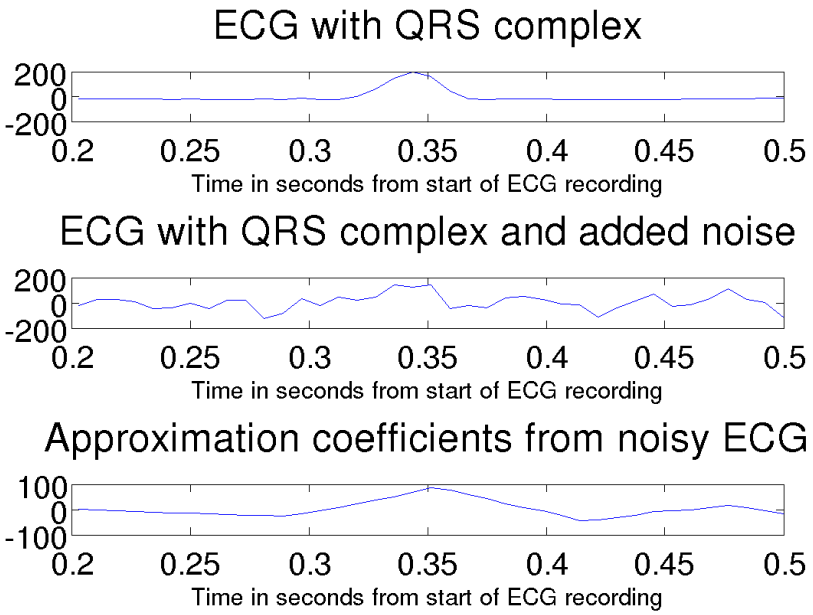

Figure I.2: The top figure shows a typical QRS complex. The middle figure is the top figure with added noise. The bottom shows the third level approximation coefficients of the signal in the middle figure. Even though the bottom figure shows a much flatter curve than the original signal, this is preferable to the noisy middle figure with several smaller peaks.

Appendix J

Confusion matrices

To give a better idea of the performance of the different models under the four scenarios described in subsection 5.4.1, we give the confusion matrices from the test data set in this chapter. We remind the reader that these scenarios are constituted from either using all ICs, or only the manually labeled ICs, and from either distinguishing between all classes or just distinguishing between neural and all other ICs. The first section (section J.1) shows the results when all features were given as input to the classification methods. Section J.2 shows the confusion matrices from the test data when both training and test data only included the features chosen by the MI criterion.

J.1 Confusion matrices from classification based on all features

In this section, we show confusion matrices from each of the four scenarios described in subsection 5.4.1 when the classifiers are given all features as input. Figures J.1, J.2, J.3, and J.4 show the confusion matrices of each of the models that we investigated, except ADJUST. Each figure shows confusion matrices for all methods under one scenario.

Firstly, we would like to point out the oddity that the binary implementations of the BCILab and the standard versions perform equally, while these same implementations show different performances when used in the voting scheme. Also, the multiclass implementation of LDA and the binary implementations perform equally well in the binary classification settings. As expected, the multiclass version of LDA performs better than both the binary versions (standard and BCILab) in the voting scheme.

Similarly, the QDA in the voting scheme had equal or better performance than the version of QDA that takes all classes into account simultaneously in all four cases. This may be due to the problem of too few observations in some classes such that it was necessary to remove features at random from the multiclass version of QDA in order to estimate the model

Strangely, multinomial regression outperforms logistic regression, and multinomial regression with forward selection outperforms logistic regression with forward selection in the binary cases. We would expect to see identical performance of logistic and multinomial regression in binary classification since multinomial regression reduces to logistic regression in this case. Conversely, logistic regression outperforms multinomial regression, both with and without forward selection, in the multiple class cases.

SVM outperforms BBCI. Since SVM is given all features, whereas the SVM used in the BBCI algorithm is only given the features described in [56], which is a subset of the features we use, the superior performance of SVM is expected.

The performances are in general better when only labeled ICs are used. Likewise, performance is better in the binary case than in the multiple class cases.

J.1.1 All features: Only manually labeled observations, several classes

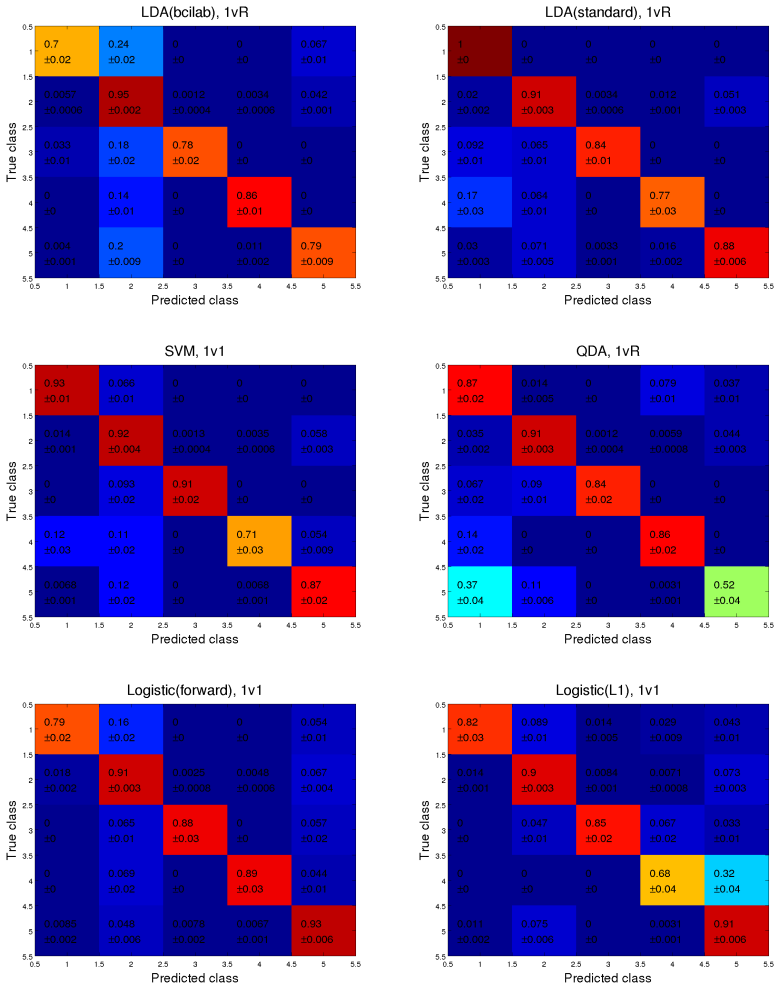

Figure J.1

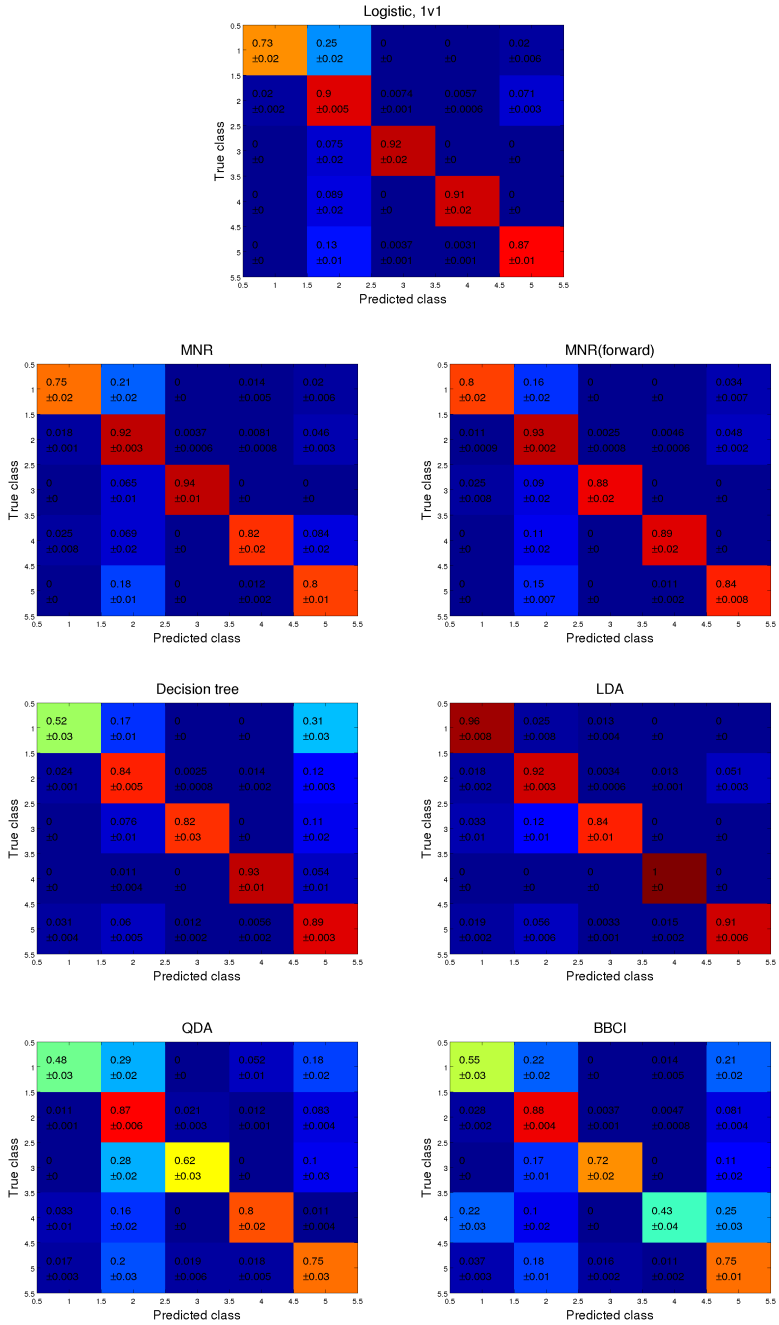

Figure J.1: Comparison of the performance of the different classification methods when only manually labeled independent components are included, taking all classes into account in the classification.

J.1.2 All features: Only manually labeled observations, binary classification

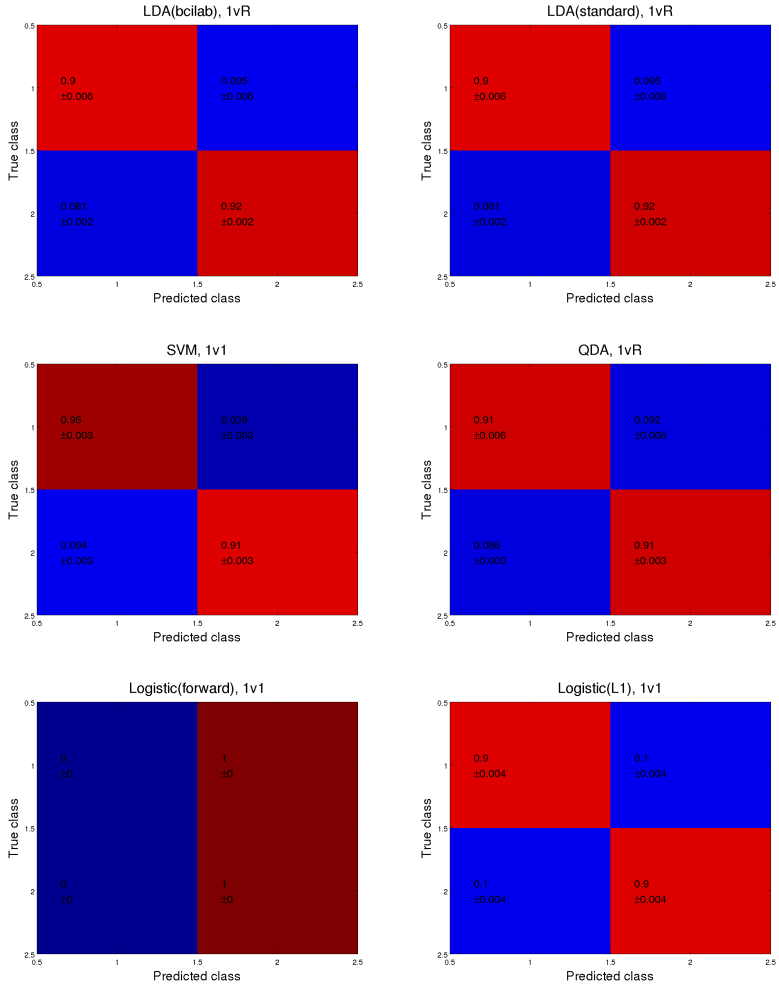

Figure J.2

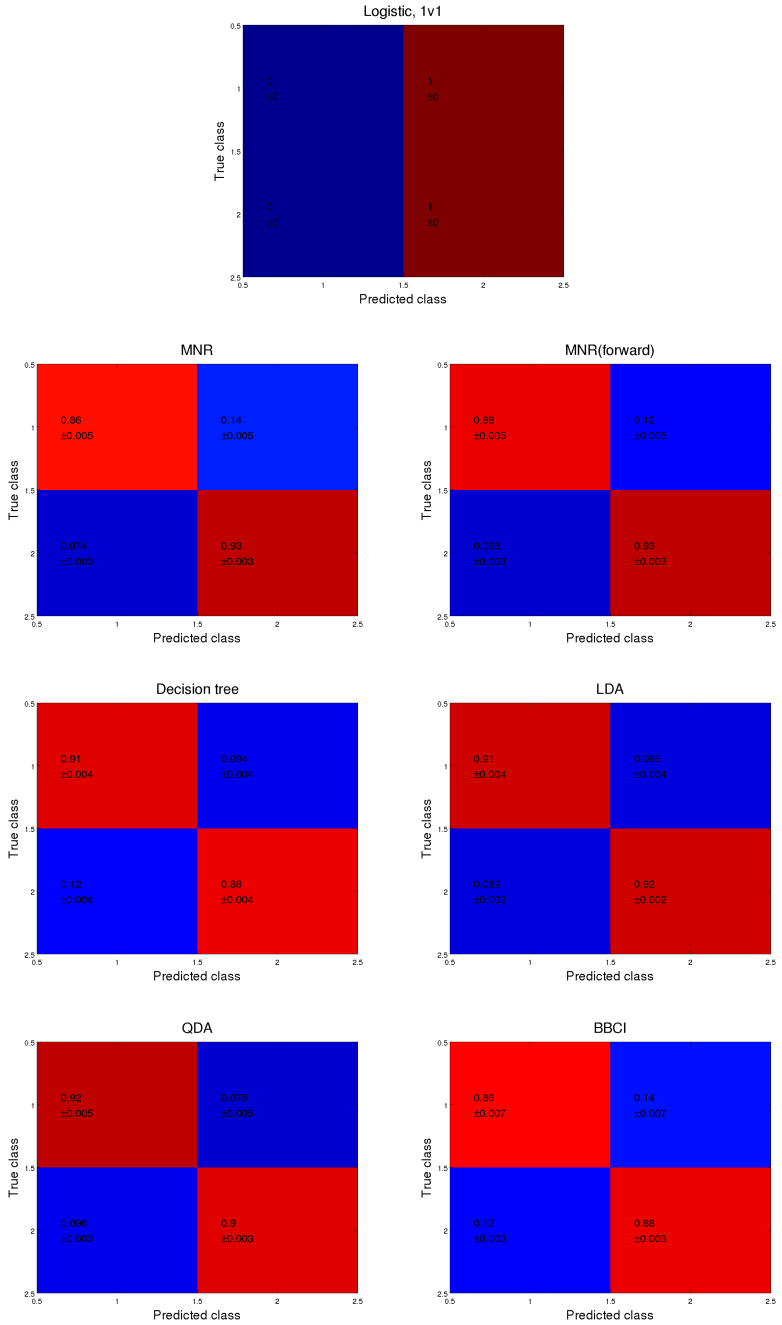

Figure J.2: Comparison of the performance of the different classification methods when only manually labeled independent components are included in the binary classification problem.

J.1.3 All features: All observations, several classes

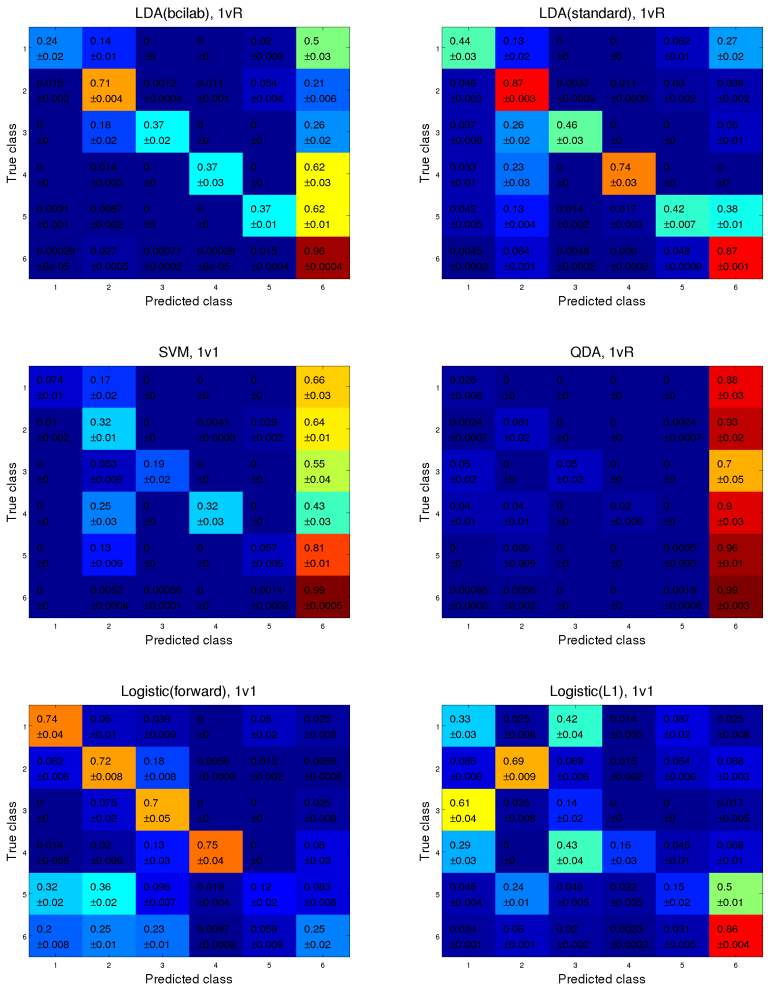

Figure J.3

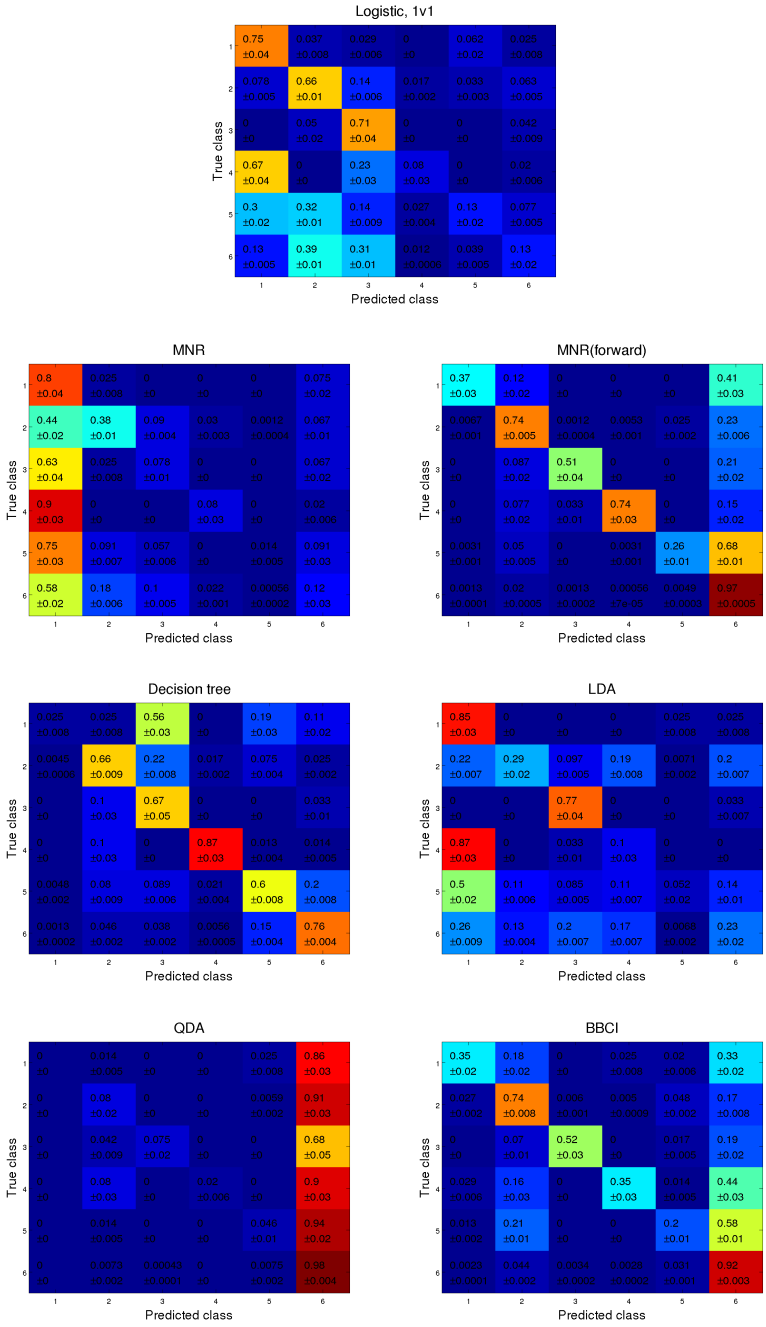

Figure J.3: Comparison of the performance of the different classification methods when all independent components are included, taking all classes into account in the classification.

J.1.4 All features: All observations, binary classification

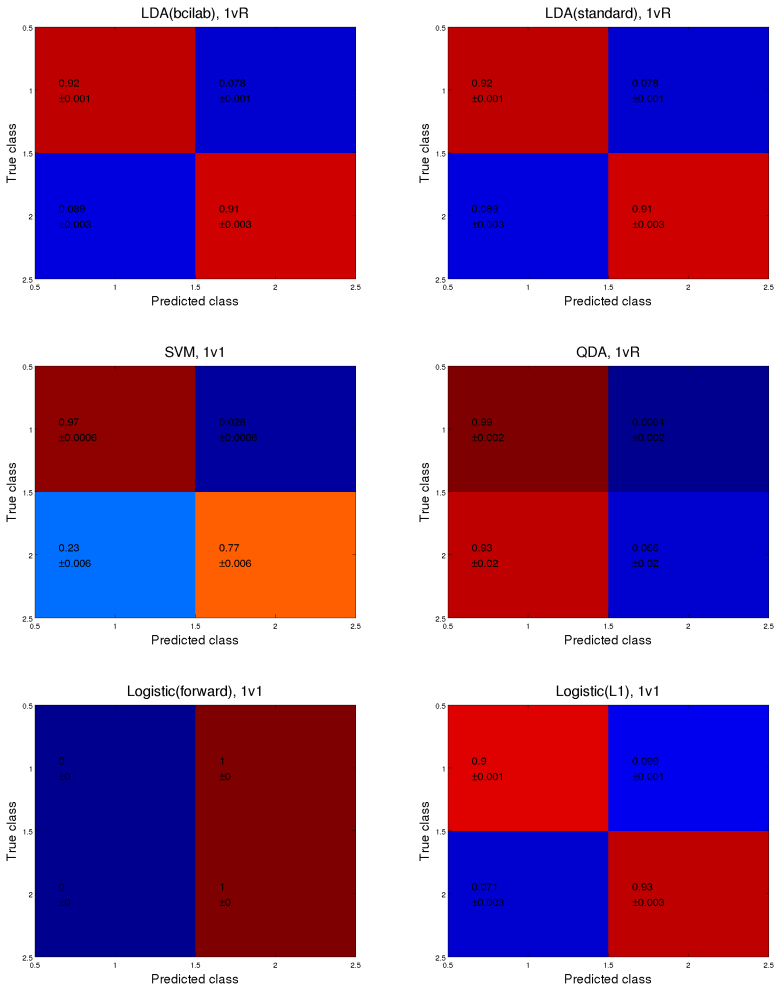

Figure J.4

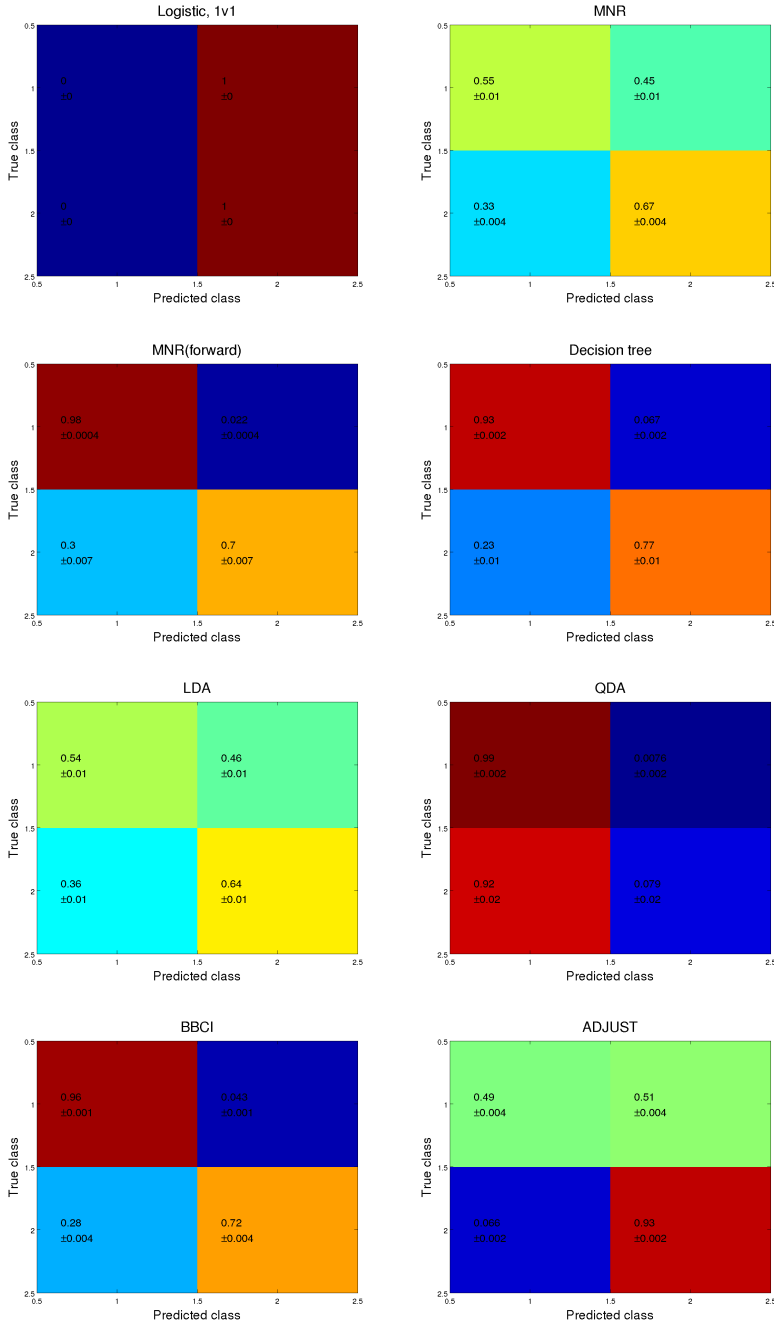

Figure J.4: Comparison of the performance of the different classification methods when all independent components are included on the binary classification problem.

J.2 Confusion matrices from classification based only on best features according to MI criterion

In this section, we show confusion matrices from each of the four scenarios described in subsection 5.4.1 when the classifiers are given only the best features according to the MI-criterion. Figures J.5, J.6, J.7, and J.8 show the confusion matrices of each of the models that we investigated, except ADJUST. Each figure shows confusion matrices for all methods under one scenario.

The patterns described for the confusion matrices for methods using all features also hold these confusion matrices, which were based on classifiers that were only given the best features according to the MI-criterion.

Compared to the classifiers using all features, those using only the best features perform equally well, or a little worse. The equal performance is probably due to the fact that almost all features were chosen by the MI-criterion in all cases, as explained in section 7.4.

J.2.1 MI-criterion features: Only manually labeled observations, several classes

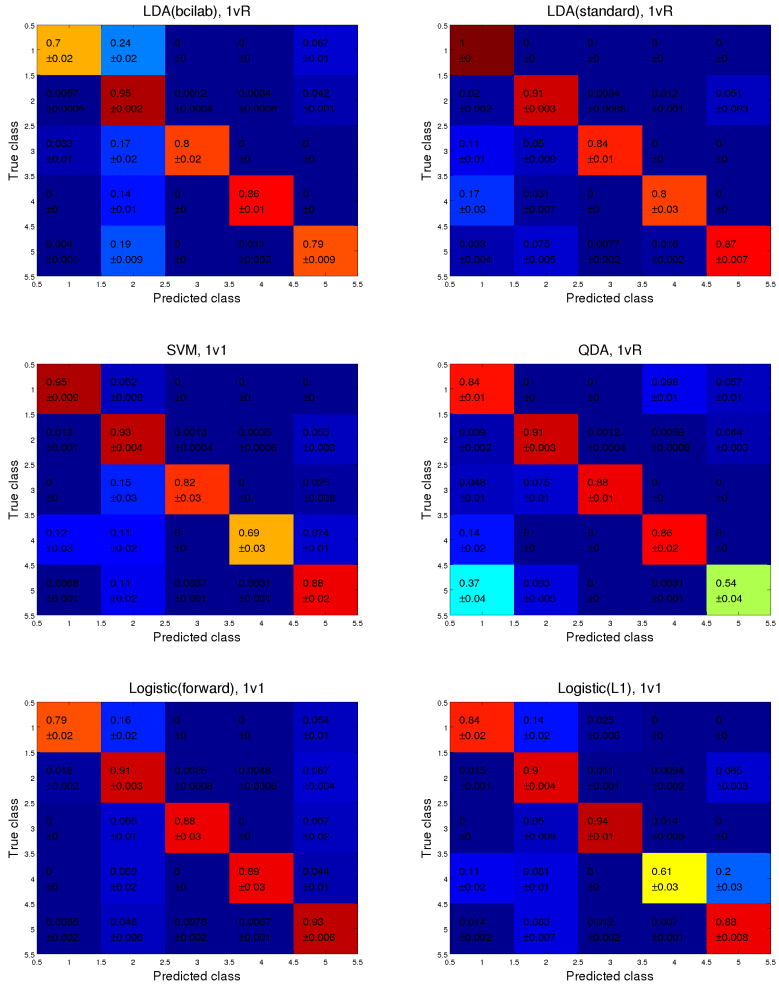

Figure J.5

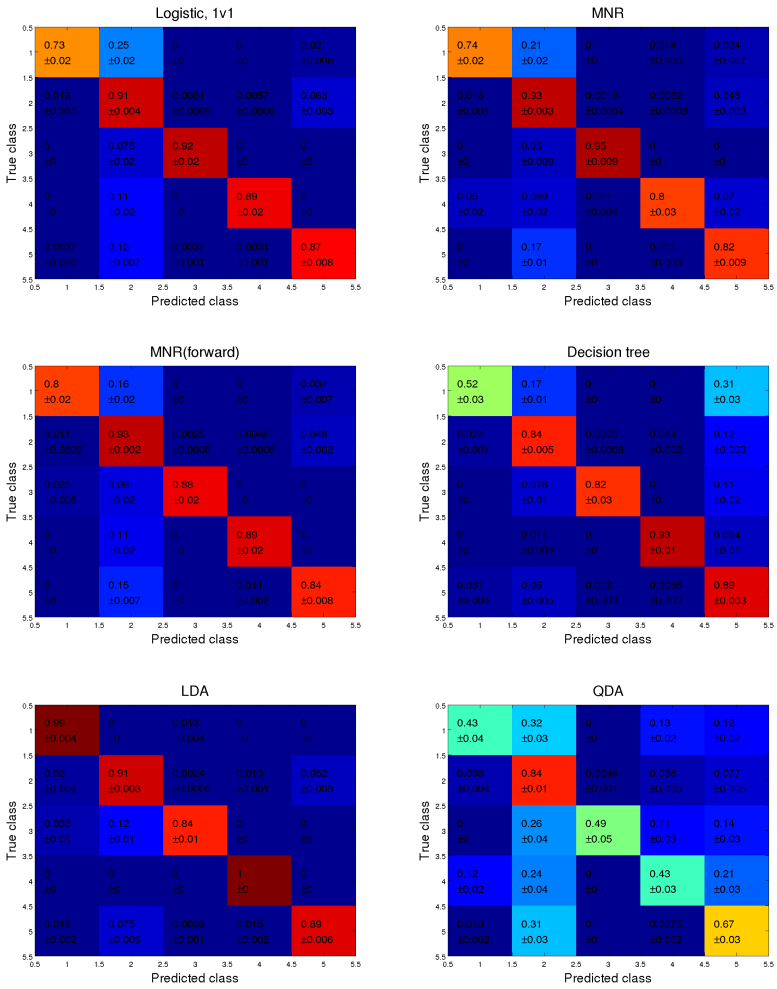

Figure J.5: Comparison of the performance of the different classification methods when only manually labeled independent components are included, taking all classes into account in the classification.. Only the best features according to the MI criterion were taken into account.

J.2.2 MI-criterion features: Only manually labeled observations, binary classification

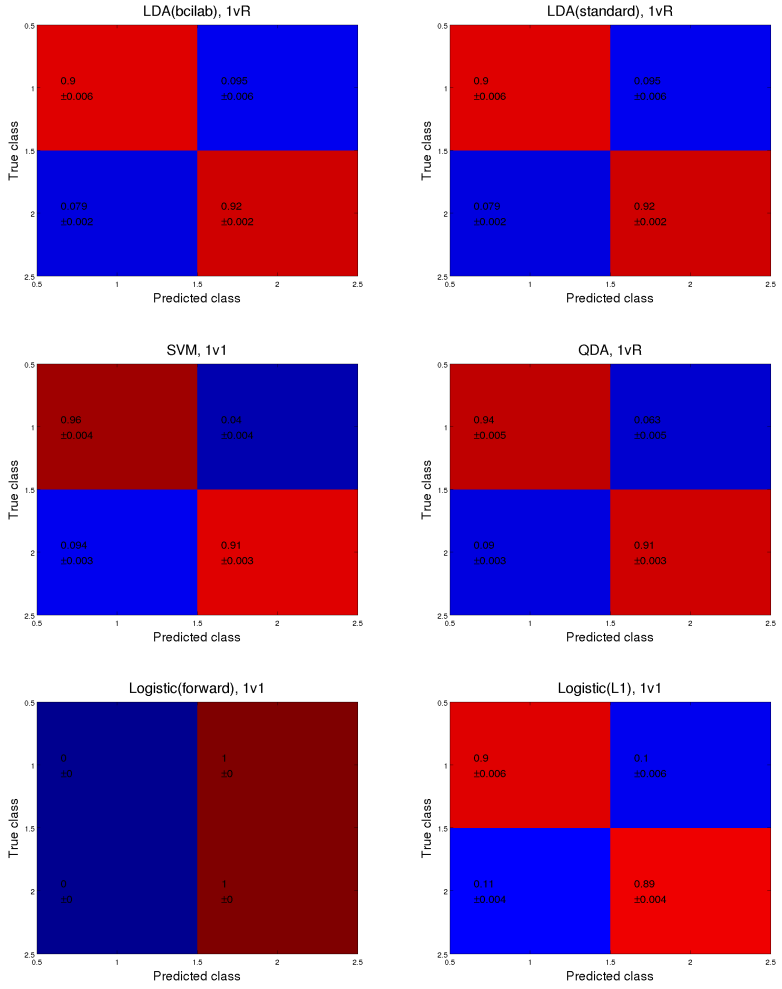

Figure J.6

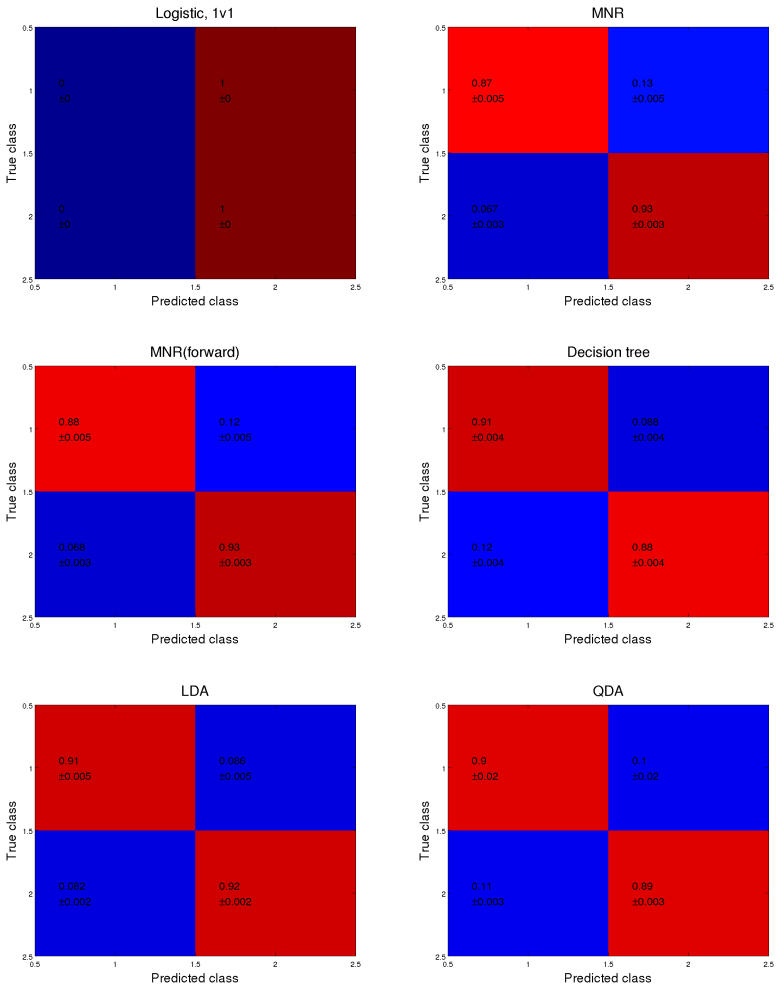

Figure J.6: Comparison of the performance of the different classification methods when only manually labeled independent components are included on the binary classification problem. Only the best features according to the MI criterion were taken into account.

J.2.3 MI-criterion features: All observations, several classes

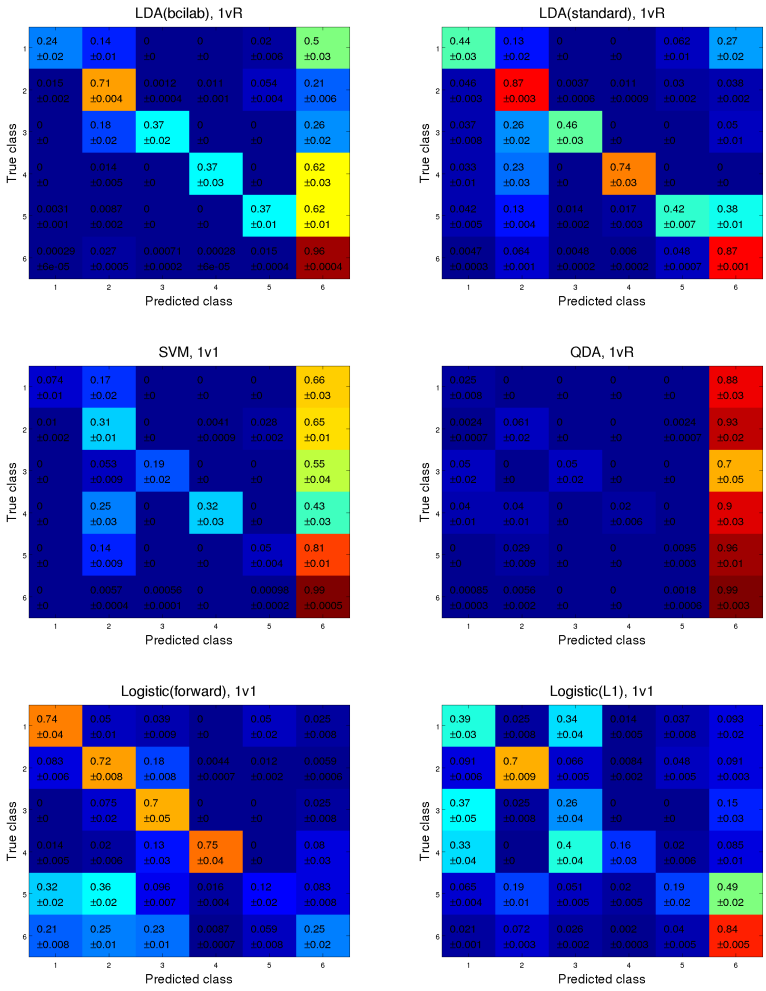

Figure J.7

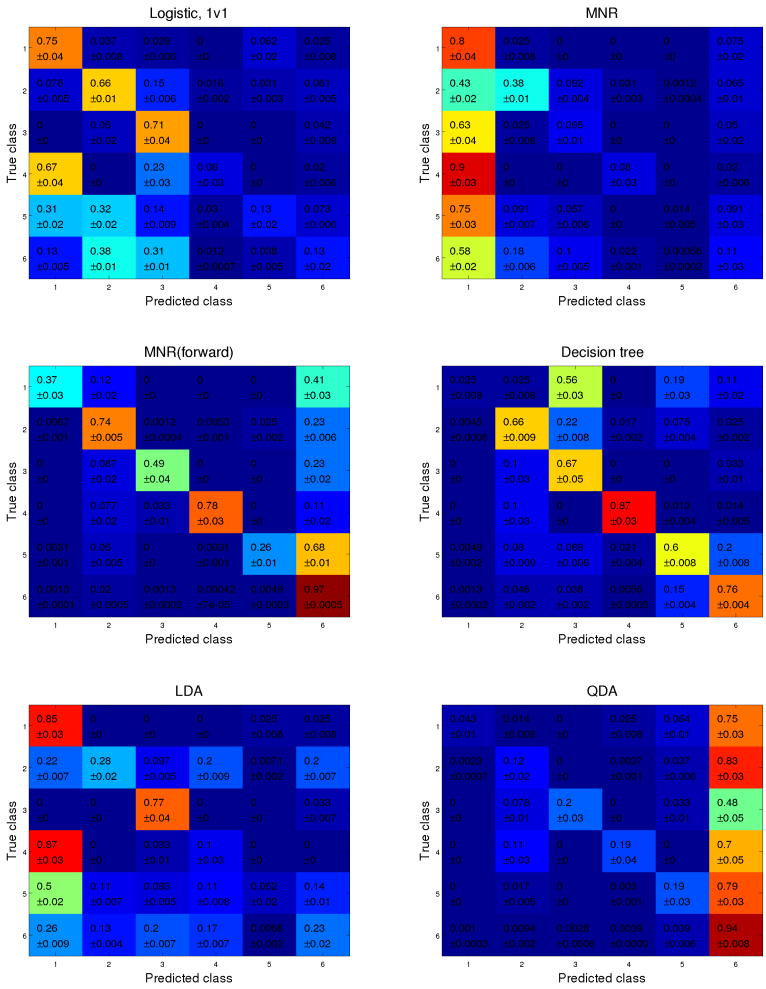

Figure J.7: Comparison of the performance of the different classification methods when all independent components are included, taking all classes into account in the classification. Only the best features according to the MI criterion were taken into account.

J.2.4 MI-criterion features: All observations, binary classification

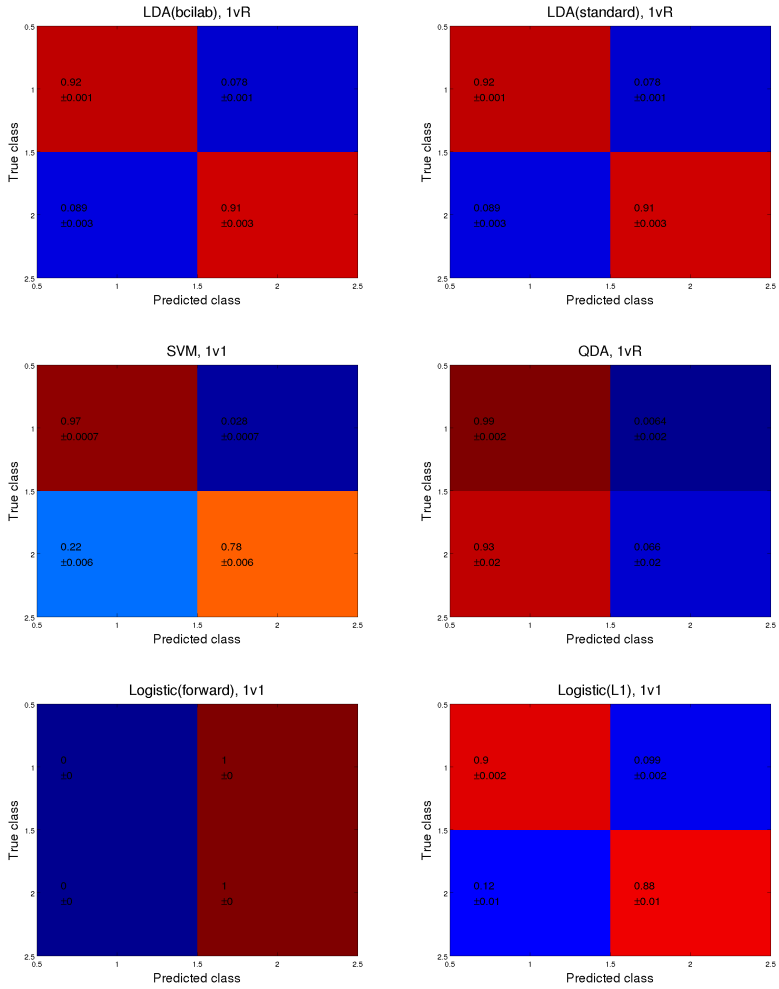

Figure J.8

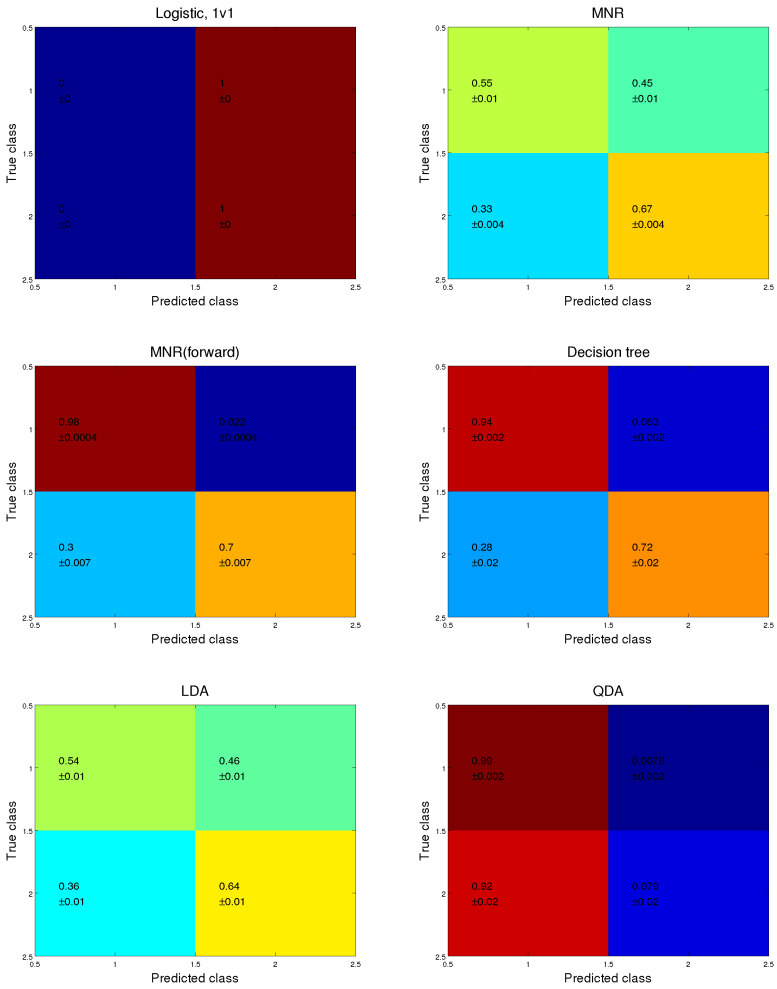

Figure J.8: Comparison of the performance of the different classification methods when all independent components are included in the binary classification problem. Only the best features according to the MI criterion were taken into account.

Appendix K

Empirical variance estimates of feature coefficient estimates

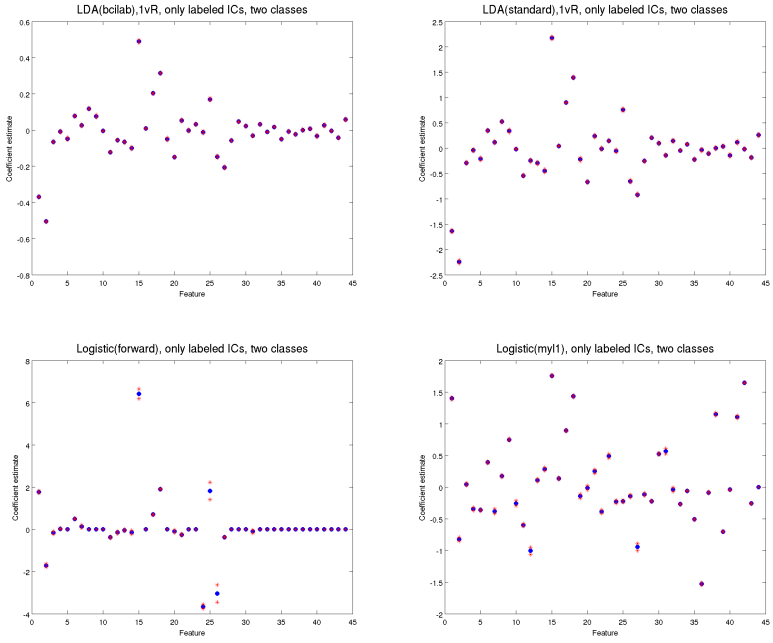
LDA(bcilab),1vR, only labeled ICs, two classes
LDA(standard),1vR, only labeled ICs, two classes
Logistic(forward), only labeled ICs, two classes
Logistic(myl1), only labeled ICs, two classes
Coefficient estimate
Feature

Figure K.1

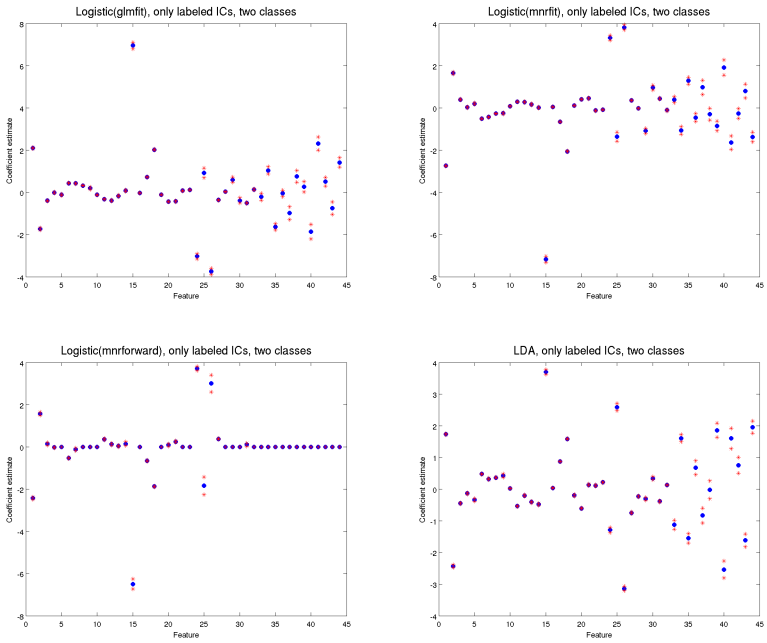

Figure K.1: Blue circles represent mean values of coefficient estimates. Red asterisks represent the mean values plus and minus one standard deviation. Only labeled ICs were used.

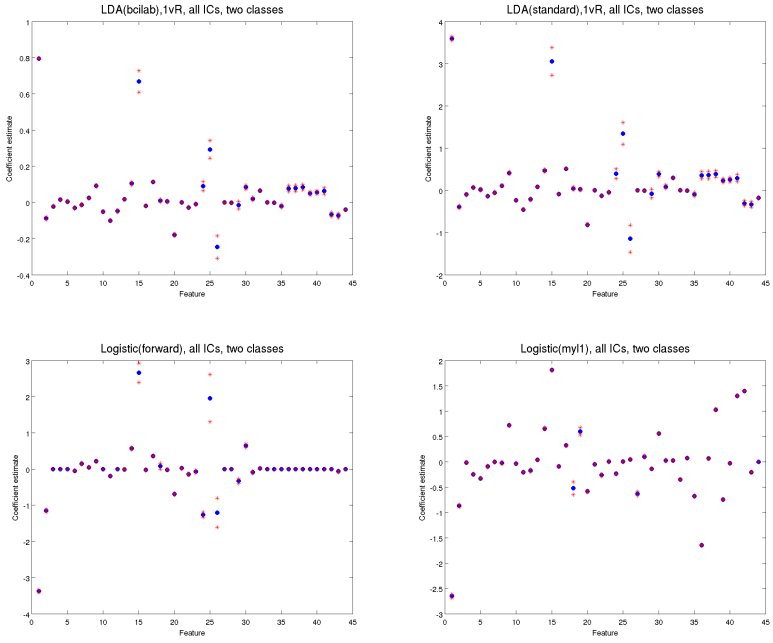
LDA(bcilab),1vR, all ICs, two classes
LDA(standard),1vR, all ICs, two classes
Logistic(forward), all ICs, two classes
Logistic(myl1), all ICs, two classes
Coefficient estimate
Feature

Figure K.2

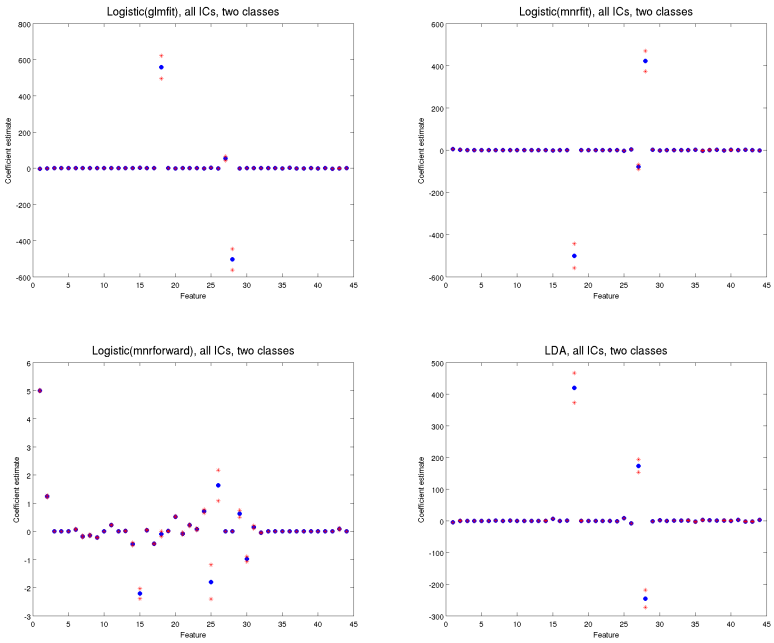

Figure K.2: Blue circles represent mean values of coefficient estimates. Red asterisks represent the mean values plus and minus one standard deviation. All ICs were used.

Appendix L

Implementation of mutual information

```
1  function MI = MutualInformation(X, nbins) % takes two arguments since
       an old version of the code required two arguments, and calls to the
        old code thus pass two arguments
2
3  nbins=ceil(min(50,3*log2(1+size(X, 1)/10)));
4
5
6  ptiles = 0:(1/nbins):1;
7  ptiles = ptiles*100;
8  %maxx = max(X);
9  binsplits = prctile(X, ptiles, 1);
10 %binsplits = [binsplits; maxx+2];
11
12 marginal_dists = zeros(nbins, size(X,2));
13 for ivariable = 1:size(X,2)
14     marginal_dists(:, ivariable) = count_interval_observations(X(:,
           ivariable), binsplits(:, ivariable));
15     %marginal_dists(:, ivariable)=marginal_dists(:, ivariable)./abs(
           diff(binsplits(:, ivariable)));
16     marginal_dists(:, ivariable)=marginal_dists(:, ivariable)/sum(
           marginal_dists(:, ivariable));
17
18 end
19
20 joint_dists = zeros(size(X,2), size(X,2), nbins, nbins);
21 for ivariable1 = 1:size(X,2)
22     for ivariable2 = 1:size(X,2)
```

```
23          joint_dists(ivariable1, ivariable2, :, :) =
                count_rectangle_observations(X(:,[ivariable1, ivariable2])
                ,...
24              binsplits(:, [ivariable1, ivariable2]));
25          joint_dists(ivariable1, ivariable2, :,:)=joint_dists(ivariable1
                , ivariable2, :,:)/size(X,1);
26
27      end
28  end
29
30  PXPY = zeros(size(X,2), size(X,2), nbins, nbins);
31  for ivariable1 = 1:size(X,2)
32      for ivariable2 = 1:size(X,2)
33          for ibin=1:nbins
34              PXPY(ivariable1, ivariable2, ibin, :) = marginal_dists(ibin
                    , ivariable1).*...
35                  marginal_dists(:, ivariable2);
36          end
37      end
38  end
39
40  MI = zeros(size(X,2), size(X,2));
41  for ivariable1 = 1:size(X,2)
42      for ivariable2 = 1:size(X,2)
43          MI(ivariable1, ivariable2) = nansum(nansum(...
44              joint_dists(ivariable1, ivariable2, :, :).*...
45              (log(joint_dists(ivariable1, ivariable2, :, :))-log(PXPY(
                    ivariable1, ivariable2, :, :)))...
46              ));
47      end
48  end
49
50  end
51
52  function nobs_intervalwise = count_interval_observations(obs, splits)
53  splits(end) = splits(end)+1;
54  nobs_intervalwise  = zeros(length(splits)-1,1);
55  for isplit=1:(length(splits)-1)
56      nobs_intervalwise(isplit) = sum(((obs < splits(isplit+1)) + (obs >=
            splits(isplit))) ==2);
57  end
58  end
59
60
61  function nobs_rectanglewise = count_rectangle_observations(obs, splits)
62  splits(end,:)=splits(end,:)+1;
63  nobs_rectanglewise = zeros(length(splits)-1,length(splits)-1);
64  for isplit1=1:(size(splits,1)-1)
65      for isplit2=1:(size(splits,1)-1)
66          nobs_rectanglewise(isplit1, isplit2) = sum(((obs(:,1) < splits(
                isplit1+1,1)) + (obs(:,1) >= splits(isplit1,1)) +...
67              (obs(:,2) < splits(isplit2+1,2)) + (obs(:,2) >= splits(
                    isplit2,2))) ==4);
68      end
69  end
```

```
70 end
```

Appendix M

Implementation of L1-regularized logistic regression

```
1  function coefs = mylogregl1(varargin)
2
3  arg_define([0 2],varargin, ...
4      arg_norep('trials'), ...
5      arg_norep('targets'), ...
6      arg({'lambd','Lambda', 'lambda', 'lam'}, 0.5, [0 1], '
           Regularization parameter. The higher the greater the penalty on
            many non-zero coefficient estimates.'), ...
7      arg({'obsweights','weights','observationWeights'}, [], [], 'The
           weight of each observation in fit of model. A higher weight for
            an observation makes the cost of misclassifying that
           observation higher.'), ...
8      arg({'coefs'}, [], [], 'Initial guess at coefficients.'), ...
9      arg({'maxeval'}, 100, [], 'Stopping criterium. Stop if number of
           function evaluations exceeds  maxeval'), ...
10     arg({'x_change'}, 10^(-8), [], 'Stopping criterium. Stop if all
           parameters change less than this.'));
11 estimate_coefficients=true;
12
13 if isempty(obsweights)
14     obsweights = ones(length(targets),1);
15 end
16
```

```
17 classes = unique(targets);
18
19 if length(classes)~=2
20     error('bcilab09_userscripts:mylogregl1:class_number', 'y must
           contain two classes')
21 end
22
23 tars = targets;
24 tars(tars==classes(1))=0;
25 tars(tars==classes(2))=1;
26 if isempty(coefs)
27     coefs = randn(size(trials,2),1);
28 end
29 if length(coefs)<1
30     warning(['bcilab:userscripts:mylogregl1', 'length of initial
           coefficient vector estimate is zero. Number of columns in input
            data is ' num2str(size(trials,2))])
31     estimate_coefficients=false;
32     coefs=zeros(size(trials,2),1);
33 end
34 intercept = randn(1,1);
35
36 %lllogistic_fit_error(coefs(2:end), coefs(1));
37 its=0;
38 x=trials;
39 y=tars;
40 observation_weights=obsweights;
41 mu = 1; % step size
42 exp_term = exp((intercept+x*coefs));
43
44 costs_current = observation_weights.*(lambd*(y.*(intercept+x*coefs)-log
       (1+exp_term)) - (1-lambd)*sum(abs(coefs)));
45
46 costs_current = -costs_current; % since we are using an algorithm that
       minimizes, we
47 %take the negative of the current costs
48 nparams_conv=0;
49 while its <= maxeval && nparams_conv < (length(coefs)+1)
50     % find gradient, i.e. first partial derivative with respect to the
51     % coefficients.
52     J = zeros(size(x,1), length(coefs)+1);
53     J(:,1) = lambd*observation_weights.*(y - exp_term./(1+exp_term) );
54     for i=2:(length(coefs)+1)
55         J(:,i) = lambd*observation_weights.*(y.*x(:,i-1) - x(:,i-1).*
               exp_term./(1+exp_term) );
56     end
57     J=-J;
58
59     if estimate_coefficients
60         for i=1:(length(coefs))
61             Jpenal = zeros(size(x,1), length(coefs));
62             Jpenal(:,i) = -observation_weights.*(1-lambd)*sign(coefs(i)
                   );
63         end
64         Jpenal=-Jpenal;
```

```
65
66          % update coefficient estimates
67          coefs_new_temp = coefs - mu*sum(J(:,2:end))';
68          coefs_new = coefs_new_temp - mu*sum(Jpenal)';
69
70          coefs_new(sign(coefs_new)~=sign(coefs_new_temp))=0;
71      else
72          coefs_new=coefs;
73      end
74      intercept_new = intercept - mu*sum(J(:,1));
75
76      exp_term_new = exp(intercept_new+x*coefs_new);
77      costs_new = -observation_weights.*(lambd*(y.*(intercept_new+x*
            coefs_new)-log(1+exp_term_new)) -...
78          (1-lambd)*sum(abs(coefs_new)));
79
80      cost_current=sum(costs_current);
81      cost_improvement = cost_current-sum(costs_new); % this is positive
            if the new estimates are better
82
83      % we only change the estimates of the coefficients if an
            improvement in
84      % the cost function was obtained.
85      if cost_improvement>0 && isfinite(cost_improvement)
86          param_changes =abs([intercept coefs']-[intercept_new coefs_new
                ']);
87          intercept=intercept_new;
88          coefs=coefs_new;
89          exp_term=exp_term_new;
90          costs_current=costs_new;
91          mu = mu*1.5;
92          nparams_conv=sum(param_changes<x_change);
93
94      else
95          mu=mu/2;
96          nparams_conv =0;
97      end
98      its=its+1;
99  end
100 coefs = [intercept coefs']';
```

Bibliography

[1] *A Primer on Wavelets and their Scientific Applications*. 2nd ed. Chapman & Hall/CRC, 2008.

[2] K. V. Allen and B. M. Frier. "Nocturnal hypoglycemia: clinical manifestations and therapeutic strategies toward prevention". In: *Endocr Pract* 9 (2003), pp. 530–543.

[3] *Applied Iterative Methods*. A K Peters/CRC Press, 2007. ISBN: 978-1-56881-342-4.

[4] G. Bartels, Li-Chen Shi, and Bao-Liang Lu. "Automatic artifact removal from EEG - a mixed approach based on double blind source separation and support vector machine". In: *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. 2010, pp. 5383 –5386. DOI: 10.1109/IEMBS.2010.5626481.

[5] HENNING BECK-NIELSEN. *METHOD AND APPARATUS FOR PREDICTION AND WARNING OF HYPOGLYCAEMIC ATTACK*. Mar. 2011. URL: \url{http://www.wipo.int/patentscope/search/en/detail.jsf?docId=EP9916832&recNum=2&office=&queryString=ALL\%3A\%28hyposafe\%29&prevFilter=&sortOption=Pub+Date+Desc&maxRec=3}.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1st ed. 2006. Corr. 2nd printing. Springer, 2007. ISBN: 0387310738. URL: \url{http://www.worldcat.org/isbn/0387310738}.

[7] B. Blankertz et al. "Optimizing Spatial filters for Robust EEG Single-Trial Analysis". In: *Signal Processing Magazine, IEEE* 25.1 (2008), pp. 41 –56. ISSN: 1053-5888. DOI: 10.1109/MSP.2008.4408441.

[8] Benjamin Blankertz. *BCI Competition IV - Final Results* -. Mar. 2011. URL: \url{http://www.bbci.de/competition/iv/results/index.html}.

[9] TARAS BODNAR and YAREMA OKHRIN. "On the Product of Inverse Wishart and Normal Distributions with Applications to Discriminant Analysis and Portfolio Theory". In: *Scandinavian Journal of Statistics* 38.2 (2011), pp. 311–331. ISSN: 1467-9469. DOI: 10.1111/j.1467-9469.2011.00729.x. URL: http://dx.doi.org/10.1111/j.1467-9469.2011.00729.x.

[10] Brian V. Bonnlander and Andreas S. Weigend. "Selecting Input Variables Using Mutual Information and Nonparametric Density Estimation". In: 1996, pp. 42–50.

[11] E. W. ter Braak et al. "Maternal hypoglycemia during pregnancy in type 1 diabetes: maternal and fetal consequences". In: *Diabetes Metab. Res. Rev.* 18 (2002), pp. 96–105.

[12] Angel Navia Vazquez Carlos Guerrero-Mosquera. "Automatic removal of ocular artifacts from EEG data using adaptive filtering and independent component analysis". In: 17th European Signal Processing Conference (EUSIPCO 2009). Glasgow, Scotland 2009.

[13] Arnaud Delorme and Scott Makeig. *Chapter 09: Decomposing Data Using ICA*. May 2010. URL: http://sccn.ucsd.edu/wiki/Chapter_09:_Decomposing_Data_Using_ICA.

[14] Arnaud Delorme and Scott Makeig. "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis". In: *Journal of Neuroscience Methods* 134 (2004), pp. 9–21.

[15] Arnaud Delorme, Terrence Sejnowski, and Scott Makeig. "Enhanced detection of artifacts in EEG data using higher-order statistics and independent component analysis". In: *NeuroImage* 34.4 (2007), pp. 1443 –1449. ISSN: 1053-8119. DOI: DOI:10.1016/j.neuroimage.2006.11.004. URL: \url{http://www.sciencedirect.com/science/article/B6WNP-4MNHY2V-4/2/93de9223a58e55c80f19ecdb50e8dcfe}.

[16] Arnaud Delorme et al. "EEGLAB, SIFT, NFT, BCILAB, and ERICA: New Tools for Advanced EEG Processing." In: *Comp. Int. and Neurosc.* 2011 (2011). URL: http://dblp.uni-trier.de/db/journals/cin/cin2011.html#DelormeMKASVM11.

[17] Makeig S Delorme A and TJ. Sejnowski. "Automatic artifact rejection for EEG data using high-order statistics and independent component analysis". In: Third International Workshop on Independent Component Analysis and Signal Separation. San Diego 2001.

[18] Rohtash Dhiman, J.S. Saini, and A.P Mittal Priyanka. "ARTIFACT REMOVAL FROM EEG RECORDINGS – AN OVERVIEW". In: NCCI 2010 -National Conference on Computational Instrumentation. 2010.

[19] C. Flykanaka-Gantenbein. "Hypoglycemia in childhood: long-term effects". In: *Pediatr Endocrinol Rev* 1 Suppl 3 (Aug. 2004), pp. 530–536.

[20] D. François, V. Wertz, and M. Verleysen. "The permutation test for feature selection by mutual information". In: *in: ESANN 2006, European Symposium on Artificial Neural Networks*. 2006, pp. 239–244.

[21] Jun Feng Gao et al. "Automatic Removal of Eye-Movement and Blink Artifacts from EEG Signals". In: *Brain Topography* 23.1 (1 2010). 10.1007/s10548-009-0131-4, pp. 105–114. ISSN: 0896-0267. URL: \url{http://dx.doi.org/10.1007/s10548-009-0131-4}.

[22] Brain Products GmbH. *Brain Vision Analyzer User Manual, version 1.05*. Tech. rep. Brain Products GmbH, 1999.

[23] K. Gramann, T. Tollner, and H. J. Muller. "Dimension-based attention modulates early visual processing". In: *Psychophysiology* 47 (Sept. 2010), pp. 968–978.

[24] HypoSafe. *CLINICAL INVESTIGATION*. Feb. 2010. URL: \url{http://www.hyposafe.com/index.php/site/Pages/clinical_trials/info}.

[25] Aapo Hyvarinen. *Independent component analysis: a tuturial*. May 2010. URL: http://www.cis.hut.fi/aapo/papers/IJCNN99_tutorialweb/node9.html.

[26] A. Hyvärinen and E. Oja. "Independent component analysis: algorithms and applications". In: *Neural Networks* 13.4-5 (2000), pp. 411 –430. ISSN: 0893-6080.

[27] Aapo Hyvärinen. "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis". In: *IEEE Transactions on Neural Networks* 10.3 (1999), pp. 626–634. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.4731.

[28] Immrama Institute. *The International 10-20 System of Electrode Placement*. Mar. 2010. URL: http://www.immrama.org/eeg/electrode.html.

[29] A. Jain, D. Sarraf, and D. Fong. "Preventing diabetic retinopathy through control of systemic factors". In: *Curr Opin Ophthalmol* 14 (Dec. 2003), pp. 389–394.

[30] Carrie A Joyce, Irina F Gorodnitsky, and Marta Kutas. "Automatic removal of eye movement and blink artifacts from EEG data using blind component separation." In: *Psychophysiology* 41.2 (Mar. 2004), pp. 313–325. ISSN: 0048-5772. DOI: 10.1111/j.1469-8986.2003.00141.x. URL: \url{http://dx.doi.org/10.1111/j.1469-8986.2003.00141.x}.

[31] Claus B. Juhl et al. "Automated detection of hypoglycemia-induced EEG changes recorded by subcutaneous electrodes in subjects with type 1 diabetes-The brain as a biosensor". In: *Diabetes Research and Clinical Practice* In Press, Corrected Proof (Jan. 2010). DOI: 10.1016/j.diabres.2010.01.007. URL: \url{http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6T5Y-4Y5H5Y1-2&_user=10&_coverDate=01%2F15%2F2010&_rdoc=1&_fmt=high&_orig=search&_sort=d&_docanchor=&view=c&_searchStrId=1226016776&_rerunOrigin=google&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=0d90939e8fffe4239677405131f33c57}.

[32] R. Jung, W. Berger, and H. Berger. "[Fiftieth anniversary of Hans Berger's publication of the electroencephalogram. His first records in 1924–1931 (author's transl)]". In: *Arch Psychiatr Nervenkr* 227 (Dec. 1979), pp. 279–300.

[33] T.-P. Jung et al. "Imaging brain dynamics using independent component analysis". In: *Proceedings of the IEEE* 89.7 (July 2001), pp. 1107 –1122. ISSN: 0018-9219. DOI: 10.1109/5.939827.

[34] H Klekowicz et al. "Automatic analysis of sleep EEG". In: Front. Neuroinform. Conference Abstract: Neuroinformatics. 2008. DOI: 10.3389/conf.neuro.11.2008.01.107.

[35] Th. Kohler et al. "Depth normalization in MEG/EEG current density imaging". In: *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE.* Vol. 2. 1996, 812 –813 vol.2. DOI: 10.1109/IEMBS.1996.651989.

[36] Z. J. Koles. "The quantitative extraction and topographic mapping of the abnormal components in the clinical EEG". In: *Electroencephalogr Clin Neurophysiol* 79 (Dec. 1991), pp. 440–447.

[37] J. L. Lancaster et al. "Automated labeling of the human brain: a preliminary report on the development and evaluation of a forward-transform method". In: *Hum Brain Mapp* 5 (1997), pp. 238–242.

[38] J. L. Lancaster et al. "Automated Talairach atlas labels for functional brain mapping". In: *Hum Brain Mapp* 10 (July 2000), pp. 120–131.

[39] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. 2nd ed. Wiley Series in Probability and Statistics. New York: Wiley, 2002.

[40] MatLab. *Wavelet Families: Additional Discussion*. Tech. rep. The MathWorks, Inc., 1994–2008.

[41] A. Mognon et al. "ADJUST: An Automatic EEG artifact Detector based on the Joint Use of Spatial and Temporal features." In: *Psychophysiology* (2010). DOI: 10.1111/j.1469-8986.2010.01061.x.

[42] *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications and Related Fields*. 6th ed. Lippincott Williams and Wilkins, 2010.

[43] Hans Bruun Nielsen. *A MATLAB TOOLBOX FOR OPTIMIZATION AND DATA FITTING*. July 2011. URL: \url{http://www2.imm.dtu.dk/~hbn/immoptibox/}.

[44] H. Nolan, R. Whelan, and R. B. Reilly. "FASTER: Fully Automated Statistical Thresholding for EEG artifact Rejection". In: *J. Neurosci. Methods* 192 (Sept. 2010), pp. 152–162.

[45] "Observed Confidence Levels, Theory and Application". In: Chapman and Hall/CRC, 2008. Chap. Appendix A. Review of Asymptotic Statistics, 235–249.

[46] Julie A Onton and Scott Makeig. "High-frequency broadband modulation of electroencephalographic spectra". In: *Frontiers in Human Neuroscience* 0 (2009), p. 12. ISSN: 1662-5161. DOI: 10.3389/neuro.09.061.2009. URL: http://www.frontiersin.org/Journal/Abstract.aspx?s=537&name=humanneuroscience&ART_DOI=10.3389/neuro.09.061.2009.

[47] Stig Pramming et al. "Glycaemic Threshold For Changes In Electroencephalograms During Hypoglycaemia In Patients With Insulin Dependent Diabetes". In: *British Medical Journal (Clinical Research Edition)* 296.6623 (Mar. 1988), pp. 665–667. URL: \url{http://www.jstor.org/pss/29529969}.

[48] Rajesh P.N. Rao and Reinhold Scherer. "Statistical Pattern Recognition and Machine Learning in Brain-Computer Interfaces". In: *Statistical Signal Processing for Neuroscience and Neurotechnology*. Ed. by Karim G. Oweiss. Oxford: Academic Press, 2010, pp. 335 –367. ISBN: 978-0-12-375027-3. DOI: DOI:10.1016/B978-0-12-375027-3.00010-7. URL: http://www.sciencedirect.com/science/article/pii/B9780123750273000107.

[49] Peter Rossing and Dick de Zeeuw. "Need for better diabetes treatment for improved renal outcome." In: *Kidney international* 79 Suppl 1 (Mar. 2011), S28–32. ISSN: 1523-1755. DOI: 10.1038/ki.2010.513. URL: http://www.ncbi.nlm.nih.gov/pubmed/21358699.

[50] Giulia Rotundo. "The Hurst's exponent in technical analysis signals". In: *Practical Fruits of Econophysics*. Ed. by Hideki Takayasu. Springer Tokyo, 2006, pp. 121–125. ISBN: 978-4-431-28915-9. URL: \url{http://dx.doi.org/10.1007/4-431-28915-1_21}.

[51] M. Sakai and D. Wei. "Separation of electrocardiographic and encephalographic components based on signal averaging and wavelet shrinkage techniques". In: *Comput. Biol. Med.* 39 (July 2009), pp. 620–629.

[52] Shi-Yun Shao et al. "Automatic EEG artifact removal: a weighted support vector machine approach with error correction." In: *IEEE Trans Biomed Eng* 56.2 (Feb. 2009), pp. 336–44.

[53] L. Shoker, S. Sanei, and J. Chambers. "Artifact removal from electroencephalograms using a hybrid BSS-SVM algorithm". In: *Signal Processing Letters, IEEE* 12.10 (Oct. 2005), pp. 721 –724. ISSN: 1070-9908. DOI: 10.1109/LSP.2005.855539.

[54] "Statistics for Environmental Engineers, Second Edition". In: CRC Press, 2002. Chap. 49.

[55] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2006.

[56] Michael Tangermann et al. "Classification of artifactual ICA components". In: *Int J Bioelectromagnetism* 11.2 (2009), pp. 110–114.

[57] *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2009.

[58] Poul Thyregod and Henrik Madsen. *Introduction to General and Generalized Linear Models (Chapman & Hall/Crc Texts in Statistical Science)*. 1st ed. CRC, 2010. ISBN: 1420091557. URL: \url{http://www.worldcat.org/isbn/1420091557}.

[59] G. Tognini et al. "Diabetes mellitus: CT findings of unusual complications related to the disease: a pictorial essay". In: *Clin Imaging* 27 (2003), pp. 325–329.

[60] G. D. Tourassi et al. "Application of the mutual information criterion for feature selection in computer-aided diagnosis". In: *Med Phys* 28 (Dec. 2001), pp. 2394–2402.

[61] Filipa Campos Viola et al. "Semi-automatic identification of independent components representing EEG artifact." In: *Clin Neurophysiol* 120.5 (2009), pp. 868–77.

[62] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, 2003. ISBN: 0387402721. URL: \url{http://www.worldcat.org/isbn/0387402721}.

[63] Eric W Weisstein. *Method of Steepest Descent*. Aug. 2011. URL: \url{http : / / mathworld . wolfram . com / MethodofSteepestDescent.html}.

[64] I. Winkler, S. Haufe, and M. Tangermann. "Automatic Classification of Artifactual ICA-Components for Artifact Removal in EEG Signals". In: *Behav Brain Funct* 7 (Aug. 2011), p. 30.

[65] Frank G. Yanowitz. *Characteristics of the Normal ECG*. May 2011. URL: \url{http://library.med.utah.edu/kw/ecg/ecg_outline/Lesson3/index.html}.