Detecting Limb Movements by Reading Minds

Lars Hiller Eidnes

Master i informatikk
Oppgaven levert: Desember 2011
Hovedveileder: Pinar Öztürk, IDI

Norges teknisk-naturvitenskapelige universitet

Problem Description

How can we most accurately predict whether or not a person is moving his or her limbs via an EEG-recording? This thesis will evaluate methods for preprocessing and classification of EEG-recordings, in order to detect a person's limb movements.

Abstract

By using EEG-recorders as input devices, the electrical activity on the scalp of humans can be used to control a computer. In later years, machine learning techniques have allowed these Brain-Computer Interfaces to be more accurate and to adapt to the individual person. In this thesis, the state of the art Sub-Band Common Spatial Patterns method for EEG classification was implemented, and then extended with the goal of improving its classification accuracy. In particular it was found that its accuracy can be improved by:

1. Employing Oracle Approximating Shrinkage in the calculation of the covariance matrix used by the Common Spatial Patterns algorithm.
2. Using the increase/decrease of the signal power over time as an additional feature.
3. Using L1-regularized Logistic Regression for classification and feature selection.
4. Boosting the final classifier.

Ultimately these alterations increased the classification accuracy from 86.4% to 91.7% on the BCI Competition III (IVa) dataset, and from 65.1% to 69.0% on the BCI2000 dataset.

Acknowledgements

This thesis has been developed with the valuable help of a number of people, who deserve credit. Advisors Gunnar Tufte and Pinar Öztürk have always been knowledgeable and motivating throughout this process. Ruud van der Weel and Øyvin Engan deserve thanks for introducing and explaining various technical and practical issues that arise with EEG machines and their recordings. Tor Ramstad was very helpful in explaining band pass filters, the various pitfalls to avoid when producing them, and the techniques for doing so.

Table of Contents

1 Introduction

1.1 Computerized interpretation of EEG

Electroencephalography (EEG) provides an insight into the firing patterns of the neurons in the brain by recording the electrical activity on the scalp. EEG is today mainly used in clinical contexts, for example to diagnose epilepsy, coma and brain death. Compared with other techniques for brain imaging, such as magnetic resonance imaging (MRI) and computed tomography (CT), EEG recordings have lower spatial resolution and higher temporal resolution.

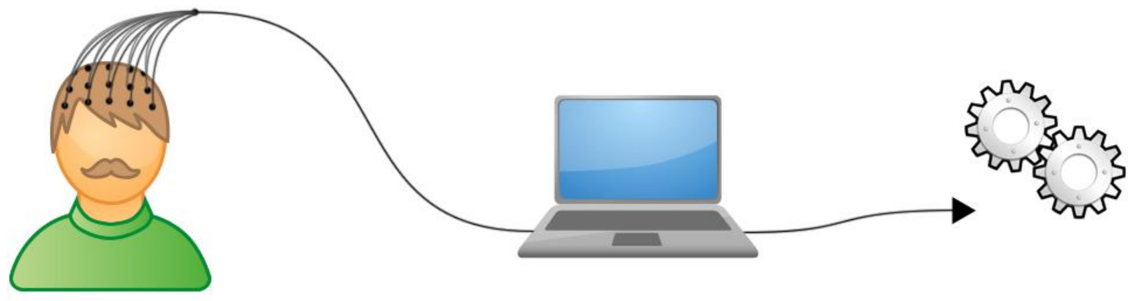
A novel application of EEG technology is its use in non-invasive Brain Computer Interfaces (BCI). A BCI is a device that tries to infer the intention or action of the user, based on his or her neural activity (Hild, et al., 2010). EEG-based devices are particularly attractive for this use because of their relatively low-cost and their portability. An EEG based BCI can have applications within many areas. One application for BCIs is to replace typical input devices for users with functional deficits that prevent them from using standard equipment. EEG-controlled spelling systems, for instance, have been developed to replace keyboards (Müller, et al., 2008). EEG-controlled wheel chairs have been developed for users who are unable to operate them with their hands (Rebsamen, et al., 2007). Outside the domain of health care, companies such as NeuroSky[1] and Emotiv Systems[2] are developing consumer level EEG products, and applying them to, for instance, computer games.

An early example of a BCI system is that of (Birbaumer, et al., 1999). There, severely paralyzed sufferers of the neurodegenerative disease amyotrophic lateral sclerosis (ALS) were enabled to communicate using a spelling system that was driven by EEG. The subjects learned to control a specific characteristic of their scalp potentials, which in turn gave a binary input to the spelling system. This process was slow; it could take a person more than an hour to spell 100 characters, and the training process could span several months. Later BCI systems have taken the approach of employing machine learning, so that the machine would learn to adapt to the particular subject, not the other way around. For such approaches, motor activity proved to be a good feature to detect. Since both imagined and actual motor activity uses the same region of the brain, motor activity is applicable even in BCIs designed for severely paralyzed people.

[1] NeuroSky is based in California, USA. http://www.neurosky.com/
[2] Emotiv Systems is of Australian origin, and based in Hong Kong. http://emotiv.com/

1.2 Main Problem

A complete BCI system needs to record EEG data, interpret it and apply this interpretation to control, for example, a wheelchair, a spelling system or a computer game. Figure 1.1 illustrates these components of a BCI system. The focus of this thesis will be exclusively on step 2, the interpretation of the EEG data. Further, this thesis will only deal with the detection of real or imagined motor activity in the brain of the user.



1) Electrodes, amplifiers and circuitry to record the electrical activity on the scalp.	2) Interpretation of EEG data, through signal processing and machine learning.	3) Output: Control of some software or device.

Figure 1.1: High level view of typical BCI architecture

The problem to be solved at step 2 of this process is to take raw EEG data from a short time span, process it, and arrive at a best guess of the user's motor activity. This function will be learned from labeled example data, and is thus an example of what within machine learning is called a supervised learning problem (Mitchell, 1997). The problem of EEG classification is typically attacked by employing several discrete stages of processing, as described in section 2.3 (page 14), before a classifier produces a final label for the given EEG-data (i.e. "right hand"). Concretely, for one of the datasets used in this thesis, we develop a function whose input is a matrix of 23600 floating-point numbers (200 samples per 118 channels) that will output a binary classification (one of two numbers).

The quality of such a system can then be evaluated by its ability to correctly label a new set of unlabeled data. The main goal of this work is to take one of the state-of-the-art methods for EEG classification, and extend and alter it in various ways in order to see if its accuracy can be improved.

1.3 Experimental Setup

The basis for this work is the Sub-Band Common Spatial Patterns (SBCSP) method (Novi, et al., 2007). The method is presented in detail in chapter 2.

The various extensions and alterations that will be done to the classification algorithm will be empirically evaluated on existing datasets. The evaluation procedure is described more in detail in section 4.1 (page 29), though the gist of it is simply to see what percentage of the data is correctly classified by the system. The specific methods that will be evaluated are presented in chapter **Feil! Fant ikke referansekilden.** (page **Feil! Bokmerke er ikke definert.**).

One data set is the publicly available BCI Competition III dataset IVa, which consist of imagined right hand and right foot movements recorded from 5 subjects (Blankertz, 2005), where the imagined movements where initiated by a visual cue.

Additionally, the dataset produced by the developers of the BCI2000 system (Schalk, et al., 2004) will be used for evaluation. This dataset contains recorded data from 109 subjects. The subjects were instructed to open (or alternatingly to imagine opening) their right or left fist, based on a visual cue. Other actions are also present in the dataset (foot movement), but these were not used in the experiments here. Note that to cut down the computational requirements, only the data from the first 50 people in this dataset are used in this thesis. Table 1.1 shows an overview of the datasets used in this thesis.

	BCI Competition III (IVa)	BCI2000
Subjects	5	109 (50 used)
Trials per subject	280	90
Activity	Foot and hand movement	Right and left fist movement
Sampling rate	100Hz	160Hz

Table 1.1: Overview of the characteristics of both datasets used in this thesis.

SBCSP itself, and other alterations of it (i.e. (Ang, et al., 2008)), have previously been evaluated against the BCI Competition III (IVa) dataset. By using the dataset in this thesis as well, we can compare the results here against previous work. Additionally testing against the BCI2000 dataset allows us to see whether the results obtained generalize to new datasets.

1.4 Brief Summary of Experiments

Building upon the Sub-band Common Spatial Patterns (SBCSP) method, we will extend and alter the system in the certain ways, and see what this does with the classification accuracy:

Common Average Referencing: This method is intended to improve signal-to-noise ratio, and will be evaluated for use with the SBCSP.

Covariance Estimation: An early step in the method is the Common Spatial Patterns (CSP) algorithm, which requires estimates of particular covariance matrices. We will assess various ways of estimating covariance matrices, with and without regularization, for use by the CSP algorithm.

Classification: The final step of the SBCSP method is classification by a classifier. We will evaluate a selection of linear and non-linear classifiers for this purpose.

Features: The classifier finally does its classification based on a vector of numbers that describe the EEG data segment. This representation is called the feature vector, and we will investigate different variations of this representation.

Feature Selection: The SBCSP method defines a particular procedure to reduce the number of features that represent the EEG data. We will experiment with alternative approaches to achieve the same goal, as well as alternative numbers of features to be used.

Boosting: We will iteratively retrain classifiers on the dataset, such that previously misclassified examples are weighted as being more important in subsequent rounds. This is known as boosting, and has been shown to improve classification accuracy in certain cases previously.

Bandpass Filter Optimization: The SBCSP method requires the extraction of certain frequency ranges from the data. This is done through a filter defined by several parameters. We will do a rough search through this parameter space.

1.5 Document Overview

This first chapter has presented the context of EEG based BCIs, and the questions that this thesis intends to answer.

Chapter 2 presents the physiological and mathematical background for the software system. The physiological phenomena that the BCI system intends to detect are briefly presented, followed by the algorithms typically used to detect them.

Chapter 3 shows some existing results on the BCI Competition III (IVa) dataset, including the results of SBCSP, and the results obtained by the reimplementation of the method produced for this thesis.

Chapter 4 presents each of the experiments done in the context of this thesis. Each experiment will be presented separately, followed by their results and a brief discussion of these results. Bit by bit we will use the results obtained to build an improved classifier.

Chapter 5 is the conclusion, containing the main findings of this thesis, followed by some proposals for future work in this area.

2 Physiological and Mathematical Background

2.1 Fundamental Difficulties

All EEG based approaches to Brain Computer Interfacing have to deal with some fundamental difficulties. In many forms of research on the brain, it is common that the researcher averages EEG signals from several hundred recordings in order to accentuate any pattern in the data. This technique cannot generally be used in a BCI however, since a BCI must produce immediate classifications of the current activity of the brain.

The EEG records electrical activity on the scalp, which means that the sensors have to observe the neurological action at a distance. Since the sensors are removed from the origin of the signals of interest they will be mixed with signals irrelevant to the activity we want to detect, arising both from the brain and from the environment. A further complication is that while an EEG recorder can produce hundreds of thousands of electrode samples per second, obtaining example data involves inconveniencing a human being for some period of time. This means that the classification problem is one with high-dimensional data and few training examples.

Fortunately, researchers have come up with several approaches to overcoming these difficulties – some of these will be described and evaluated later in this document.

Another complication is that motor activity presents itself differently from person to person. In particular, there is a variation in the frequency at which the data is most discriminable. This means a classifier that has been trained on one person will not in general work very well on other people. Even on the same person, the act of removing and rewearing the EEG device may reposition the electrodes such that the classifier has to be retrained to work properly. The problem of applying previously learned information to new sessions and new people is known as subject-to-subject transfer. This topic is investigated in for example (Krauledat, et al., 2007) and (Kang, et al., 2009), but will not be explored in this thesis.

2.2 What We Are Looking For

It is known that both real and imagined motor activity cause an observable change in the EEG reading of the brain, called Event Related Desynchronization (ERD). In EEG *synchrony*, neurons are oscillating in phase, which means the individual contributions of each neuron can add up and thus produce higher amplitudes in the EEG recording. Therefore *desynchronization* is associated with decreased amplitudes. It is this change in amplitude over the relevant areas of the motor cortex that we try to detect. The change can only be observed at certain frequency ranges. An introduction to the basic principles of Event Related Desynchronization and Synchronization can be found in (Pfurtscheller, et al., 1999).

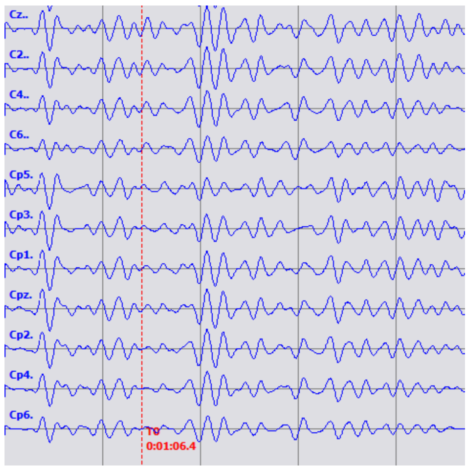


Figure 2.1: Some EEG channels during a movement

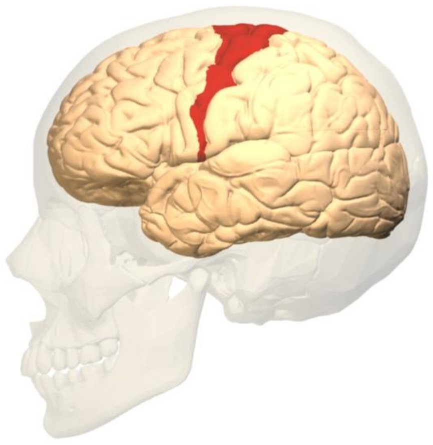


Figure 2.2: Highlight of the Motor Cortex of the human brain

Where the ERD occurs on the brain is critical in distinguishing which activity occurred. While motor activity will be detectable over the motor cortex (shown in figure 2.2), a right hand movement will cause ERD on the left side of the brain, and vice versa. Figure 2.1 shows a plot of some EEG channels during a movement, where the dotted red line indicates the start of a hand movement.

2.3 State of the Art: The General Approach

Most approaches to EEG classification of motor imagery follow a certain general pattern. Typically, the process starts by extracting certain frequencies from the data, followed by a spatial filtering and concluded by a classification by a linear classifier, as illustrated in figure 2.3. The reasoning behind this general approach will be elucidated below.

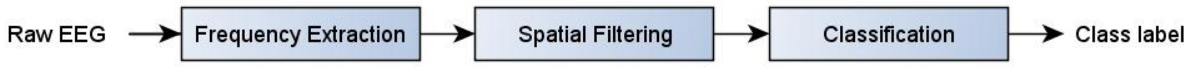


Figure 2.3: Illustration of the typical dataflow in an EEG classification system

2.3.1 Frequency Range Extraction

The amplitude change associated with ERD can be detected in what is traditionally called the mu (8-12Hz) and beta (12-30Hz) rhythms (Wang, et al., 2006). Unfortunately for the designer of an EEG classifier, the specific frequencies at which it can be optimally detected vary from subject to subject. Several approaches have been suggested to remedy this.

CSP: A common approach is that an expert tunes the frequency range for each subject by hand, typically by visually inspecting the resulting spatial patterns produced by the Common Spatial Patterns (CSP) algorithm (see section 2.6, page 18). This approach is typically called simply a CSP based classifier. The required manual tuning makes this approach cumbersome and impractical for many use cases.

CSSP: In the Common Spatio-Spectral Patterns algorithm (Lemm, et al., 2005), a simple Finite Impulse Response (FIR) filter is optimized along with the spatial filter. This extracts a single frequency band, and does not require manual tuning.

CSSSP: The Common Sparse Spectral Spatial Patterns algorithm (Dornhege, et al., 2006) optimizes an arbitrary FIR-filter along with the spatial filter. However, the optimization problem is non-convex and thus may get stuck in a local optimum.

SBCSP: The Sub-band Common Spatial Patterns algorithm takes the approach of simply including several 4Hz frequency bands, and lets a classifier determine which of these bands contain discriminative information. It is this method that is extended in this thesis.

A positive side effect of extracting frequency ranges is that it serves to remove – to some degree – the noise and artifacts that are caused by events such as eye-blinks.

2.3.2 Spatial Filtering

In addition to knowing what frequencies to look at, we want to know where on the brain we should look in order to discriminate between different motor activities. To do this we employ a *spatial filter*, which is a weighing of each EEG channel, telling us where to look in order to discriminate between the activities. These can be visualized by means of a scalp map, as in figure 2.4.

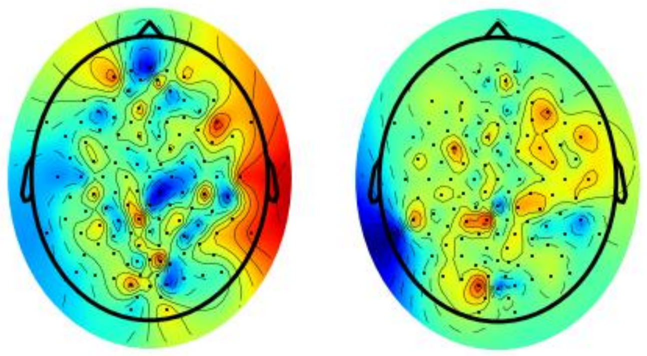


Figure 2.4: The scalp maps shows two spatial filters obtained from a subject from the BCI Competition 2003 dataset, using the CSP method. These two maps maximize the variance of one class while minimizing the variance of the other. Together, they cover both classes of activity.

It may be surprising how non-smooth these spatial filters are. After all, a specific motor activity will produce ERD originating at a specific location in the motor cortex, which will diffuse smoothly through the volume of the brain. If one where to plot this propagation – that is a forward-model relating the source signals to the scalp signals – one would see a smooth plot with large values around the areas of the motor cortex related to the specific activities. Such models are called *spatial patterns*. However, the *spatial filter* is calculated in order optimize the discriminability of two or more types of signal, and must also account for noise in the data. This will typically result in a more complex structure of the filter. (Blankertz, et al., 2010) give a more in depth argument for why this is the case, along with a numerical simulation showing how this can occur.

2.3.3 Linear Classification

Commonly, the final classification of the extracted features is done by a linear classifier. A particularly favored approach is classification by Linear Discriminant Analysis (LDA). (Lotte, et al., 2010) reviewed several linear and non-linear classifiers, and found that LDA and Support Vector Machines (SVMs) tended to give good results. On certain datasets SVMs with non-linear kernels outperformed linear SVMs. See section 4.4.2, page 33 for a description of SVMs and kernels.

The accuracies obtained by the various classifiers depend heavily on two factors: The features that are extracted from the data, and the number of training examples that are available. There is no reason to expect that all features one can generate from EEG data will be linearly separable. However, with EEG data the number of training examples from a particular subject will typically be small. Since non-linear models typically require more parameters to describe than linear models, they are harder to fit to the data, and are less likely to generalize well when the number of training examples is sufficiently small.

2.4 Notation

In order to improve readability, a common notation will be used in the mathematical descriptions of the methods presented in this chapter. Matrices will be denoted with a bold, capital letter (such as $\mathbf{X}$), vectors will be denoted in bold, lower case letters (such as $\mathbf{y}$), while scalars are written in non-bold, lower case letters (such as z). Indexing into vectors and matrices will be denoted as a superscripted number in parenthesis, so $\mathbf{y}^{(\mathrm{i})}$ denotes the i-th element of the vector $\mathbf{y}$.

2.5 Sub-Band Common Spatial Patterns

The classification system reproduced and extended in this work is called Sub-Band Common Spatial Patterns (SBCSP), and was introduced in (Novi, et al., 2007). This model was further developed in (Ang, et al., 2008) and (Ang, et al., 2009).

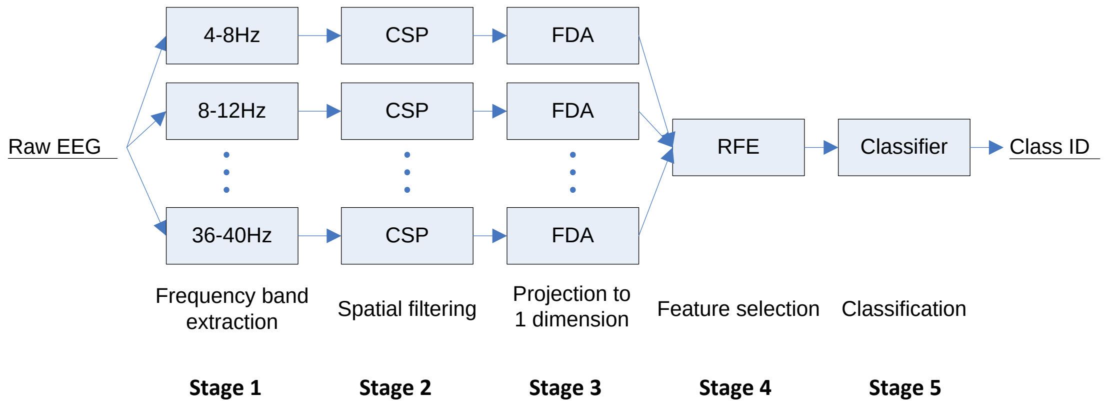


Figure 2.5: The SBCSP architecture.

This section will give a high level presentation of the steps involved in the SBCSP method, while the detailed description of each step will be given in subsequent chapters. Figure 2.5 shows a high level view of the architecture and the dataflow of the method.

It may be useful to take note of the size of the data as it passes through each step. Initially each EEG-trial is represented as a matrix $\mathbf{X}$, with dimensions $c \times s$, where c is the number of channels, and s is the number of samples. For the BCI competition III (IVa) data that the SBCSP was evaluated on, $c = 118$, $s = 350$. Each trial was 3.5 seconds long (sampled at 100Hz). The SBCSP method looks only on the samples from 0.5-2.5 seconds, thus reducing s to 200.

Stage 1: Frequency band extraction

First, the raw EEG-data is split into frequency bands of 4Hz each. This is done by adapting a Gabor filter as a bandpass filter, and convoluting this with the EEG signal. The output of this stage is a frequency filtered $c \times s$ matrix for each frequency range.

Stage 2: Spatial filtering

Each subband is then used to train the Common Spatial Patterns (CSP) algorithm (see section 2.6, page 18 for an in depth description of CSP). This supervised learning algorithm produces a $\mathrm{c} \times \mathrm{c}$ projection matrix $\mathbf{M}$, which we can use to project the data in a way that maximizes the variance of one class while minimizing the variance of another. The projection matrix defined by the first r and the last r rows of $\mathbf{M}$ give us the projection that contain the most discriminative information. Typically, the chosen value of r is 1.

Next, we use as features the logarithm of the variance of the signal in each of the $2\mathrm{r}$ most significant projection directions, after normalizing the values by dividing by their sum.

$$f_{log-variance} = \left\langle \log\left(\frac{\mathrm{var}(\tilde{\mathbf{Z}}^{(1)})}{\sum_{k=1}^{2r} \mathrm{var}(\tilde{\mathbf{Z}}^{(k)})}\right), \dots, \log\left(\frac{\mathrm{var}(\tilde{\mathbf{Z}}^{(2\mathrm{r})})}{\sum_{k=1}^{2r} \mathrm{var}(\tilde{\mathbf{Z}}^{(k)})}\right)\right\rangle$$

Here $\tilde{\mathbf{Z}}$ denotes the EEG data after projection into the $2\mathrm{r}$ most significant projection directions found by the CSP algorithm.

Thus this step reduces the $\mathrm{c} \times \mathrm{s}$ matrix for each subband to a vector of $2\mathrm{r}$ numbers.

Stage 3: Projection to one dimension

The $2\mathrm{r}$ numbers from the CSP stage is then projected to one dimension using the projection defined by Linear Discriminant Analysis (see section 2.7.1, page 20). This aims to increase separability of 2-classes of data, by finding the projection that maximizes the ratio of between-class variance to within-class variance. The resulting projection matrix projects the data to one dimension, a single number.

Stage 4: Feature selection

Each number produced by the LDA projection in each subband are then concatenated to a feature vector that describes the trial. (Novi, et al., 2007) evaluate two feature selection approaches, one based on Support Vector Machine Recursive Feature Elimination (SVM-RFE), and one based on combining a Bayesian feature ranking with a SVM classifier.

The SVM-RFE approach uses an SVM classifier to rank the features by their predictive power. An SVM classifier finds a hyperplane that separates two classes, while maximizing the margins between the hyperplane and each class. This hyperplane is defined by a weight vector $\boldsymbol{\theta}$, and an offset parameter b. Each weight corresponds to a specific feature. Recursive Feature Elimination works by iteratively training an SVM, removing the feature with the lowest weight, and repeating until the set of features is empty. This effectively ranks the features by their predictive power, and we can choose the top n features for the final classification. Good values for n can be empirically determined, and was set to 10 in the SBCSP method.

Stage 5: Classification
With the top n features selected, these are finally fed into a classifier which determines the class of the data. The SBCSP paper proposes using a linear SVM for this classification.

2.6 Common Spatial Patterns

2.6.1 Introduction
The Common Spatial Patterns (CSP) algorithm was originally introduced in (Fukunaga, et al., 1970) and became known as the Fukunaga-Koontz transform. It was since introduced to the context of EEG-analysis in (Koles, 1991), and eventually became known as the Common Spatial Patterns algorithm in the EEG context.

The main idea is to produce a transformation of the multi-channel EEG-data to a lower dimensional subspace such that the variance of each class of data is contrasted. The transform contrasts the data by simultaneously maximizing the variance of one class while minimizing the variance of the other, and vice versa, thus increasing separability. The method can be seen as an extension of the well-known method of Principal Component Analysis, first developed in (Pearson, 1901). It has been very successful in EEG classification because ERD manifests as changes in the signal power, and because signal variance and signal power are essentially equivalent measures in band pass filtered EEG data (see section 4.5, page 36 for more on this relationship).

2.6.2 Description
The CSP method performs a whitening of the data (decorrelating, while maintaining variance), and then rotates the data to contrast the within-class variance, as illustrated in figure 2.6.

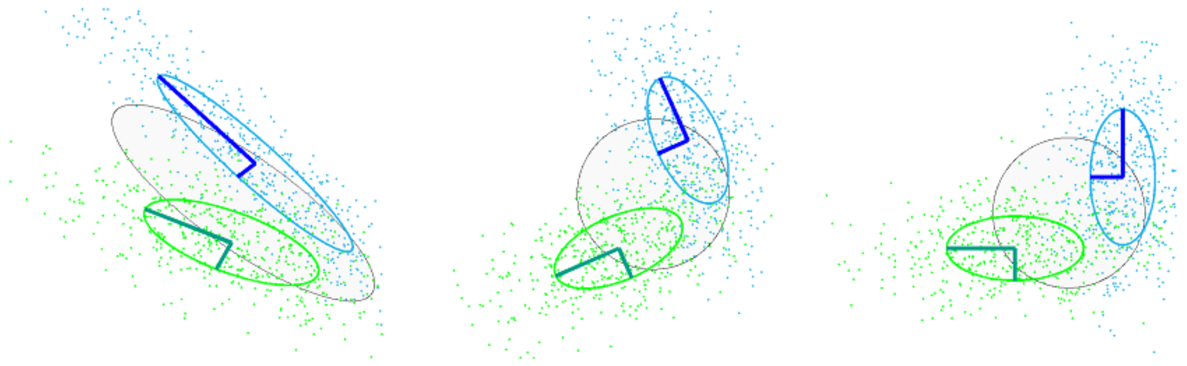


Figure 2.6: The steps of CSP: The blue and green ellipses correspond to the within-class covariance matrices along with the principal axes, while the mutual covariance is depicted in grey. Left: The raw EEG data. Center: The data after whitening. Right: The data after rotating to contrast within-class variance. Picture from (Lemm, et al., 2011).

In the following description we have the EEG data of each trial in matrices $\mathbf{X}$, each with dimension $c \times s$, where c is the number of channels and s is the number of samples. Let $\mathbf{X}_\mathrm{a}$ and $\mathbf{X}_\mathrm{b}$ denote EEG from classes a and b respectively.

The normalized spatial covariance of a trial can be calculated as[3]:

$$\mathbf{C}_\mathrm{a} = \frac{\mathbf{X}_\mathrm{a}\mathbf{X}_\mathrm{a}^\top}{\mathrm{trace}(\mathbf{X}_\mathrm{a}\mathbf{X}_\mathrm{a}^\top)} \qquad \mathbf{C}_\mathrm{b} = \frac{\mathbf{X}_\mathrm{b}\mathbf{X}_\mathrm{b}^\top}{\mathrm{trace}\left(\mathbf{X}_\mathrm{b}\mathbf{X}_\mathrm{b}^\top\right)}$$

Where $\mathbf{X}^\top$is the transpose of $\mathbf{X}$, and $\mathrm{trace}(\mathbf{A})$ sums of the diagonal elements of $\mathbf{A}$. We then find these matrices for each trial and average them for each class, obtaining $\overline{\mathbf{C}_\mathrm{a}}$ and $\overline{\mathbf{C}_\mathrm{b}}$, the averaged spatial covariance matrices for each class. Next, we sum these matrices, and decompose the sum into eigenvalues and eigenvectors:

$$\bar{\mathbf{C}} \ = \overline{\mathbf{C}_\mathrm{a}} + \overline{\mathbf{C}_\mathrm{b}} = \mathbf{U}_0\mathbf{\Sigma}\mathbf{U}_0^\top$$

Where $\mathbf{U}_0$ is the matrix of eigenvectors and $\mathbf{\Sigma}$ is the diagonal matrix of eigenvalues. We sort these two matrices by eigenvalues in descending order. We can then find the whitening transformation matrix as:

$$\mathbf{W} = \ \mathbf{\Sigma}^{-1/2}\mathbf{U}_0^\top$$

The whitening transforms $\bar{\mathbf{C}}$ into the identity matrix, $\mathbf{W}\bar{\mathbf{C}}\mathbf{W}^\top = \mathbf{I}$. It transforms the average spatial covariance matrices as:

$$\mathbf{S}_\mathrm{a} \ = \ \mathbf{W}\overline{\mathbf{C}_\mathrm{a}}\mathbf{W}^\top \qquad \mathbf{S}_\mathrm{b} \ = \ \mathbf{W}\overline{\mathbf{C}_\mathrm{b}}\mathbf{W}^\top$$

$\mathbf{S}_\mathrm{a}$ and $\mathbf{S}_\mathrm{b}$ have the properties that they share common eigenvectors, and the sum of corresponding eigenvalues will always be one:

$$\mathbf{S}_\mathrm{a} \ = \mathbf{U}\mathbf{\Sigma}_\mathrm{a}\mathbf{U}^\top \qquad \mathbf{S}_\mathrm{b} \ = \mathbf{U}\mathbf{\Sigma}_\mathrm{b}\mathbf{U}^\top \qquad \mathbf{\Sigma}_\mathrm{a} + \mathbf{\Sigma}_\mathrm{b} = \mathbf{I}$$

So if $\boldsymbol{\sigma}$ is an eigenvector of $\mathbf{S}_\mathrm{a}$ with eigenvalue u, it is also an eigenvector of $\mathbf{S}_\mathrm{b}$ with eigenvalue $(1 - \ \mathrm{u})$. Thus the most important eigenvector for one class is the least important for the other class, and vice versa. The transformation by these eigenvectors gives us the optimal separation of the variance of each class. By combining the whitening with these eigenvectors we obtain the transformation matrix:

$$\mathbf{M} = (\mathbf{U}^\top\mathbf{W})^\top$$

Finally we can transform the raw EEG trial data $\mathbf{X}$ by

$$\mathbf{Z} = \mathbf{M}\mathbf{X}$$

[3] Alternative ways of calculating covariance matrices for this step are explored in section 4.3, page 30.

The top and bottom rows of $\mathbf{Z}$ correspond to the top and the bottom of the eigenvalue spectrum, and are the directions that contrast the two classes of data. We can use the variance of these rows as features for classification.

2.7 Linear Classification

2.7.1 Linear Discriminant Analysis

One way to approach linearly separating data is to look at it as dimensionality reduction problem. If we project our length n feature vector down to one dimension, we get a vector of length 1 – in other words a single number. For two-class classification, the problem of classification is then as simple as picking a suitable number on the number line and claiming that numbers larger than it are of one class, and of the other class otherwise. This is the approach taken in Linear Discriminant Analysis (LDA).

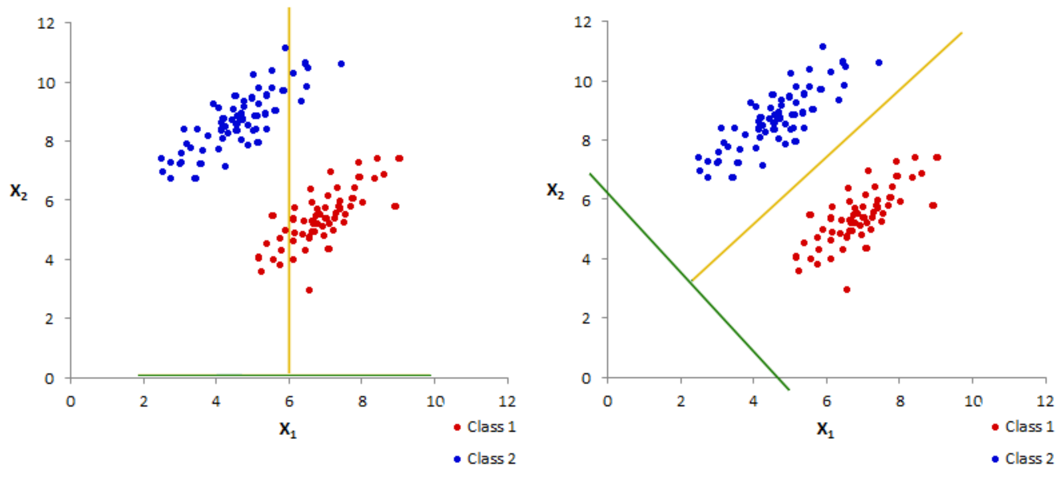


Figure 2.7: Illustration of two choices of projection directions. Left: The dataset is projected down to the X1 axis, giving poor separation of the classes. Right: The data is projected using Fishers criterion, giving good separation of the classes.

We can project our feature vector $\mathbf{x}$ down to one dimension using a transformation vector $\boldsymbol{\theta}$:

$$y = \boldsymbol{\theta}^{\top}\mathbf{x}$$

This is what is done by both LDA and linear regression, and it is the essence of what is done by logistic regression, although all methods choose different values of $\boldsymbol{\theta}$. When we look at $\boldsymbol{\theta}$ as a projection, we want to choose the projection that maximizes the separability of the classes. The idea proposed by (Fisher, 1936) was to choose $\boldsymbol{\theta}$ so as to maximize variance between the classes, while minimizing the variance within each class. Such a projection is shown in figure 2.7. The mean $\boldsymbol{\mu}_{\mathrm{K}}$ of a class K is given by:

$$\boldsymbol{\mu}_{\mathrm{K}} = \frac{1}{m}\sum_{\mathbf{x}\in \mathrm{K}} \mathbf{x}$$

The between-class covariance matrix is given by:

$$\mathbf{B} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\top}$$

The within-class covariance matrix for class K is given by:

$$\mathbf{W}_K = \sum_{\mathbf{x} \in K} (\mathbf{x} - \boldsymbol{\mu}_K)(\mathbf{x} - \boldsymbol{\mu}_K)^\top$$

We can then define the separability of two classes as the ratio of the between-class variance to within-class variance. Thus we want to choose the projection $\boldsymbol{\theta}$ so as to maximize:

$$\frac{\boldsymbol{\theta}^\top \mathbf{B} \boldsymbol{\theta}}{\boldsymbol{\theta}^\top (\mathbf{W}_1 + \mathbf{W}_2) \boldsymbol{\theta}}$$

We can differentiate this with respect to $\boldsymbol{\theta}$, find the maximum, simplify the result and obtain:

$$\boldsymbol{\theta} \propto (\mathbf{W}_1 + \mathbf{W}_2)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

We then have the projection we want, and need to decide on the threshold which separates the classes. There is no one given choice here, but a common choice is to put the threshold in middle of the means of the two classes.

This discriminant gives the Bayes optimal prediction if two conditions are satisfied: The feature vectors must be normally distributed, and the classes must have equal covariance. Unfortunately, in the case of EEG classification these conditions are only approximately met.

2.7.2 Logistic Regression

Like ordinary linear regression, the logistic regression classifier fits a linear model to the data. However, while linear regression predicts a continuous valued target variable, logistic regression predicts a discrete valued target variable. While one can easily discretize the output of a linear regression in order to use it as a classifier, logistic regression allows us to achieve better accuracy by specifically optimizing the model to minimize classification error.

In linear regression, the learned function $\mathrm{h}_{\boldsymbol{\theta}}(\mathbf{x})$ is determined by linear combination of the features $\mathbf{x}$ and the parameters $\boldsymbol{\theta}$:

$$\mathrm{h}_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 \mathbf{x}_1 + \cdots + \boldsymbol{\theta}_n \mathbf{x}_n$$

For conciseness, we can define $x_o = 1$, and write:

$$\mathrm{h}_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}_0 \mathbf{x}_0 + \boldsymbol{\theta}_1 \mathbf{x}_1 + \cdots + \boldsymbol{\theta}_n \mathbf{x}_n = \boldsymbol{\theta}^\top \mathbf{x}$$

In order to make the output fall inside the range $\langle 0, 1 \rangle$, we wrap the linear regression model in the sigmoid function (also called the logistic function), denoted here by $\mathrm{s}(\mathrm{z})$. The sigmoid function has an output between 0 and 1, and is exactly 0.5 when its parameter is 0, as shown in figure 2.8.

$$\mathrm{s}(\mathrm{z}) = \frac{1}{1+e^{-z}}$$

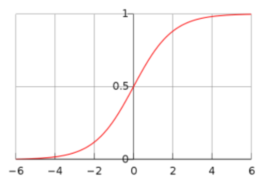


Figure 2.8: Plot of the sigmoid function

Thus the following is the model for logistic regression:

$$\mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathrm{s}\big(\mathrm{h}_{\boldsymbol{\theta}}(\mathbf{x})\big) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\top}\mathbf{x}}}$$

For this model to be useful, we need to find good values of the parameters $\boldsymbol{\theta}$. To do this, we can define an error function $\mathrm{error}(\mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}), y)$, where y is the true class label of $\mathbf{x}$, and find good values of the parameters by minimizing the error function. The standard squared error function is not a good choice in this case, because the resulting function would be non-convex and therefore hard to minimize. Therefore we choose the following error function for logistic regression:

$$\mathrm{error}(\mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}), y) = \begin{cases} -\log \mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}), & \mathrm{y} = 1 \\ -\log(1 - \mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x})), & \mathrm{y} = 0 \end{cases}$$

$$= -y \log \mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}) - (1 - y) \log(1 - \mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}))$$

This function has the reassuring property that its value approaches infinity as the prediction approaches the wrong value, while it approaches zero as the prediction approaches the right value.

Our final optimization objective $\mathrm{J}(\boldsymbol{\theta})$ then is the average error over all the m examples

$$\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=0}^{m} \mathrm{error}\big(\mathrm{L}_{\boldsymbol{\theta}}\big(\mathbf{x}^{(\mathrm{i})}\big), y^{(\mathrm{i})}\big)$$

$$= \frac{1}{m}\sum_{i=0}^{m} \big[-y^{(\mathrm{i})} \log \mathrm{L}_{\boldsymbol{\theta}}\big(\mathbf{x}^{(\mathrm{i})}\big) - \big(1 - y^{(\mathrm{i})}\big) \log\big(1 - \mathrm{L}_{\boldsymbol{\theta}}\big(\mathbf{x}^{(\mathrm{i})}\big)\big)\big]$$

Minimizing this function yields the Maximum Likelihood Estimate of the regression parameters. The function can be minimized by, for example, Gradient Descent or Newton's Method.

2.7.3 Regularized Logistic Regression

When training a classifier, we might run into problems where the model fails to generalize to new examples, because the learned model is too complex. This is known as overfitting (Mitchell, 1997). This is particularly likely to occur when we have few training examples relative to the number of features, as is often the case with EEG data. In logistic regression, one way to reduce the complexity is to make each of the parameters in $\boldsymbol{\theta}$ *matter less*, by reducing their size. A measure of the size of a vector is called a norm, and it can be defined in many ways. Here are two examples that are useful in this context:

$$\mathrm{L1}(\mathbf{x}) = \frac{1}{n}\sum_{i=0}^{n} |\mathbf{x}_{\mathrm{i}}| \qquad\qquad \mathrm{L2}(\mathbf{x}) = \frac{1}{n}\sum_{i=0}^{n} \mathbf{x}_{\mathrm{i}}^{2}$$

These are called the L1-norm and L2-norm of a vector. By adding $\mathrm{L1}(\boldsymbol{\theta})$ or $\mathrm{L2}(\boldsymbol{\theta})$ as a term to our minimization objective, it will be minimized along with the error function, thus restricting the growth of the regression parameters. However, we need to decide on the degree to which we wish to restrict growth of the parameter vector. We can make this trade-off explicit by weighting the norm with a parameter λ. For example if we pick the L2 norm, we have:

$$\mathrm{J}(\boldsymbol{\theta}) = \left(\frac{1}{m}\sum_{i=0}^{m} \mathrm{error}\left(\mathrm{L}_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathrm{i})}\right), y^{(\mathrm{i})}\right)\right) + \frac{\lambda}{2m}\mathrm{L2}(\boldsymbol{\theta})$$

By dividing λ with the number of training examples m, we get the nice property that the degree of regularization diminishes as the number of training examples grow larger.

3 Existing Results and Reproduction

3.1 Existing Results

3.1.1 Results of Sub-Band Common Spatial Patterns

(Novi, et al., 2007) presented results of the SBCSP method over the BCI Competition III (IVa) dataset. Table 3.1 shows the accuracies of the various methods, after a 10-fold cross-validation (see section 4.1.1, page 29 for a description of cross-validation).

Subject	CSP	CSSP	CSSSP	SBCSP-MC	SBCSP-RFE
aa	91.5±5.4	85.4±6.2	88.4±6.3	89.3±5.6	90.8±4.5
al	99.2±1.8	97.7±3.0	97.9±2.7	98.6±1.8	97.8±3.4
av	70.9±8.2	67.4±7.6	68.2±7.7	70.4±5.3	69.0±7.3
aw	96.9±2.8	96.5±3.3	93.5±4.3	95.7±4.0	95.8±3.3
ay	94.7±3.8	94±3.9	89.5±5.7	95.7±2.8	95.0±3.4
Average	90.6	88.2	87.5	90.0	89.7

Table 3.1: Result from (Novi, et al., 2007), of the SBCSP method over the BCI Competition III (IVa) dataset.

Here CSP, CSSP and CSSSP are single band methods where only a single frequency band is extracted (see section 2.3.1, page 14 for a detailed description). The column for SBCSP-MC in the chart is for the SBCSP variant where the results from each sub-band is classified by a Bayesian “Meta-Classifier” is used as a preprocessing step before it is fed to a SVM classifier. Finally, the SBCSP-RFE column contains the results for the SBCSP variant that does Recursive-Feature Elimination before doing SVM classification.

3.1.2 Results of Filter Bank Common Spatial Patterns

(Ang, et al., 2008) proposed choosing different feature selection and classification mechanisms for the SBCSP method. Several mechanisms were empirically evaluated, and the two variants with the most promising results were labeled FBCSP$_w$ and FBCSP$_f$. Their results on the BCI Competition III dataset IVa are shown in the table 3.2.

	CSP (Ang)	SBCSP (Ang)	FBCSP-w	FBCSP-f
Average	86.6	86.3	89.2	90.3

Table 3.2: Experimental results from (Ang, et al., 2008).

3.2 Reproduction

In order to do experiments for this thesis, the SBCSP method was reimplemented. In particular, the variant using SVM based Recursive Feature Elimination was implemented (SBCSP-RFE). The results obtained by this method are shown in table 3.3, over both the BCI Competition III (IVa) dataset, and the BCI2000 dataset. A summary of the results of all methods is given in figure 3.2.

	SBCSP (This thesis)
Average accuracy over the BCI Competition III (IVa) dataset	86.4%
Average accuracy over the BCI2000 dataset	65.1%

Table 3.3: Accuracy of the SBCSP reproduction over two datasets.

The lower accuracy over the BCI2000 dataset can be explained by the significantly lower number of training examples per person in this dataset.

3.3 Discrepancies

As can be seen, the results given for the SBCSP method are different between (Novi, et al., 2007), (Ang, et al., 2008) and the new reproduction in this thesis. The differences can be explained by the following implementation differences:

1) Novi, et al. extract 24 frequency bands, each with a bandwidth of 4Hz. Ang, et al. extract 9 bands (the bandwidth is not specified). The reproduction in this thesis extracted 22 bands.[4]
2) Novi, et al. uses a set of Gabor filters to extract frequency ranges, while Ang, et al. uses a set of Chebyshev Type II filters. This thesis uses a set of FIR-filters, in order to facilitate the experiments done in section 4.8. These filter types have different properties and will give slightly different results.
3) Unspecified bands: In the previous SBCSP productions, it has not been stated which specific frequency bands were extracted from the data.
4) Setting of filter parameters: The specific parameters used for the filters are not stated. These parameters determine exactly what data is extracted from the raw EEG data, which naturally affects the results obtained. The plots in figure 3.1 exemplify this.

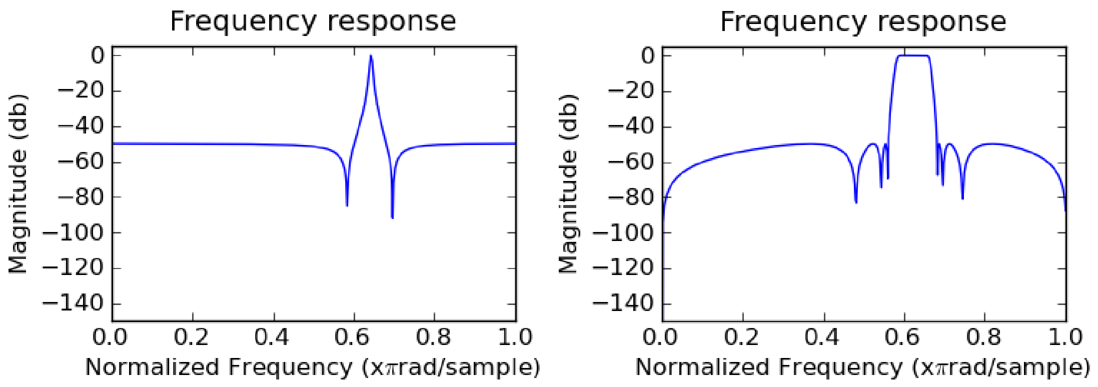


Figure 3.1: The frequency response of two Chebyshev Type II filters, which attempt to extract the same frequency range. To the left, a naïve parameter setting results in only fully extracting a very narrow band. To the right, a broader band is extracted.

The filter settings can have a dramatic effect on the accuracy of the method, however there is no generally agreed upon procedure for how to set these parameters. Good results may be obtained by doing an exhaustive search over the parameters, however this carries a significant risk of overfitting the parameters to the particular dataset. In other words, the set of parameters that give optimal results for one dataset may not give the optimal results for other datasets. This will be further investigated in section 4.8 (page 46).

[4] 22 bands of 4Hz were extracted, because they can span 2-48Hz by having an overlap of 2Hz. Covering the 0-2Hz band and 48-50Hz band would make some experiments in section 4.8 impossible due to technicalities of FIR-filter design.

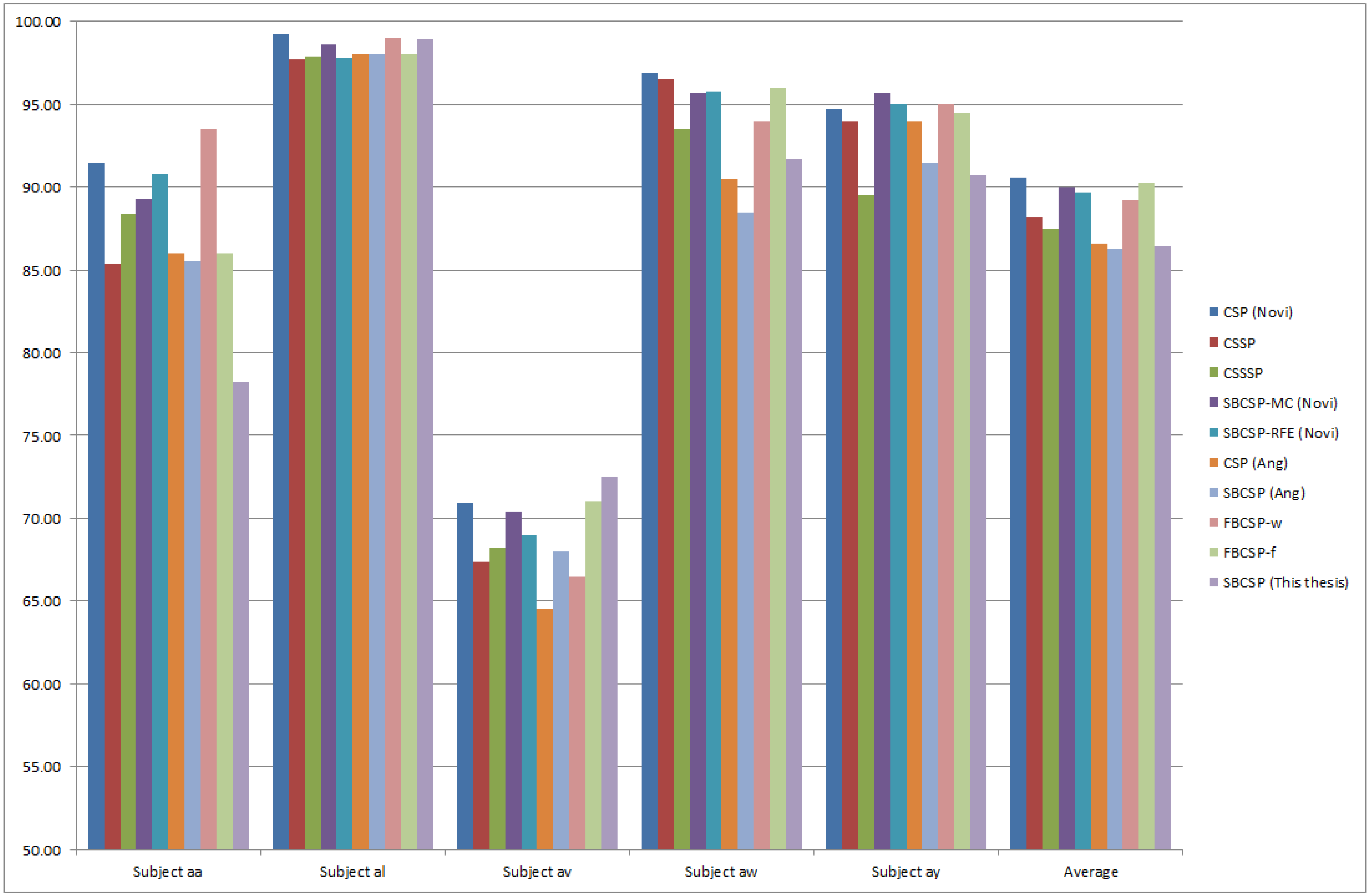


Figure 3.2: Bar chart showing the accuracies obtained for each subject by the various methods over the BCI Competition III (IVa) dataset.

4 Experiments and New Results

4.1 Evaluation

4.1.1 Cross Validation

The classification accuracies of the various approaches are validated in this section by a variant of 10-fold cross validation, where the original order of the data is maintained. Each dataset, containing n examples, is split in 10 parts (folds), each containing $n/10$ examples. This split is done while maintaining the order in which they were recorded, such that each fold contains examples that were recorded one after another from the subject.   One of these folds is then used as test data, while the remaining folds are used to train the model. The accuracy of the classifier is then measured over the test set. This process is then repeated 10 times, such that every example in the dataset appears in the test set exactly once. The mean of the 10 calculated accuracies is then used as the final assessment of accuracy.

This methodology differs from typical k-fold cross validation in that it preserves the order in which the data was sampled. Typically, the data is shuffled before being split into folds, such that each fold contains a randomly sampled subset of the data. There are two reasons randomization is not done in this work. First, since it is known that EEG-data is non-stationary – effects such as subject fatigue can cause the data to change over time – we believe that randomly sampling the example data will gloss over the effects of such changes, and may consequently give an unrealistic picture of real world performance. Second, not using random samples makes the results given here reproducible.

4.1.2 Comparing Methods

In order to do a statistically rigorous comparison of classification methods, a typical approach is to view the mean accuracy as a *sample mean* estimating some *true mean* accuracy of the method. By cross-validation we can then find confidence interval for the mean, and compare the means of two methods by a paired t-test.

This is, however, much less straight-forward for the experiments in this thesis. Both datasets used in these experiments contain multiple subjects. So while we can find the mean accuracy per person, the *mean of those means* is the true measure of the accuracy of a method. Calculating a confidence interval on this mean of means is an estimation problem in itself, known as a linear mixed effects model. (Jarŝová, 2010) performs a numerical simulation comparing three methods for approximating a confidence interval in such a setting. The conclusion of the study is that such approximations can be highly unstable and are therefore often worthless. Because of these problems, methods in this thesis are not compared with paired t-tests, as it would be unclear how such tests should be interpreted.

In order to still be able to make some comparison between methods we will perform cross validation across two different datasets. We will first run all experiments on the BCI Competition III (IVa) dataset, and then run the same tests on the BCI2000 dataset. If a method that appeared better on the 5 subjects in the former dataset still outperforms other methods on the 50 subjects in the latter, we can be quite confident this difference is not due to chance.

4.2 Common Average Reference

4.2.1 Introduction

A very simple approach to denoising an EEG signal is to apply a common average reference. The idea is simply to subtract from each channel the average of all channels at that point in time.

$$x_i \coloneqq x_i - \bar{x}$$

This computationally cheap procedure is meant to accentuate the activity in each channel from its surrounding noise, and has been shown to improve the signal-to-noise ratio in certain applications (Ludwig, et al., 2009). Since it works with the spatial domain, it can be considered a spatial filter.

4.2.2 Results

The SBCSP method was run, with the addition of applying a common average reference (CAR). The experiment was run in two variants, one where the CAR was run *before* the data was split into multiple frequency bands (named pre-CAR here), and one where CAR was applied *after* the frequency split (named post-CAR here). The average accuracies of these variants versus the original SBCSP after a 10-fold cross validation over the BCI Competition III dataset are shown in table 4.1.

Method	Average accuracy
Baseline (SBCSP)	**86.4%**
SBCSP with pre-CAR	84.4%
SBCSP with post-CAR	83.7%

Table 4.1. Average accuracies obtained on the BCI Competition III (IVa) dataset.

4.2.3 Discussion

Adding the CAR procedure to this method yields a significantly worse accuracy. This indicates that the CAR procedure is in conflict with some other step of the SBCSP method. A reasonable suspect is the CSP algorithm. CSP and CAR are both spatial filters attempting some form of denoising, however, while CAR attempts to accentuate the individual channel activity from the other channels, CSP uses all channels to accentuate the difference in signal variance. Since the classifier is ultimately based on the signal variance, CAR can be seen as optimizing for a criterion that is irrelevant to the classification task. In doing so, it appears to harm the classification accuracy.

4.3 Covariance Matrix Estimation

4.3.1 Introduction

The CSP algorithm diagonalizes the covariance matrices estimated for the two classes of data, in order to find its projection. The inputs to the algorithm are the estimates of the covariance matrices. Since the data for each class is found in separate chunks of EEG data (one for each trial), the actual covariance estimate used for each class in SBCSP is the average of the covariance estimate for the trials in that class.

Typically, the unbiased empirical covariance matrix or the maximum likelihood estimate is used. These estimates are known to have certain problems. When the number of variables are large relative to the number of samples, the large values of the true covariance matrix are estimated to be too large, and the small values of the true covariance matrix are estimated to be too small (Schäfer, et al., 2005). These estimates are also sensitive to outliers. Therefore it is worthwhile to investigate the various approaches designed to deal with these problems.

4.3.2 Baseline: The Unbiased Empirical Covariance Matrix and the Maximum Likelihood Estimate

The unbiased empirical covariance matrix is calculated as follows:

$$\boldsymbol{\Sigma}_{\mathrm{E}} = \frac{(\mathbf{X}\text{-}\overline{\mathbf{X}})(\mathbf{X}\text{-}\overline{\mathbf{X}})^{\top}}{\mathrm{n}\text{-}1}$$

Where n is the number of samples, and $\overline{\mathbf{X}}$ is the mean vector of $\mathbf{X}$. When the random variables are normally distributed, one can derive that the following very similar estimate is the Maximum Likelihood estimate:

$$\boldsymbol{\Sigma}_{\mathrm{ML}} = \frac{(\mathbf{X}\text{-}\overline{\mathbf{X}})(\mathbf{X}\text{-}\overline{\mathbf{X}})^{\top}}{\mathrm{n}}$$

Clearly the difference between these two becomes vanishingly small as n grows large.

In the case of data that is already centered around 0 – which is approximately true for bandpass filtered EEG data – the $\overline{\mathbf{X}}$ becomes zero and can be omitted, giving the simpler form $\mathbf{X}\mathbf{X}^{\top}/\mathrm{n}$. This assumption is used in the SBCSP method. SBCSP also normalizes the covariance estimate by dividing by the sum of the diagonal elements of $\mathbf{X}\mathbf{X}^{\top}$, yielding the following matrix:

$$\boldsymbol{\Sigma}_{\mathrm{SBCSP}} = \frac{\mathbf{X}\mathbf{X}^{\top}}{\mathrm{trace}(\mathbf{X}\mathbf{X}^{\top})}$$

4.3.3 Ledoit-Wolf Shrinkage

The empirical covariance estimates above have appealing qualities such as being unbiased and being maximum likelihood, so it's somewhat surprising that they can be outperformed in certain applications. However, the maximum likelihood property is reached asymptotically, as the number of observations of a variable goes towards infinity. When the number of observations of each variable is small, they suffer from systematic errors – large values are estimated too large and small values are estimated too small.

(Stein, 1956) introduced a trick to minimize this problem, called *shrinkage*. The idea is to take a weighted average of empirical covariance estimate and some other matrix, for example the identity matrix, in order to reduce the systematic error. Here the shrunk estimate $\boldsymbol{\Sigma}_{S}$ is calculated as the weighted average of the empirical estimate $\boldsymbol{\Sigma}$ and the target matrix $\mathbf{T}$, weighted by α.

$$\boldsymbol{\Sigma}_{S} = \alpha\mathbf{T} + (1\text{-}\alpha)\boldsymbol{\Sigma}$$

The weight α determines the degree to which the empirical estimate is pushed towards the target matrix. Good choices of α can be found empirically via a computationally expensive cross-validation. However, Ledoit and Wolf in (Ledoit, et al., 2003) found an analytical closed form solution that computes an optimal value of α. The optimality guarantee of the Ledoit-Wolf theorem is that it gives the shrinkage that minimizes the distance between the true and estimated covariance matrices, where the distance is defined in terms of the Frobenius norm of the matrices. The Frobenius norm of a matrix is the square root of the sum of its squared elements:

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=0}^{n} \sum_{j=0}^{n} \mathrm{x}_{\mathrm{ij}}^2}$$

This is a quadratic measure of distance, and minimizing it gives us a solution that is optimal in a least-squares sense. A remarkable property of the Ledoit-Wolf shrinkage is that it is guaranteed to not increase this squared error no matter what the choice of target matrix $\mathbf{T}$ is. While $\mathbf{T}$ can be any matrix, good choices are the identity matrix, or other matrices that are computed from the dataset but that contain significantly less estimated parameters than the empirical covariance matrix. Various applicable choices of target matrices are presented and empirically evaluated on genomics data in (Schäfer, et al., 2005).

4.3.4 Oracle Approximating Shrinkage

The Oracle Approximating Shrinkage (Chen, et al., 2009) approach is similar to, and builds on the Ledoit-Wolf theorem. However, it introduces the additional assumption that the data is normally distributed. Under this assumption, they derive an estimator for the shrinkage parameter that is shown numerically to outperform the Ledoit-Wolf estimator in terms of Mean Squared Error, particularly when the number samples is very small compared to the number of variables. The Oracle Approximating Shrinkage is found through a closed form analytical solution, so it is of similarly low computational complexity as the Ledoit-Wolf approach.

4.3.5 Results

The various covariance estimators were used to produce the inputs to the CSP algorithm, and average accuracies were calculated through a 10-fold cross validation over the BCI Competition III (IVa) dataset (table 4.2), and over the BCI2000 dataset (table 4.3). The identity matrix was used as the target matrix for the methods with shrinkage.

Covariance estimator	Average accuracy
Normalized Covariance (SBCSP)	86.4%
Maximum Likelihood Estimate (MLE)	87.7%
MLE with Ledoit-Wolf Shrinkage	89.3%
MLE with Oracle Approximating Shrinkage	**89.7%**

Table 4.2: Average accuracies over the BCI Competition III (IVa) dataset.

Covariance estimator	Average accuracy
Normalized Covariance (SBCSP)	65.1%
Maximum Likelihood Estimate (MLE)	64.8%
MLE with Ledoit-Wolf Shrinkage	67.9%
MLE with Oracle Approximating Shrinkage	**68.0%**

Table 4.3: Average accuracies over the BCI2000 dataset.

4.3.6 Discussion

The baseline and Maximum Likelihood Estimate is outperformed by the covariance estimates with shrinkage. The MLE outperformed the baseline only in the BCI Competition dataset. Comparing the Maximum Likelihood Estimate with the SBCSP Normalized Covariance is akin to seeing the effect of assuming zero-centeredness in the data. While it is true that bandpass filtered EEG data will have a mean that is close to zero, the mean will not in general be exactly zero. How close the data is to having a mean of zero depends highly on the filter used, and it can deviate significantly from zero for certain filters. No clear winner arises between the Normalized Covariance and the MLE in this test. However both of these are outperformed by the shrunk variants.

That the shrunk variants outperform the MLE is testament to the relatively small number of samples as compared to the number of variables. The result is not entirely surprising, considering that the Ledoit-Wolf estimator is formally guaranteed to never have higher squared error than the MLE estimate. Oracle Approximating Shrinkage improves on Ledoit-Wolf in the case where data is normally distributed. For EEG data, this assumption can be justified by the central limit theorem, when we consider that the scalp activity is the result of a combination of the contributions of individual neurons in the brain. The empirical accuracies found support this assertion.

(Ledoit, et al., 2003b) open their paper with the statement: *"The central message of this paper is that nobody should be using the sample covariance matrix for the purpose of portfolio optimization."* This advice can be extended to the domain of EEG classification. The gain in accuracy should be worth the slight increase in computational cost for most applications.

4.4 Classification

4.4.1 Introduction

The final step of the SBCSP method is the component that learns to look at a vector of numbers and from it produce a prediction of what it describes, the classifier. Numerous approaches to classification exist, all of which make different trade-offs of various qualities and make different assumptions about the data. We will investigate how well these perform when applied on the features produced by the SBCSP method.

4.4.2 Baseline: Linear SVM

The Support Vector Machine finds a hyperplane to separate the classes of data. The hyperplane is chosen so as to maximize the margins between the hyperplane and the two classes. The model has been extended to a so-called soft-margin formulation, which allows it to be applied to linearly inseparable problems. This is done by adding an error term to the optimization problem, such that we simultaneously maximize the margin and minimize the classification error. This optimization problem is convex, and can be solved with Quadratic Programming techniques. However, there are different ways to formulate this optimization problem, and there are parameters that define what trade-offs are made by the model. Presented here is the formulation and the parameters that were found to give good results on the BCI Competition III dataset.

The SVM implementation used here is that of liblinear (Fan, et al., 2008). Good results were found using a L2-regularized solver to optimize a formulation of the optimization problem with an L2 squared loss function. The parameter C which weights the error term was set to 1.

4.4.3 Radial Basis Function SVM

When the dataset cannot be linearly separated, one possible approach is to apply a non-linear transformation of the dataset into a higher dimensional space, and hopefully find a better linear separating hyperplane in that space. SVMs are formulated to allow for just such a transformation, without increasing computational cost. This is done by the use of what is called the Kernel Trick. The SVM optimization problem involves evaluating the inner product of two vectors. Given a function $m(\mathbf{x})$ which maps the data to a higher dimensional space, it is known that the inner product of two mapped points can be found without explicitly evaluating the mapping, such that $m(\mathbf{x}) \cdot m(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$, where $k(\mathbf{x}, \mathbf{y})$ is the kernel function. By substituting the inner product with the kernel, we can solve the optimization problem without increasing the computational cost. A popular choice of kernel function is the Radial Basis Function: $k_{rbf}(\mathbf{x}, \mathbf{y}) \ = exp(-\gamma \ \|\mathbf{x} - \mathbf{y}\|^2)$

The SVM implementation used for this is the so-called nu-SVM formulation, which introduces a parameter $\nu \in (0, 1]$ which represents an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Good accuracies where found with $\nu = 0.2, \ \gamma = 0.05$.

4.4.4 Linear Discriminant Analysis

See the section under Linear Classification (page 20) for a description of Linear Discriminant Analysis.

To see the effects of regularization, three variants of this classifier was tested. One did not use shrinkage in the calculation of the covariance matrices, while the two other versions used Ledoit-Wolf shrinkage and Oracle Approximating Shrinkage respectively.

4.4.5 Logistic Regression

See the section under Linear Classification (page 21) for a description of Logistic Regression.

The regression parameters where penalized using the L1-norm. This regularization term was weighted with the parameter λ, which was found through cross-validation on each subject. That is, each training set (which is itself a subset of the entire dataset), was further split into a training and test set and cross-validated so as to find good values for the parameters.

4.4.6 Naïve Bayes

The probability of a class label c, given a feature vector **f,** can through Bayes theorem be stated as:

$$\mathrm{P}(\mathrm{c}|\mathbf{f}) = \frac{\mathrm{P}(\mathbf{f}|\mathrm{c})\mathrm{P}(\mathrm{c})}{\mathrm{P}(\mathbf{f})}$$

Through the naïve assumption that the features are independent, we can simplify to:

$$\mathrm{P}(\mathrm{c}|\mathbf{f}) \propto \mathrm{P}(\mathrm{c}) \prod_{i=1}^{n} \mathrm{P}\left(\mathbf{f}^{(\mathrm{i})}\middle|\mathrm{c}\right)$$

With the further assumption that the likelihood of the features are Gaussian-distributed, this gives rise to a classifier that finds the most likely class, given the feature vector.

4.4.7 Ridge Regression

While Logistic Regression is actually a classification algorithm, Ridge Regression does indeed perform regression: It produces a model with a continuous valued output. This continuous model was turned into a classifier for the purposes of this experiment. The class labels where represented as 1 and -1, and a classification was performed based on whether the regression models output was positive or negative. The Ridge Regression model finds the model $\mathrm{L}_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^{\top}\mathbf{x}$ that minimizes the error function

$$\mathrm{J}(\boldsymbol{\theta}) = \left(\sum_{i=0}^{m} \left(\mathrm{L}_{\boldsymbol{\theta}}\left(\mathbf{x}^{(\mathrm{i})}\right) - y^{(\mathrm{i})} \right)^2 \right) + \lambda\, \mathrm{L2}(\boldsymbol{\theta})$$

This makes it very much like a L2-regularized Logistic Regression, except instead of minimizing the classification error it minimizes the squared error of its continuous valued output. The regularization parameter λ was found through cross-validation on each subject, as with Logistic Regression.

4.4.8 Least Angle Regression (LARS)

The LARS algorithm (Efron, et al., 2004) creates a linear regression model by selecting a subset of the variables to use for the regression. As such it is a feature selection algorithm as well, that is optimized along with the classifier. The LARS algorithm can be seen as a natural basis for two other feature selecting regression algorithms: The L1-regularized least squares regression (called the Lasso), and an algorithm called Forward Stagewise linear regression. LARS is particularly useful when the number of variables is large relative to the number of examples. The size of the variable subset chosen by the algorithm was found through cross-validation for each subject.

4.4.9 Results

Over the BCI Competition III (IVa) dataset and over the BCI2000 dataset the classifiers reached the accuracies shown in the table 4.4 and 4.5 respectively.

Covariance Estimate	Normalized Covariance	Oracle Approximating Shrinkage
Linear SVM (SBCSP)	86.4%	89.7%
Rbf SVM	87.4%	88.8%
LDA	89.1%	90.2%
LDA (Ledoit Wolf Shrinkage)	89.2%	90.9%
LDA (OAS)	89.2%	90.8%
Logistic Regression	**89.5%**	**91.2%**
Naïve Bayes	89.1%	87.6%
Ridge Regression	89.2%	90.7%
LARS	89.4%	85.3%

Table 4.4: Average accuracies over the BCI Competition III (IVa) dataset.

Covariance Estimate	Normalized Covariance	Oracle Approximating Shrinkage
Linear SVM (SBCSP)	65.1%	68.0%
Rbf SVM	64.7%	68.0%
LDA	64.9%	67.5%
LDA (Ledoit Wolf Shrinkage)	**65.4%**	67.8%
LDA (OAS)	**65.4%**	67.8%
Logistic Regression	**65.4%**	**68.5%**
Naïve Bayes	64.9%	66.6%
Ridge Regression	65.1%	67.7%
LARS	64.2%	67.2%

Table 4.5: Average accuracies over the BCI2000 dataset.

4.4.10 Discussion

The non-linear Radial Basis Function SVM performed on par with the linear alternatives, suggesting that there is perhaps not much to gain by trying to separate these log-variance features non-linearly.

While the shrunk variants of LDA outperformed the non-shrunk variant as expected, the Oracle Approximating shrinkage did not outperform the Ledoit-Wolf shrinkage. OAS only outperforms Ledoit-Wolf on the condition that the data is normally distributed. This assumption may not hold for the log-variance features.

The four classifiers LDA, Logistic Regression, Ridge Regression and LARS all use quite similar models of the data. Of these, the last two are regressors and do not directly minimize the classification error. It is somewhat surprising that they never the less work rather well. However, the overall impression is still that one should prefer the classifiers, in particular Logistic Regression, which performs best in all four experiments.

4.5 Feature Extraction

4.5.1 Introduction

Once the CSP algorithm has produced its projection matrix, the projected EEG data typically goes through further processing where certain features are extracted, which are finally fed to a classifier. In the literature, several features have been proposed for CSP based classifiers. This section will describe some of these, and present the accuracies obtained when using them with the SBCSP method.

In the following descriptions, we will denote the raw EEG trial data by $\mathbf{X}$, the projection matrix by $\mathbf{M}$, and the resulting projected data by $\mathbf{Z}$.

$$\mathbf{Z} = \mathbf{MX}$$

The $2\mathrm{r}$ most significant projection directions of M are used for projection. So when $\mathrm{r} = 1$, we only use the first and the last row of $\mathbf{M}$ when extracting features. If $\mathrm{r} = 2$, we are picking the two first and the two last rows, and so on. To simplify the notation, we will denote the reduced matrix as $\tilde{\mathbf{Z}}$, which has size $2\mathrm{r} \times \mathrm{s}$, where s is the number of samples in the EEG recording.

Seeing as how the CSP algorithm maximizes the difference in variance, all the features presented below relate directly to variance, but have subtle differences between them.

4.5.2 Baseline: Normalized Log-Variance

Since the CSP algorithm maximizes the contrast of variance for the classes of data, a natural approach is to use the variance as a feature. Here the variance for each projection direction is normalized so they sum to one. Finally, the logarithms of these numbers are concatenated, giving a feature vector of length 2r.

$$f_{log-variance} = \left\langle \log\left(\frac{\mathrm{var}(\tilde{\mathbf{Z}}^{(1)})}{\sum_{1}^{k=2r}\mathrm{var}(\tilde{\mathbf{Z}}^{(k)})}\right), \ldots, \log\left(\frac{\mathrm{var}(\tilde{\mathbf{Z}}^{(2\mathrm{r})})}{\sum_{1}^{k=2r}\mathrm{var}(\tilde{\mathbf{Z}}^{(k)})}\right)\right\rangle$$

This is feature was proposed in (Ramoser, et al., 2000) and is the one used in the SBCSP method. In SBCSP, $r = 1$ thus this feature vector is of length 2, and is subsequently projected to one dimension using Fischer's discriminant (LDA). The resulting numbers for each subband are then concatenated to form the feature vector.

4.5.3 Normalized Log-Power

In (Ang, et al., 2009), the feature vector was formulated as follows:

$$f_{log-power} = \log\left(\frac{\mathrm{diag}\left(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^{\top}\right)}{\mathrm{trace}\left(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^{\top}\right)}\right)$$

Note that $\mathrm{diag}\left(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^{\top}\right)$ is equivalent to the sum of the squared samples for each of the $2r$ projection directions. That means it is a measure of the power of the signal. The calculation gives a feature vector of length 2r, which was then projected to one dimension using LDA.

Note also that if the data has a mean of zero, this measure is exactly proportional to the normalized log-variance measure presented above. The bandpass filtered EEG data will have a mean that is approximately, but not quite zero. The experiment will show whether this assumption affects the classification accuracy.

4.5.4 Log-Power Difference

When r=1, the normalized log-power yields two numbers that are fed to the classifier. For one class we expect the first number to be large and the other small, while we have the opposite expectation for the other class. A natural and simple feature to use then is the difference between these numbers. This feature was used, for example, by (Lei, et al., 2009).

$$f_{log-power-diff} = \log\left(\mathrm{diag}\left(\tilde{\mathbf{Z}}^{(1)}\tilde{\mathbf{Z}}^{(1)^{\top}}\right)\right) - \log\left(\mathrm{diag}\left(\tilde{\mathbf{Z}}^{(2)}\tilde{\mathbf{Z}}^{(2)^{\top}}\right)\right)$$

4.5.5 Log-Variance Difference

For completeness, we also present using the difference between the log-variances as a feature.

$$f_{log-variance-diff} = \log\left(\mathrm{var}(\tilde{\mathbf{Z}}^{(1)})\right) - \log\left(\mathrm{var}(\tilde{\mathbf{Z}}^{(2)})\right)$$

In the case where the data has zero mean, the log-variance is proportional to the log-power, so this and the previous measure are only subtly different.

4.5.6 CSP as a Channel Selector

(Wang, et al., 2005) proposed using CSP as a channel selector. We can view the spatial patterns, that is the columns of $\mathbf{M}^{-1}$, as defining a certain ranking of the channels. The method proposes picking the first and last spatial pattern, and picking the channels corresponding to the highest coefficients in the spatial pattern. Let **a** and **b** be the first and last columns of $\mathbf{M}^{-1}$. We then select the channels and compute the feature vector as follows:

$$\mathrm{c_a} = \underset{i}{\mathrm{argmax}}(\mathbf{a}^{(\mathrm{i})}) \qquad \mathrm{c_b} = \underset{i}{\mathrm{argmax}}(\mathbf{b}^{(\mathrm{i})})$$

$$f_{channel-selection} = \left\langle \mathrm{mean}\left(\left|\mathbf{X}^{(\mathrm{c_a})}\right|\right), \mathrm{mean}\left(\left|\mathbf{X}^{(\mathrm{c_b})}\right|\right), \mathrm{mean}\left(\mathbf{X}^{(\mathrm{c_a})}\right), \mathrm{mean}\left(\mathbf{X}^{(\mathrm{c_b})}\right)\right\rangle$$

The idea of using the average of the signal as a feature is in hope of detecting a physiological phenomenon known as Bereitschaftspotential or Readiness Potential (RP). The RP is a subtle build of scalp activity preceding voluntary movement, which can be observed at low frequencies.

4.5.7 Power Trend

All of the above features disregard the development of the signal over time. Perhaps some discriminative trend can be found in the time domain. This was investigated by plotting the average projected signal in a frequency band for a subject, as exemplified in figure 4.1.

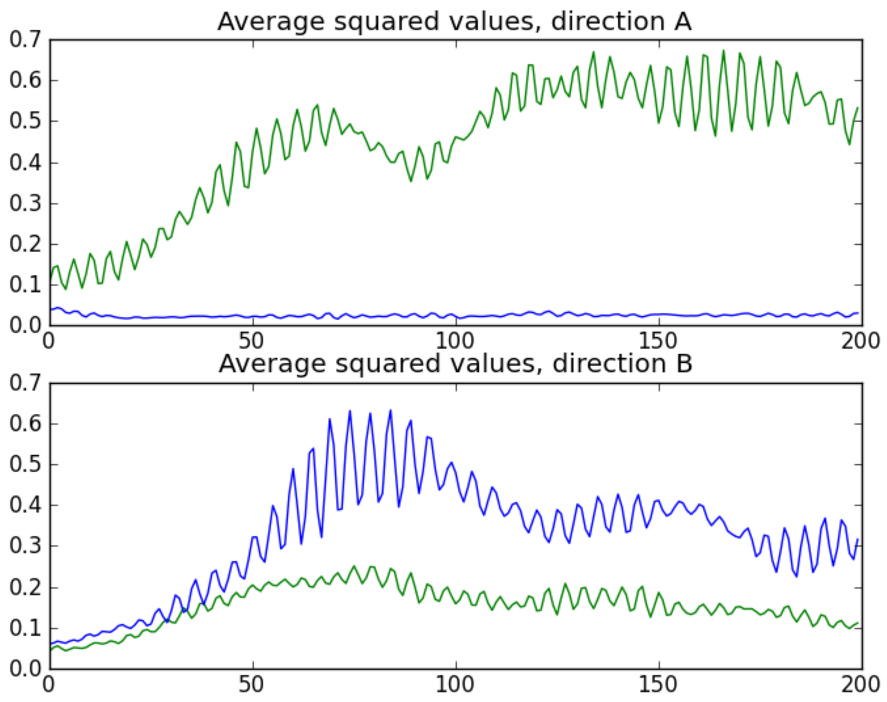


Figure 4.1: The plot shows the average of the squared values of the signal in the 10-14Hz frequency for subject "al" in the BCI Competition III dataset. The two most discriminatory projection directions from CSP are shown, with each class plotted in a separate color.

The plot shows that the CSP projection indeed does emphasize difference in signal variance, however there also appears to be a trend where the accentuated variance component increases over time.

A new feature was introduced in order to capture this development over time. By expressing the squared sample value as a function of time, we have a simple one variable Linear Regression model of the form:

$$s(t) = a + bt$$

Here a is the intercept, and b is the slope. The least squares solution to this model can be found quickly. The slope b describes if the signal increases or decreases over time. Thus, we find the slope for each of the $2r$ CSP projection directions, and use those $2r$ numbers as our feature vector.

4.5.8 Results

A 10-fold cross validation for every subject in the BCI Competition III (IVa) and the BCI2000 datasets yields the average accuracies for each of the feature sets as shown in Table 4.6 and Table 4.7. They were evaluated in combination with two types of covariance estimates for CSP: The normalized covariance estimate as in SBCSP, and the estimate given using Oracle Approximating Shrinkage.

Covariance Estimate	Normalized Covariance	Oracle Approximating Shrinkage
Normalized log-variance (SBCSP)	86.4%	**89.7%**
Log-variance difference	**87.4%**	**89.7%**
Normalized log-power	85.9%	88.4%
Log-power difference	86.9%	88.4%
CSP channel selection	68.3%	62.4%
Power trend	61.6%	61.6%

Table 4.6: Average accuracies over the BCI Competition III (IVa) dataset.

Covariance Estimate	Normalized Covariance	Oracle Approximating Shrinkage
Normalized log-variance (SBCSP)	65.1%	68.0%
Log-variance difference	65.1%	68.1%
Normalized log-power	65.4%	68.0%
Log-power difference	**66.5%**	**68.2%**
CSP channel selection	60.3%	59.6%
Power trend	52.9%	52.6%

Table 4.7: Average accuracies over the BCI2000 dataset.

Unlike the other features, the performance of the Power trend feature varied largely with the classifier used. Table 4.8 shows the accuracies with various classifiers.

Power Trend	Normalized Covariance	Oracle Approximating Shrinkage
Classifier	Average accuracy	Average accuracy
Linear SVM	61.6%	61.6%
Rbf SVM	70.9%	69.9%
LDA (Ledoit Wolf Shrinkage)	60.6%	62.0%
Logistic Regression	61.0%	62.3%
Naïve Bayes	**75.1%**	**75.6%**
Ridge Regression	59.9%	61.4%
LARS	60.9%	60.8%

Table 4.8: Average accuracies of various classifiers with the Power Trend feature over the BCI Competition III (IVa) dataset.

4.5.9 Discussion

There are several questions one could try to answer with this experiment. Is there any detrimental effect of assuming zero centered data, i.e. calculating the log-power instead of the log-variance? Is the data best described through an LDA projection of the variance or power in each CSP projection direction, or through simply calculating their arithmetic difference? How does using CSP as a channel selector compare to using the fully projected data?

The clearest result from this experiment is that using CSP as a channel selector performs poorer than the full projection. This is not surprising, given that information from all channels but one is discarded, while the full CSP projection produces a weighted combination of all channels. Much of the appeal of the channel selection approach is that its implementation is comparatively more computationally efficient.

When looking at calculating signal power versus signal variance – in other words, the effect of assuming zero centered data – we can see, surprisingly, that the power based features actually perform slightly better than the variance based features on the BCI2000 dataset. While we know analytically that the variance based features are strictly better (variance is what is contrasted by CSP), the results demonstrate that assuming data to be zero-centered will not have drastic negative effects on accuracy.

The question of whether on should use the LDA projection of normalized data, or their arithmetic difference, the results indicate a slight advantage to simply using the arithmetic difference. In none of the experiments is the difference feature outperformed by the LDA projected feature. An additional argument in favor of the arithmetic difference is that finding the LDA projection is a much more complicated calculation than doing a simple subtraction.

That an accuracy of 75.6% was achieved by Naïve Bayes using the Power Trend feature is interesting, seeing as how it is intended to detect something else than the magnitude of the power of the signal. Still, it may be the case that what is detected with this feature is in fact the signal power. For example, a positive slope of the regression line indicates an increase in power over time. But an *increase* in power will be correlated with a having a *larger* power, so this feature may be only capturing the magnitude of the power, just like the other features. To see if any information is added by knowing the *trend* of the power, we must see if any increased accuracy is gained when combining this feature with other features.

4.5.10 Combining features

Table 4.9 shows the results of combining the Power trend (PT) feature with the Normalized log-variance (NLV) and the Log-variance difference (LVD). The feature vectors were simply concatenated and projected to one dimension through Linear Discriminant Analysis. Here we only consider the results where Oracle Approximating Shrinkage was used with CSP.

	NLV	NLV & PT	LVD	LVD & PT
Linear SVM	89.7%	89.8%	89.7%	89.4%
Rbf SVM	88.8%	89.6%	90.1%	89.3%
LDA (Ledoit Wolf Shrinkage)	90.9%	91.1%	90.6%	90.8%
Logistic Regression	91.2%	**91.5%**	90.9%	91.0%
Naïve Bayes	87.6%	87.8%	85.9%	87.9%
Ridge Regression	90.7%	90.8%	90.6%	90.9%
LARS	85.3%	88.0%	88.9%	88.8%
Average	*89.2%*	*89.8%*	*89.5%*	*89.7%*

Table 4.9: Average accuracies of feature combinations over the BCI Competition III (IVa) dataset.

The results suggest that Logistic Regression with the NLV & PT features is a good choice of classifier and feature set. For the NLV feature, adding the PT feature improved accuracies for all classifiers. Adding PT to LVD yielded improved results on average. The overall improvement suggests that the better results are not flukes, but that some new discriminating information was obtained from the PT feature. Comparing table 4.5 (page 36) and table 4.16 (page 46), we can see that adding PT to NLV was beneficial also over the BCI2000 dataset, increasing accuracy from 68.5% to 68.7%.

4.6 Feature Selection

4.6.1 Introduction

Before classification, the SBCSP method performs a ranking of the features from each subband, after which the top 10 features are used in further classifications. This section will present results of using different combinations of feature eliminators and classifiers, and will also show the effect of selecting different numbers of features.

4.6.2 Baseline: Recursive Feature Elimination by Linear SVM

Whenever a classifier produces some weighing of the importance of each feature, we can do recursive feature elimination. The classifier is trained with all features, and the least important feature is discarded from the set of features. This is repeated until only one feature is left. This produces a rank on the importance of each feature, and we can select the top n features from this ranking. In the SBCSP method $n = 10$, and the weight vector of a linear SVM is used to rank the features. In this experiment, we will also show results for other choices of n. The parameters of the SVM were the same as those used previously for classification.

4.6.3 Recursive Feature Elimination by Logistic Regression

Since Logistic Regression has given the best accuracies for classification, we will try using it for feature selection. Two versions are tried here, with L1 or L2-regularization. It should be noted that when using L1-regularization, Logistic Regression tends to produce a so called sparse solution, which is to say that many of the features will be weighted to 0. L1-regularized Logistic Regression thus eliminates features as a result of the regularization of the model, which means that the least important feature will frequently have to be picked at random from the 0-weighted features under recursive feature elimination.

4.6.4 Results

Each feature elimination method was tested in combination with both Linear SVM and L1-regularized Logistic Regression as classifiers. The feature type used here is the Normalized log-variance, as in the original SBCSP. The CSP projection was found using Oracle Approximating Shrinkage for the covariance estimate. The dataset was the BCI Competition III dataset IVa. Table 4.10 and figure 4.2 shows the average accuracies obtained after 10-fold cross validation.

Feature Eliminator	Linear SVM	Linear SVM	Log. Regr. (L1-reg.)	Log. Regr. (L1-reg.)	Log. Regr. (L2-reg.)	Log. Regr. (L2-reg.)
Classifier	Linear SVM	Log. Regr. (L1-reg.)	Linear SVM	Log. Regr. (L1-reg.)	Linear SVM	Log. Regr. (L1-reg.)
n=2	90.2 %	90.0 %	90.5 %	90.5 %	90.3 %	90.5 %
n=3	90.3 %	91.0 %	**90.6 %**	91.2 %	**90.8 %**	91.3 %
n=4	89.9 %	90.9 %	89.6 %	90.9 %	90.6 %	91.1 %
n=5	90.5 %	90.9 %	90.0 %	91.1 %	90.0 %	91.1 %
n=6	**90.8 %**	**91.3 %**	89.6 %	**91.4 %**	90.1 %	**91.4 %**
n=7	89.9 %	91.2 %	89.9 %	91.2 %	90.6 %	**91.4 %**
n=8	89.0 %	**91.3 %**	89.9 %	91.1 %	90.3 %	91.3 %
n=9	89.5 %	91.1 %	90.3 %	91.2 %	89.9 %	91.3 %
n=10	89.7 %	91.1 %	89.8 %	91.2 %	89.5 %	91.2 %
n=11	89.0 %	91.1 %	89.5 %	91.3 %	89.4 %	91.2 %
n=12	88.8 %	90.9 %	89.8 %	91.3 %	89.4 %	91.1 %
n=13	88.9 %	90.9 %	89.5 %	91.1 %	89.6 %	91.1 %
n=14	88.4 %	90.9 %	89.4 %	91.1 %	89.2 %	91.1 %
n=15	88.6 %	91.0 %	89.6 %	91.1 %	89.3 %	91.1 %
n=16	88.2 %	91.1 %	89.1 %	91.1 %	89.1 %	91.1 %
n=17	88.1 %	91.1 %	89.4 %	91.1 %	88.9 %	91.1 %
n=18	88.1 %	91.1 %	89.1 %	91.1 %	88.2 %	91.1 %
n=19	88.4 %	91.1 %	88.6 %	91.1 %	88.4 %	91.1 %
n=20	88.4 %	91.1 %	88.8 %	91.1 %	88.5 %	91.1 %
n=21	88.4 %	91.1 %	89.1 %	91.1 %	88.1 %	91.1 %

Table 4.10: Average accuracies of classifiers, feature eliminators and feature counts, over the BCI Competition III dataset.

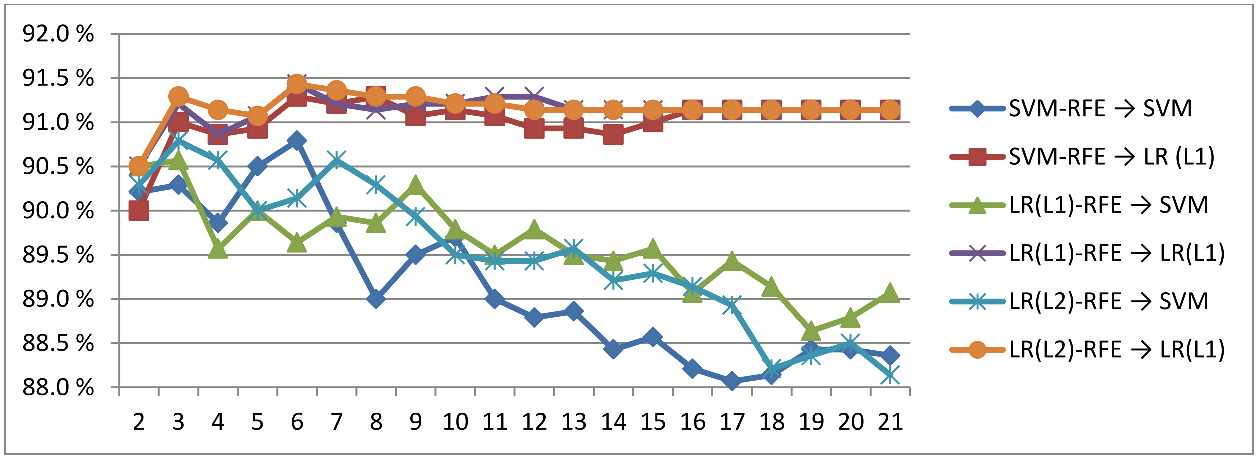


Figure 4.2: Plot of the accuracies of classifier and feature eliminator combinations over BCI Competition III (IVa) dataset. The L1-regularized Logistic Regression classifier can be seen to be more stable than the Linear SVM with regards to variations in the feature selection.

4.6.5 Discussion

A conclusion to be drawn from this experiment is that when L1-regularized Logistic Regression is used as classifier, the feature selection step is *nearly unnecessary*. For feature counts larger than 15, the L1-regularized Logistic Regression classifier achieved the exact same accuracy no matter how its features were selected. This is presumably because the regularization will assign a weight of 0 to the excessive features. While the benefit of the feature selection step was small for this classifier, peak accuracies were achieved when 6 features were selected, no matter how they were selected.

The L1 and L2-regularized Logistic Regression feature eliminators achieved about the same accuracies on average when combined with the L1-regularized Logistic Regression classifier. While the L1-regularized eliminator optimizes for the same accuracy as the classifier, it produces a sparse weighing of the features, which gives a coarser rank of the features than its L2-regularized counterpart. The results show that one can expect very little difference in accuracy between the two approaches.

This experiment solidifies the preference for L1-regularized Logistic Regression as a classifier over the Linear SVM variant. The Logistic Regression classifier gave overall better accuracies, and appears to be more stable with regards to feature count and feature quality.

4.7 Boosting

4.7.1 Introduction

The idea of boosting algorithms is to build an ensemble of classifiers that are iteratively trained on reweighted training data, or on weighted random samples of the training data. The intent is to reduce the misclassification rate in the ensemble. For example, the popular AdaBoost algorithm (Freund, et al., 1997) will increase weights on previously misclassified examples. On the other hand, BrownBoost (Freund, 2001) takes the opposite approach of effectively “giving up” on the repeatedly misclassified examples. The classifiers are finally combined through majority voting. Emphasizing misclassified examples, as done by AdaBoost, can be shown to be similar to the margin maximization that is done by an SVM. An introduction to boosting can be found in (Freund, et al., 1999).

4.7.2 Algorithm

The boosting algorithm used here is similar, but not equivalent, to that of AdaBoost, in that it emphasizes misclassified examples. A weight vector is maintained that weighs each training example, and another weight vector weighs each classifier. In each of the maximum 100 iterations, a classifier is trained on a subset of the training data and tested on the whole set. The subset is found through a weighted random sample, with replacement, of the whole training set. The sample weights for misclassified examples are increased by 0.1, while sample weights for correctly classified examples are decreased by 0.1. Finally the accuracy of the classifier is measured. If the accuracy is less than 0.5, the process is stopped. Otherwise, the classifier is added to the ensemble, weighted by its accuracy. The final classification of the ensemble is then the weighted sum of all their predictions.

4.7.3 Results

Boosting performs iterative random sampling of the dataset, and this randomization may cause the measured accuracy to vary from run to run. Thus, to get a clearer picture of the true expected accuracy of this procedure, it was run 200 times. Tables 4.11 – 14 show the frequencies of the various accuracies obtained by these runs. Figures 4.3 – 4.6 visualize this with histograms. The boosting procedure was tested with four different combinations of features, classifiers and datasets.

Test 1: Baseline (SBCSP)

CSP Covariance: Normalized, no shrinkage. *Classifier*: Linear SVM. *Feature Elimination*: Linear SVM (10-features). *Features*: Normalized Log-Variance. *Dataset:* BCI Competition III (IVa).

Accuracy	Frequency
86.0 %	3
86.1 %	1
86.2 %	3
86.3 %	5
86.4 %	16
86.5 %	7
86.6 %	15
86.7 %	32
86.8 %	18
86.9 %	30
87.0 %	16
87.1 %	16
87.2 %	23
87.3 %	6
87.4 %	6
87.5 %	2
87.6 %	1
Mean boosted accuracy	86.9%
Unboosted accuracy	86.4%

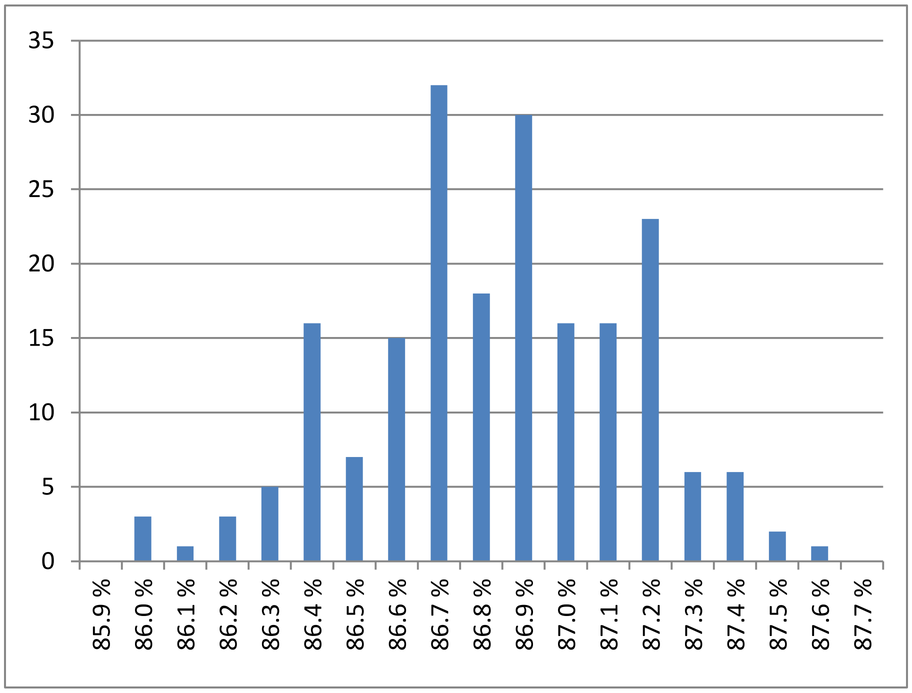


Table 4.11: Frequency of accuracies after boosting, test 1.
Figure 4.3: Histogram of accuracies after boosting, test 1.

Test 2: Logistic Regression with Normalized Log-Variance features

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Logistic Regression (L1-regularized). *Feature Elimination*: Logistic Regression (L1-regularized, 6 features). *Features*: Normalized Log-Variance. *Dataset:* BCI Competition III (IVa)

Accuracy	Frequency
90.8 %	0
90.9 %	11
91.0 %	27
91.1 %	34
91.2 %	62
91.3 %	24
91.4 %	38
91.5 %	4
91.6 %	1
91.7 %	0
Mean boosted accuracy	91.3%
Unboosted accuracy	91.4%

Table 4.12: Frequency of accuracies after boosting, test 2.

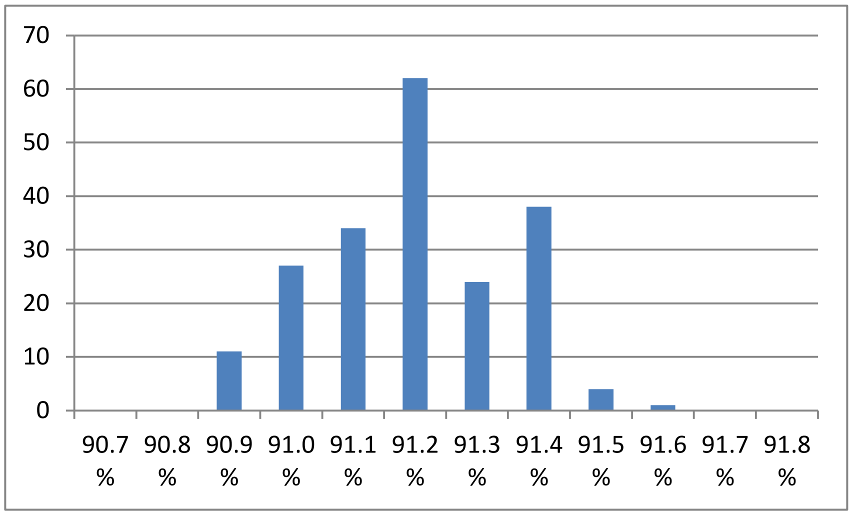


Figure 4.4: Histogram of accuracies after boosting, test 2.

Test 3: Logistic Regression with Normalized Log-Variance and Power Trend features

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Logistic Regression (L1-regularized). *Feature Elimination*: Logistic Regression (L1-regularized, 6 features). *Features*: Normalized Log-Variance + Power Trend. *Dataset:* BCI Competition III (IVa)

Accuracy	Frequency
91.1 %	0
91.2 %	4
91.3 %	9
91.4 %	34
91.5 %	20
91.6 %	35
91.7 %	58
91.8 %	24
91.9 %	15
92.0 %	0
92.1 %	1
92.2 %	1
Mean boosted accuracy	91.7%
Unboosted accuracy	91.1%

Table 4.13: Frequency of accuracies after boosting, test 3.

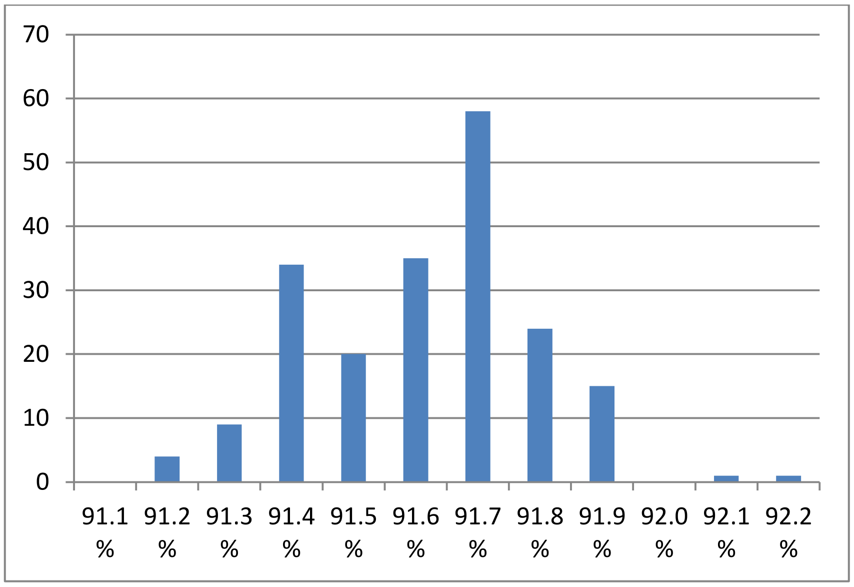


Figure 4.5: Histogram of accuracies after boosting, test 3.

Test 4: Logistic Regression with Normalized Log-Variance and Power Trend features

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Logistic Regression (L1-regularized). *Feature Elimination*: Logistic Regression (L1-regularized, 6 features). *Features*: Normalized Log-Variance + Power Trend. *Dataset:* BCI2000

Accuracy	Frequency
68.3 %	1
68.4 %	0
68.5 %	3
68.6 %	8
68.7 %	23
68.8 %	21
68.9 %	45
69.0 %	42
69.1 %	25
69.2 %	20
69.3 %	10
69.4 %	2
Mean boosted accuracy	69.0%
Unboosted accuracy	68.7%

Table 4.14: Frequency of accuracies after boosting, test 4.

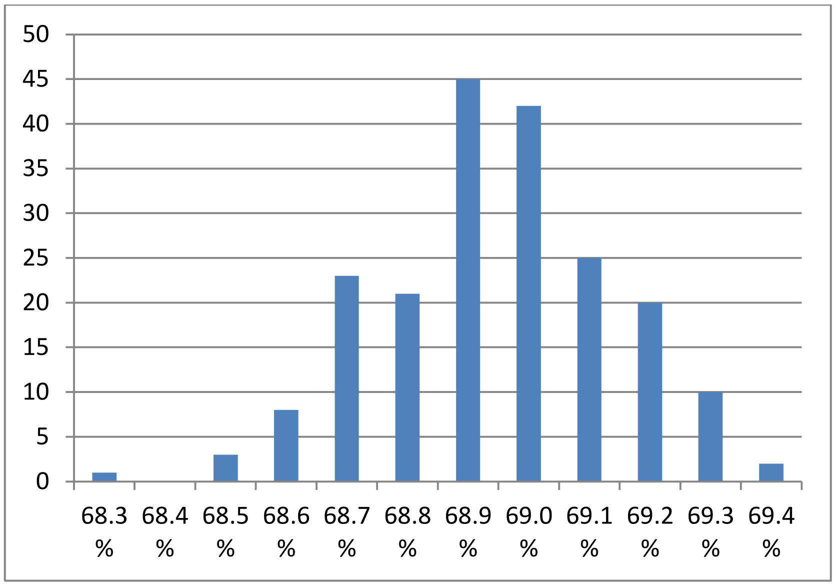


Figure 4.6: Histogram of accuracies after boosting, test 4.

4.7.4 Discussion

The main question to be answered by this experiment is whether or not boosting is beneficial when used against each of the three feature/classifier combinations tested. The clearest conclusion is to be drawn from test 3. In test 3, each of the 200 boosted classifiers were more accurate than the unboosted classifier, and we can confidently conclude that boosting is beneficial for this feature/classifier combination. A benefit is also seen in the other tests, except in test 2. Still, no test showed any significant detrimental effect by using boosting, and the general impression is that boosting will often be beneficial, and is certainly worth trying in similar EEG classification setups.

4.8 Bandpass Filter Optimization

4.8.1 Introduction

The first step of the SBCSP algorithm is to split the signal into multiple frequency ranges. There are many ways of implementing such a filter. (Novi, et al., 2007) originally performed the filtering with a set of Gabor filters. In their reproduction, (Ang, et al., 2008) used a set of Chebyshev Type II filters. For the experiments in this thesis, a set of Finite Impulse Response (FIR) filters have been used. The purpose of this section is not to compare these methods, but rather to compare the different parameters with which one can design such a filter. Since the goal of extracting, for example, the 6-10Hz range of a signal can be achieved through several different FIR-filters, we wish to investigate the effect such changes have on the classification accuracy. Further, we wish to determine whether or not 'good' settings for these parameters can be expected to hold for new datasets or if the apparently better parameters performed better due to random fluctuations in accuracy.

4.8.2 Parameterization

A FIR-filter is determined by its filter coefficients, and these coefficients are typically found through some filter design procedure. The frequency range that is not attenuated is called the *passband*, the area that is attenuated is called the *stopband* and the gap between these two is called the *transition band*. For this experiment, we will characterize the filters we design by two parameters. First, the parameter w denotes the desired width of the passband, in hertz. The parameter d denotes the desired width of the transition band. Figure 4.7 illustrates what is meant by these parameters.

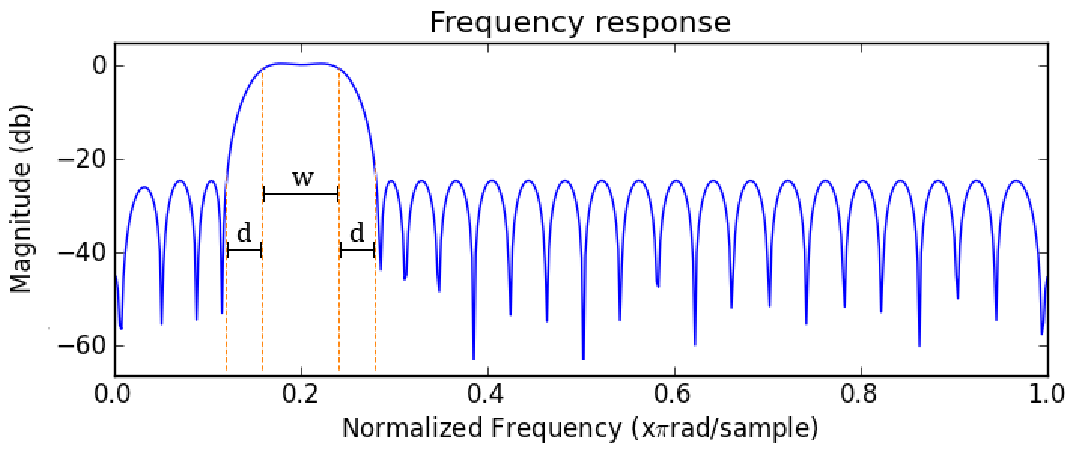


Figure 4.7: Illustration of the terminology used in these experiments. We will denote the desired width of the passband by w, and denote the size of the transition between the passband and stopband by d.

The parameters w and d will be used to design filters using two methods, the Parks-McClellan filter design algorithm and the window method using a Hamming window. See e.g. (Oppenheim, et al., 2009) for a description of these. Figure 4.8 shows examples such filters. All filters were of order 52. The implementations in the Scipy library (Jones, et al., 2001) were used.

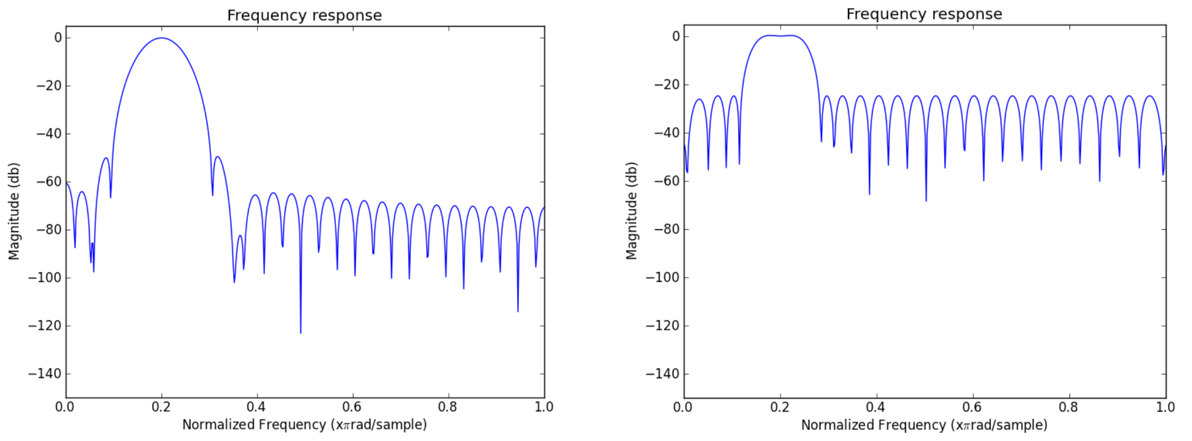


Figure 4.8: The frequency response of two filters. Left: A filter designed by the window method, w = 4, d = 0. Right: Filter designed by the Parks-McClellan algorithm, w = 4, d = 2. This is the filter used in the previous experiments.

4.8.3 Results

Presented here in figures 4.9-4.13 are the average accuracies after cross validation, obtained using different filter parameters. In order to aid visual interpretation of the data, the results are presented as bubble diagrams, where the difference in circle diameter is proportional to the relative difference in performance within each of the two filter types.

Test 1: SBCSP (Baseline)

CSP Covariance: Normalized, no shrinkage. *Classifier*: Linear SVM. *Feature Elimination*: Linear SVM (10-features). *Features*: Normalized Log-Variance. *Dataset:* BCI Competition III (IVa)

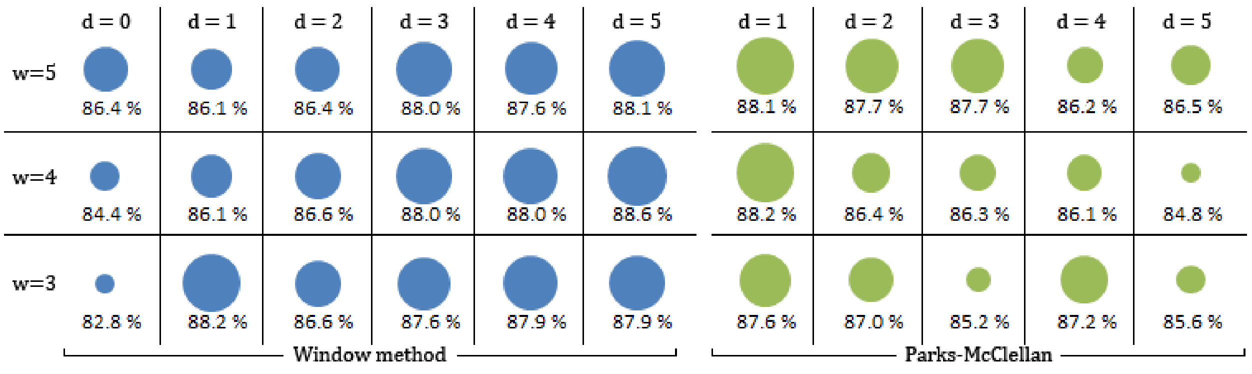


Figure 4.9: Accuracies obtained with various filter parameters.

Test 2: With shrinkage

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Linear SVM. *Feature Elimination*: Linear SVM (10-features). *Features*: Normalized Log-Variance. *Dataset:* BCI Competition III (IVa)

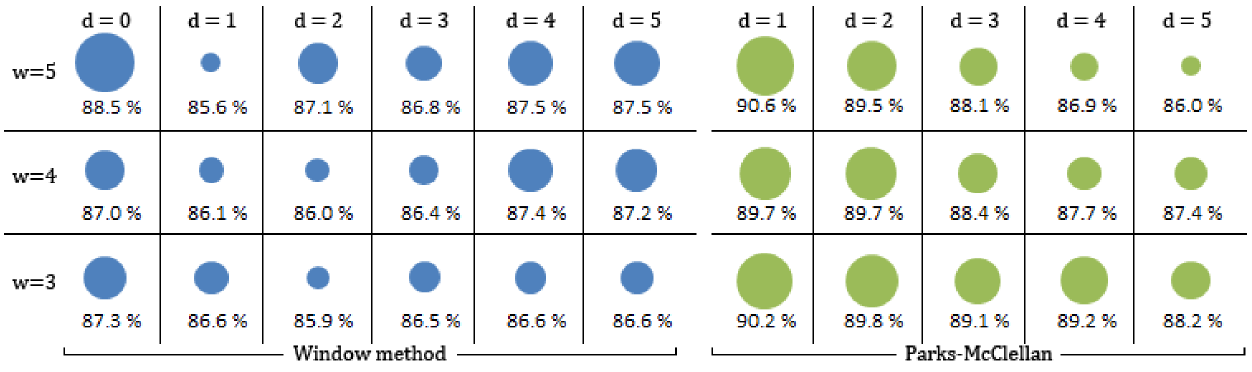


Figure 4.10: Accuracies obtained with various filter parameters.

Test 3: With shrinkage, Logistic Regression

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Logistic Regression (L1-regularized).
Feature Elimination: Linear SVM (10-features). *Features*: Normalized Log-Variance. *Dataset:* BCI Competition III (IVa)

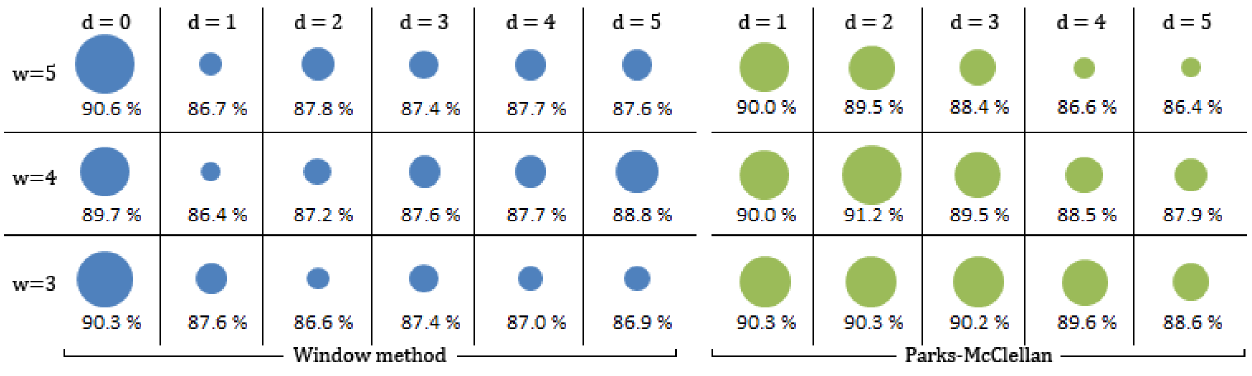


Figure 4.11: Accuracies obtained with various filter parameters.

Test 4: With shrinkage, Logistic Regression (BCI2000)

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Logistic Regression (L1-regularized). *Feature Elimination*: Linear SVM (10-features). *Features*: Normalized Log-Variance. *Dataset:* BCI2000

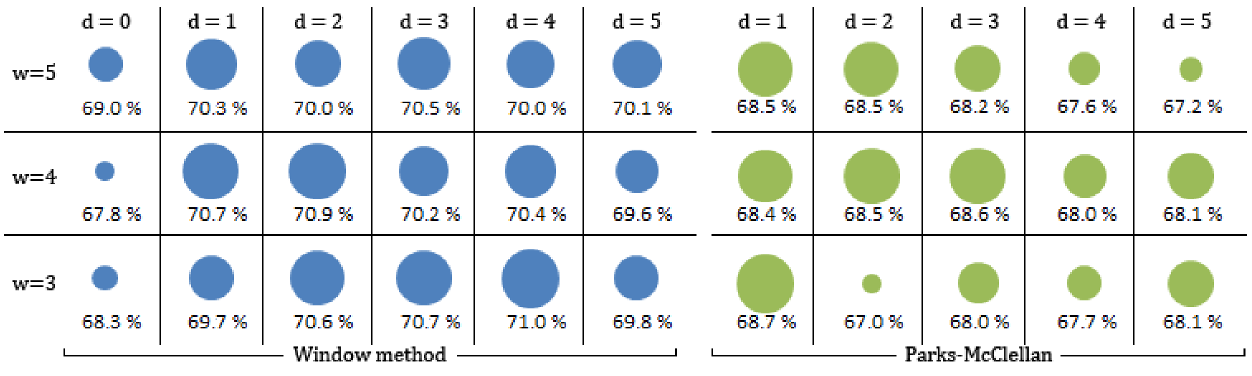


Figure 4.12: Accuracies obtained with various filter parameters.

Test 5: With shrinkage, Logistic Regression, Normalized Log-Variance and Power Trend features

CSP Covariance: Oracle Approximating Shrinkage. *Classifier*: Logistic Regression (L1-regularized). *Feature Elimination*: Linear SVM (10-features). *Features*: Normalized Log-Variance + Power Trend. *Dataset:* BCI Competition III (IVa)

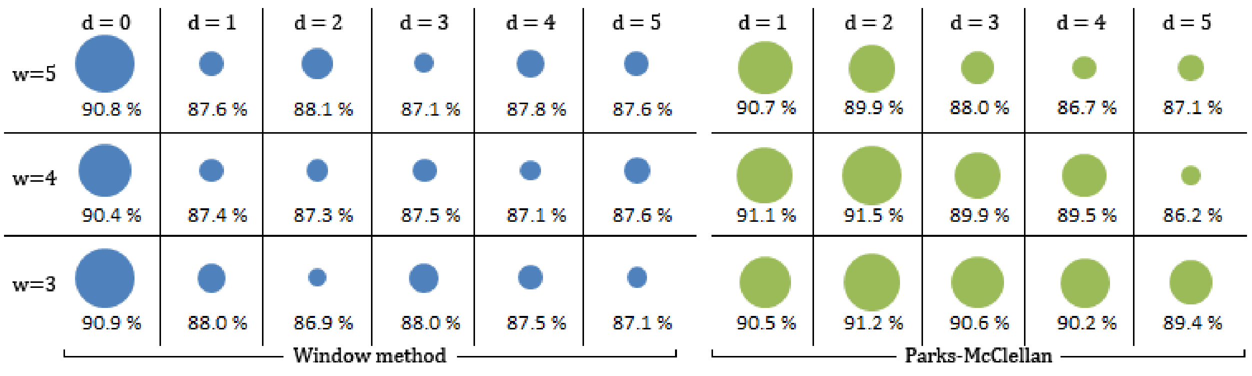


Figure 4.13: Accuracies obtained with various filter parameters.

4.8.4 Discussion

To the question of which parameters one should choose for filter design, this experiment does not give definitive answer. It is hard to discern any “best” filter setting by this data. For example, the best Window method filter in test 5 was w = 4, d= 0. This filter was the *worst* Window method filter in test 1. Similarly contradictory results can be seen between test 3 and test 4, which used exactly equal methods on different datasets. This builds the impression that most fluctuations in accuracy per parameter setting is random. However, a set of filters that performed well in all tests were the sharpest Parks-McClellan filters (i.e. d = 1). These appear to be robust parameter choices.

What these result indicate is that while the choice of filter parameters affects classification accuracy, the increase or decrease in accuracy is not generally due to the filter being better or worse, but rather due to the variance of the classifier. Thus changing filter parameters allow us to produce numerous variations of the same dataset, causing fluctuations in the accuracy that may well be best attributed to chance.

5 Conclusion

The contribution of this work has been to implement a state of the art classification system, to develop several hypotheses as to how to improve the system, and then to test these empirically.

Common Average Referencing: Using CAR reduced the classification accuracy. The procedure is a spatial filter that attempts to isolate the contribution of each channel from its environment (the other channels). While this is a useful goal in many contexts, it appears to be entirely irrelevant to maximizing differences in variance, which is the basis of the classifier. The results indicate that it is not only irrelevant, but in fact harmful to the classification accuracy.

Covariance Estimation: Accuracy was improved significantly by shrinking the covariance estimates used by the CSP algorithm. Ledoit-Wolf (LW) shrinkage is proven to never do worse than the standard covariance estimate, so the increase in accuracy from using it was unsurprising. However, Oracle Approximating Shrinkage (OAS) does not have the same guarantees. Still, we argue that OAS is preferable to LW in this context. The argument for this is two-fold. 1) The underlying assumption of OAS that the data is normally distributed is reasonable for EEG data. 2) Empirically, the accuracies gained using OAS were slightly better than with LW.

Classification: Several classifiers were compared. The results show that linear classifiers give good results, and that of these L1-regularized Logistic Regression performed the best in all tests.

Features: The experiments showed that the simplifying assumption of zero-centered data in the calculation of the features could be detrimental to overall accuracy. The Power Trend feature was introduced and shown to have some additional discriminatory information not contained in the variance based features. Overall, the very best results were gained by combining the Normalized Log-Variance and the Power Trend features.

Feature Selection: When L1-regularized Logistic Regression was used as a classifier, the feature selection step proved to become much less important, because the regularization procedure in effect performs the feature selection itself. Still, of the feature selectors, the Logistic Regression based one outperformed the SVM based selection. The highest accuracy was obtained when selecting 6 features.

Boosting: The results with regards to boosting varied largely with the classifier and the features it was combined with. However, the picture arising from the tests is that boosting probably will not hurt the accuracy, and that it in some cases will help a lot. Therefore boosting seems advisable.

Bandpass Filter Optimization: Searching through the space of filter parameters may result in increased classification accuracy, but such increases may very well be attributable to chance and cannot generally be expected to generalize to new datasets.

Overall the improvements have taken the classification accuracy on the BCI competition III dataset from 86.4%, to an accuracy after boosting in the range of 91.2-92.2%, with 91.7% being the average. On the BCI2000 dataset, the methods accuracy was increased from 65.1% to 69.0%.

5.1 Future Work

Certain other EEG-classification systems are based on extracting a single frequency band by optimizing a frequency filter along with the spatial filter. Some of the improvements found in this work would likely be beneficial in these methods as well, although this has not been confirmed empirically. Boosting, and the usage of the power trend feature are particularly promising candidates for such an experiment. Additionally, the use of covariance matrix shrinkage is theoretically guaranteed to be beneficial, although it is unclear if shrinkage can be directly applied during the optimization of the spatio-spectral filter.

The result that the Power Trend feature on its own is significantly better discriminated by a Naïve Bayes classifier suggest that perhaps a better accuracy could possibly be achieved by some combination of Logistic Regression and Naïve Bayes in an ensemble, trained on different features with separate runs of feature selection. It should also be noted that the discriminability found in the signals development over time after CSP projection is somewhat “by accident”, as the CSP algorithm ignores the time domain. If a spatial filter could be directly optimized for such time domain discriminability, it could possibly improve classification accuracy significantly when combined with traditional CSP.

References

Ang Kai Keng [et al.] Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface [Konferansebidrag] // International Joint Conference on Neural Networks. - San Jose, California, USA : [s.n.], 2008. - ss. 2390-2397.

Ang Kai Keng [et al.] Robust Filter Bank Common Spatial Pattern (RFBCSP) in motor-imagery-based Brain-Computer Interface [Konferansebidrag] // 31st Annual International Conference of the IEEE EMBS. - Minneapolis, Minnesota, USA : [s.n.], 2009. - ss. 578-571.

Birbaumer N. [et al.] A spelling device for the paralysed [Artikkel] // Nature. - [s.l.] : Nature Publishing Group, 1999. - 398. - ss. 297-298.

Blankertz Benjamin [et al.] Single-trial analysis and classification of ERP components--A tutorial [Artikkel] // NeuroImage (preprint). - Berlin : [s.n.], 2010.

Blankertz Benjamin BCI Competition III [Internett]. - Fraunhofer FIRST (IDA), 2005. - http://www.bbci.de/competition/iii/.

Chen Yilun, Wiesel Ami og III Alfred O Hero Shrinkage Estimation of High Dimensional Covariance Matrices [Konferansebidrag] // ICASSP '09 Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. - Washington, DC : IEEE Computer Society, 2009.

Dornhege Guido [et al.] Optimizing spatio-temporal filters for improving Brain-Computer Interfacing [Artikkel] // Advances in Neural Inf. Proc. Systems. - 2006.

Efron Bradley [et al.] Least Angle Regression [Artikkel] // Annals of Statistics. - 2004.

Fan Rong-En [et al.] LIBLINEAR: A Library for Large Linear Classification [Artikkel] // Journal of Machine Learning Research. - 2008. - 9. - Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

Fisher Ronald The Use of Multiple Measurements in Taxonomic Problems [Artikkel] // Annals of Eugenics. - 1936.

Freund Yoav An Adaptive Version of the Boost by Majority Algorithm [Artikkel] // Machine Learning. - 2001.

Freund Yoav og Schapire Robert E. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting [Artikkel] // Journal of Computer and System Sciences. - 1997.

Freund Yoav og Schapire Robert E. A Short Introduction to Boosting [Artikkel] // Journal of Japanese Society for Artificial Intelligence. - 1999.

Fukunaga Keinosuke og Koontz Warren L. G. Application of the Karhunen-Loeve Expansion to Feature Selection and Ordering [Artikkel] // IEEE Transactions on Computers. - 1970. - 19. - 4.

Hild Kenneth E., Kurimo Mikko og Calhoun Vince D. The Sixth Annual MLSP Competition [Rapport]. - 2010.

Jarŝová Eva Estimation With The Linear Mixed Effect Model [Artikkel] // Journal of Applied Mathematics. - 2010. - 3. - 3.

Jones Eric, Oliphant Travis og others Pearu Peterson and SciPy: Open Source Scientific Tools for Python. - 2001. - http://www.scipy.org/.

Kang Hyohyeong, Nam Yunjun og Choi Seungjin Composite Common Spatial Pattern for Subject-to-Subject Transfer [Artikkel] // Signal Processing Letters, IEEE. - 2009. - 16.

Koles Z. J. The quantitative extraction and topographic mapping of the abnormal components in the clinical EEG [Artikkel] // Electroencephalography and clinical Neurophysiology. - [s.l.] : Elsevier Scientific Publishers Ireland, 1991. - 79.

Krauledat Matthias [et al.] Reducing calibration time for brain-computer interfaces: A clustering approach [Artikkel] // Advances in Neural Information Processing Systems. - 2007. - 19.

Ledoit Olivier og Wolf Michael Honey, I Shrunk the Sample Covariance Matrix [Artikkel]. - 2003b.

Ledoit Olivier og Wolf Michael Improved estimation of the covariance matrix of stock returns with an application to portfolio selection [Artikkel] // Journal of Empirical Finance. - [s.l.] : Elsevier, 2003.

Lei Xu [et al.] Common Spatial Pattern Ensemble Classifier and Its Application in Brain-Computer Interface [Artikkel] // JOURNAL OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA. - 2009. - 7. - 1.

Lemm S. [et al.] Spatio-spectral filters for improving the classification of single trial EEG [Artikkel] // IEEE Transactions on Biomedical Engineering. - 2005.

Lemm Steven [et al.] Introduction to Machine Learning for Brain Imaging [Artikkel] // NeuroImage. - 2011. - 56.

Lotte F [et al.] A Review of Classication Algorithms for EEG-based Brain-Computer Interfaces [Rapport]. - 2010.

Ludwig Kip A. [et al.] Using a Common Average Reference to Improve Cortical Neuron Recordings From Microelectrode Arrays [Artikkel] // Journal of Neurophysiology. - 2009. - Vol. 101.

McClellan J., Parks T. og Rabiner L. A computer program for designing optimum FIR linear phase digital filters [Artikkel] // IEEE Transactions on Audio and Electroacoustics. - 1973. - 21.

Mitchell Tom Michael Machine Learning [Bok]. - [s.l.] : MIT Press & McGraw-Hill Book Co, 1997.

Müller Klaus-Robert [et al.] Machine learning for real-time single-trial EEG-analysis: From brain–computer interfacing to mental state monitoring [Artikkel] // Journal of Neuroscience Methods. - 2008. - 1 : Vol. 167.

Novi Quadrianto [et al.] Sub-band Common Spatial Pattern (SBCSP) for Brain-Computer Interface [Konferansebidrag] // IEEE EMBS Conference on Neural Engineering. - Kohala Coast, Hawaii, USA : [s.n.], 2007. - ss. 204-207.

Oppenheim Alan V. og Schafer Ronald W. Discrete-Time Signal Processing [Bok]. - [s.l.] : Prentice Hall Press, 2009.

Pearson Karl On Lines and Planes of Closest Fit to Systems of Points is Space [Artikkel] // Philosophical Magazine Series. - 1901.

Pfurtscheller G. og Silva F.H. Lopes da Event-related EEG/MEG synchronization and desynchronization: basic principles [Artikkel] // Clinical Neurophysiology. - 1999.

Ramoser Herbert, Müller-Gerking Johannes og Pfurtscheller Gert Optimal Spatial Filtering of Single Trial EEG During // IEEE Trans. Rehab. Eng.. - 2000. - Vol. 8. - ss. 441-446.

Rebsamen Brice [et al.] Controlling a Wheelchair Indoors Using Thought [Artikkel] // IEEE Intelligent Systems. - 2007. - 2 : Vol. 22. - ss. 18-24.

Schalk G. [et al.] BCI2000: A General-Purpose Brain-Computer Interface (BCI) System [Artikkel] // IEEE Transactions on Biomedical Engineering. - 2004.

Schäfer Juliane og Strimmer Korbinian A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics [Artikkel] // Statistical Applications in Genetics and Molecular Biology. - [s.l.] : The Berkeley Electronic Press, 2005. - 1 : Vol. 4.

Stein Charles Inadmissibility of the usual estimator for the mean of a multivariate normal distribution [Konferansebidrag] // Proceedings of the third Berkeley symposium on Mathematical and Statistical Probability. - [s.l.] : Stanford University, 1956.

Verhulst P. F. Notice sur la loi que la populations suit dans son accroissement [Artikkel] // Correspondence Mathematique et Physique. - 1838. - 10.

Wang Yijun, Gao Shangkai og Gao Xiaorong Common Spatial Pattern Method for Channel Selelction in Motor Imagery Based Brain-computer Interface [Konferansebidrag] // 27th Annual International Conference of the Engineering in Medicine and Biology Society. - Shanghai : [s.n.], 2006.