Classification of EEG data using machine learning techniques

Martin Heyden

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6019
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

Automatic interpretation of reading from the brain could allow for many interesting applications including movement of prosthetic limbs and more seamless man-machine interaction.

This work studied classification of EEG signals used in a study of memory. The goal was to evaluate the performance of the state of the art algorithms. A secondary goal was to try to improve upon the result of a method that was used in a study similar to the one used in this work.

For the experiment, the signals were transformed into the frequency domain and their magnitudes were used as features. A subset of these features was then selected and fed into a support vector machine classifier. The first part of this work tried to improve the selection of features that was used to discriminate between different memory categories. The second part investigated the uses of time series as features instead of time points.

Two feature selection methods, genetic algorithm and correlation-based, were implemented and tested. Both of them performed worse than the baseline ANOVA method.

The time series classifier also performed worse than the standard classifier. However, experiments showed that there was information to gain by using the time series, motivating more advanced methods to be explored.

Both the results achieved by this thesis and in other work are above chance. However, high accuracies can only be achieved at the cost of long delays and few output alternatives. This limits the information that can be extracted from the EEG sensor and its usability.

Acknowledgements

Thanks to Inês Bramão and Mikael Johansson at the Department of Psychology Lund University, for answering my questions and making this project a possibility.

Thanks to everyone involved in the project at LTH: Bo Bernhardsson, Anders Robertsson and Michelle Chong for their guidance and helpful feedback.

Thanks to Johan Eker at Ericsson for many good discussions.

Thanks to the Department of Automatic Control for giving me an inspiring work environment.

Contents

1 Introduction

1. Motivation

The human body is very capable of interacting with its surrounding. It can share its thoughts by speaking and move around via muscles. In some cases this is not possible due to disabilities or injuries. In other cases it would be beneficial if the brain could interact directly with its surrounding without first having to go through the rest of the body. For example activating an emergency button at the moment the brain wants to could be faster than the person physically having to press it.

This might sound like science fiction but the technology is not that far away. One way for the neurons in the brain to communicate is through electrical activity. Those electrical activities will change depending of what the brain is trying to accomplish. If that activity can be measured and interpreted, the intentions of a person could be read without relying on the rest of his body.

One such method for measuring brain activity is electroencephalography (EEG). It measures the mean membrane potential of populations of neurons through a series of electrodes, usually attached to the scalp, see Figure 1.1. These devices have been around for a long time and have been used to assist in the diagnosis of epilepsy and sleep disorders by looking at the electroencephalographic activity. This has already been aided by automatic classification of the data where it has been found that certain EEG activities occur before the onset of seizures [Berg et al., 2010]. It can also be used to monitor the mental state of pilots [*The pilot brain*], pilot quadcopters [LaFleur et al., 2013], control of humanoid robots [Chae et al., 2012] or controlling exoskeletons for disabled persons [Schaap, 2016].

A system that reads signals from the brain and then interprets the signals to make decisions or execute commands is called a Brain Computer Interface (BCI) and this is an emerging research field.

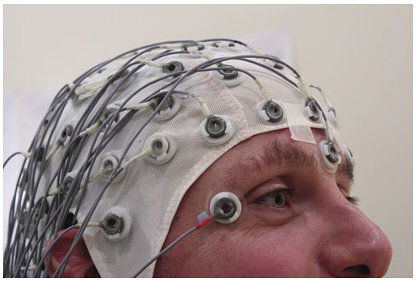


Figure 1.1 A cap used for EEG measurements. The measurements are read from a series of electrodes placed evenly over the head. Taken from `http://www.flickr.com/photos/tim_uk/8135755109/`.

2. Project Partners

This project has been a cooperation between three entities: The Department of Psychology at Lund University, the Department of Automatic Control at Lund University and Ericsson Research. Different goals were envisioned by each entitiy, but it was believed that they could be combined into one project. These goals are described below.

The psychology department is conducting research within memory research and specifically, in the mechanism of memory creation and retrieval.

When someone is trying to remember something, the memory is said to be *encoded* by the brain. When that memory is later being remembered, it is said to be *recollected*. An interesting question is if the processes that happens during the encoding of a memory is later replayed during recollection. One way of testing this is to see if patterns present during encoding is also present during recollection [Jafarpour et al., 2014].

There is currently no unifying theory for how this process work [Widmaier et al., 2014].

The Department of Automatic Control is interested in using the EEG as a sensor in robotics applications. It could also be used as a sensor in vehicles as a safety system to monitor the state and intentions of the driver.

Ericsson is interested in finding new sensors and techniques that can be utilized with the help of the new 5G technology. The EEG sensor could then be wireless and the calculations done in the cloud. It could also be an interesting part of the internet of things.

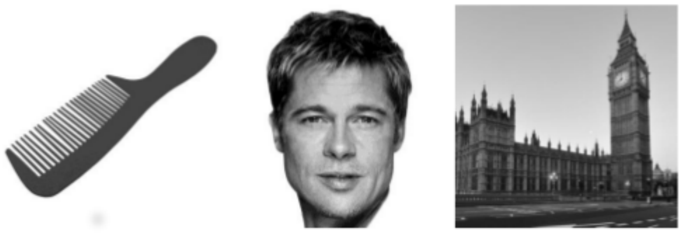


Figure 1.2 Example of images used in the memory study used in this project. The pictures come from three different categories: objects, faces and landmarks. The participants are asked to try to memorize word/picture pairs and are later asked to recall the picture when presented with the word. [Image source: Inês Bramão, Department of Psychology, Lund University.]

3. The Data

The data for this project was supplied by the Department of Psychology at Lund University and was designed to study if patterns during encoding emerged during recollection.

The study consisted of the participants trying to remember word-picture pairs. The picture was from three categories: faces, landmarks and, objects. An example of the pictures used in the study is depicted in Figure 1.2.

EEG measurements were recorded during the entire experiment. However, only the recordings from memorization were used in this work. More details about the data is given in Chapter 5.

4. Challenges

It is challenging to automatically classify EEG signals due to the problems with the signals described below. Advances in computing power, signal processing, machine learning, and EEG reading devices have however increased the capabilities of automatic interpretations of the results.

The Signals

Electroencephalography (EEG) uses electrodes placed over the skull to measure electric activity in the brain.

A conductive gel is usually applied to the electrodes to improve reading quality. For an every day BCI system this is not feasible. And even then the EEG measurements are on the μV scale and have low signal-to-noise ratio. The signals are further corrupted by artifacts, electrical signals originating from, for example, eye and muscle

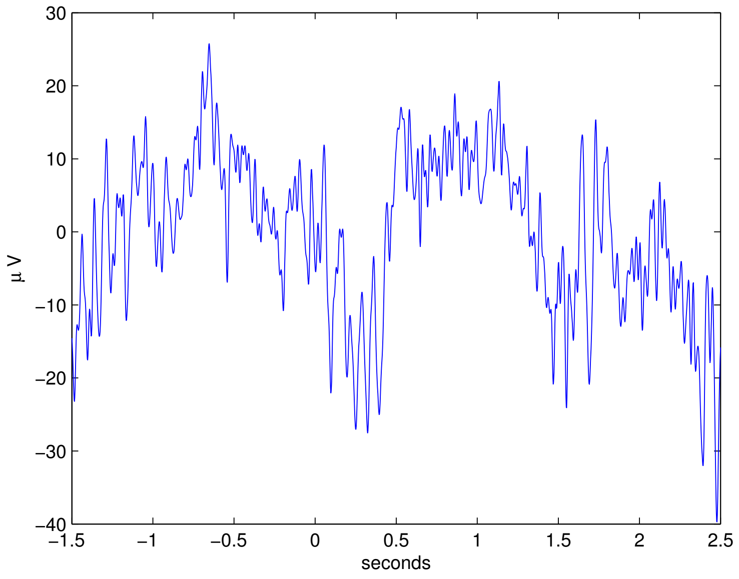


Figure 1.3 Plot of EEG readings from one of the channels during the memory experiment. Notice that it is on the magnitude of micro Volts and that at least two different frequency components can be seen.

movement. These artifacts might be much stronger than the electrical activity that is studied. The signals also often have multiple different frequency components even in the absence of artifacts. Another problem is the limited spatial resolution.

The strength of EEG lies in its high temporal resolution and that it is cheaper and more portable than many other brain imaging techniques.

An example of the readings from channel 24 can be seen in Figure 1.3. It can be seen that there are at least two different frequencies present. Notice also the low amplitude of the magnitude of ten micro Volts.

Classification and feature selection

This work will focus on the problem of classification. The goal of classification is to use measurements to decide what the source of the measurements is. Choosing what measurements to use is a large part of the problem. The chosen measurements are called features. The different sources are called classes.

When trying to classify between a mouse and an elephant there are many possible measurements. One could measure the color of the animals presented but that would

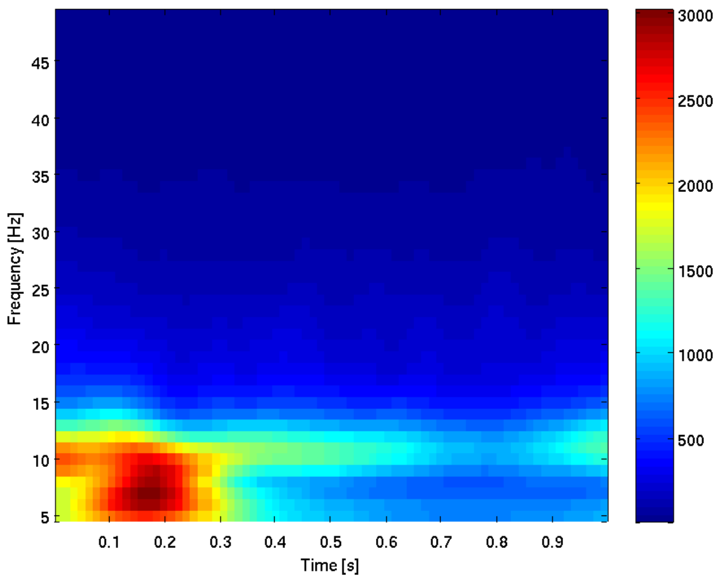


Figure 1.4 Time-frequency transformation of EEG readings. The transformation is done independently for each channels and the mean of the results is displayed. The stimuli is presented at time zero.

probably not be a very good feature since they both are gray. Their weight would however be sufficient to discriminate between the two classes of animals.

For harder problems multiple features must be used to do efficient classification. In this thesis, frequency transformation of the EEG signals was used to classify what category the user is trying to remember.

Frequency transformation is a method for finding out the magnitude of the different frequency parts of a signal, much like the Fourier Transform, but with the magnitudes varying over time. A picture of the mean of all time-frequency transforms from an EEG recording can be seen in Figure 1.4

The time-frequency transformation gives many features and the amount needs to be reduced somehow to help the training of the classifier. This reduction could either be done manually, where the designer of the classifier chooses which features are relevant for classification, or automatically where the feature selection is data driven. This work will focus on testing new methods for automatic feature selection.

The EEG readings are time series. There are two different ways to classify such measurements. A static classifier uses each measurement within the time series as a feature, and does not use the time information. A dynamic classifier instead uses each time series as a feature. Both type of classifiers were studied in this thesis.

5. Goals

This thesis will focus on two tasks.

The first task was to get an overview of the state of the art of EEG classification and understand what methods are being used. This work is presented in chapters 2 and 3. While some methods was described in detail there was no time for a thorough theoretical handling of all methods being found. The interested reader can read further in the references given for methods that were not described with enough detail.

The second task was to design an EEG classifier for the encoding phase of the memory study. First a classifier that was previously used in similar research was implemented as baseline. Then two new classifiers using different feature selection were then implemented with the goal of improving on the baseline.

Limitations

In a field as big as brain computer interface there had to be limitations on what was explored. In this project there was only one feature type, power spectral density via continuous wavelet transform. There was also only one classifier type, support vector machines. These were used in previous work. Instead, only feature selection methods were explored. The usage of an SVM is also motivated by its convexity. Always finding a global maximum for the design of the classifier is very convenient when testing feature selection methods.

There were many possible techniques to explore in this project. Some notable that had to be discarded due to time constraint included outlier rejection or to weight samples differently. Deep learning techniques were also not explored.

6. Report Outline

Chapter 2 gives an overview of the BCI field. Chapter 3 covers classification and Chapter 4 covers the feature reduction method used in this work. Chapter 5 describes the experiment in detail and the framework used for classification. Chapter 6 describes the method used in this thesis and their results are presented in Chapter 7. The final discussion and suggestion for future work is given in Chapter 8.

2

Brain Computer Interface Overview

1. Introduction

A Brain Computer Interface (BCI) is a system that takes inputs from the brain, for example EEG, and translates it to commands on a computer. This is usually done by classification, where several commands are trained. The training consists of the user executing the different commands. The goal is then to build a system which uses this information to classify future commands.

This work will not look into model-based EEG classification. This has been done for seizure detection [Chong, 2013] where the signals are much stronger than for ordinary brain activity. It is not assumed to be impossible to use a model-based approach for normal signals, but such possibilities were not explored.

Instead of making a new literature study, results will be presented from Hwang et al. [Hwang et al., 2013] that reviews the literature from 2007 to 2011. It does not include the most recent research but should still be sufficient for giving an overview of the field.

An illustration of a complete BCI system can be seen in Figure 2.1. The first step is signal acquisition where data is acquired, for example via EEG. Next features are extracted from the data and commands are decided based on those features.

Memory classification is not a BCI system since there are no commands executed. Everything up the the command execution is however the same, and the execution of commands is generally not the hard part of building a BCI system. Memory classification will hence be treated as a BCI system throughout this report.

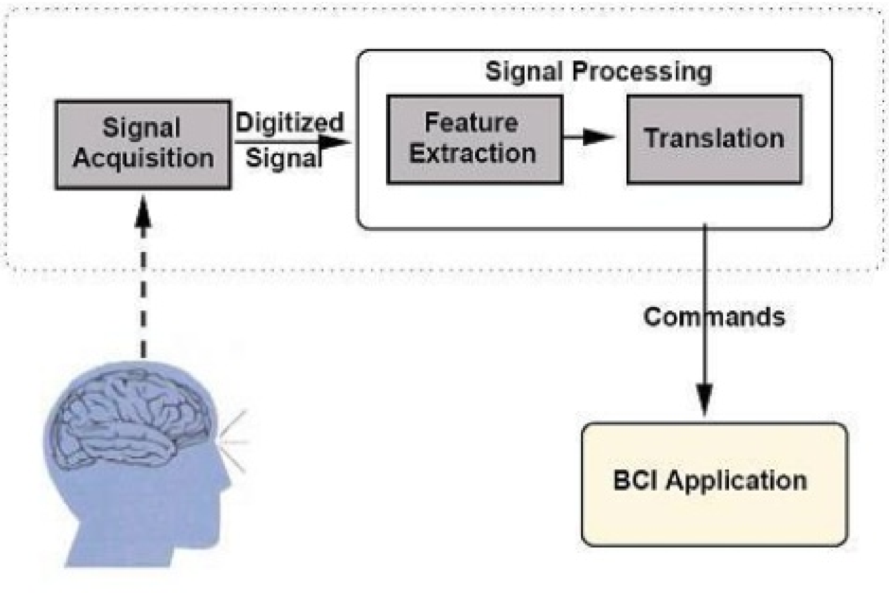


Figure 2.1 An overview of an BCI system taken from [Thorpe et al., 2005]. The signal acquisition might for example be from EEG signals. Features are then extracted and translated to commands via, for example, machine learning.

2. BCI as a Classification System

The following properties and problems need to be considered for all BCI systems that rely on classification [Lotte et al., 2007].

Noise and Outliers: BCI features are very noisy and might contain outliers due to the poor signal-to-noise ratio of EEG signals (or other measurement methods).

High Dimensionality: Due to the many channels and potential features per channel the feature vectors for BCI systems are usually of very high dimension. This can be handled by feature reduction or by a classification method that can handle many features.

Time Information: BCI contains information over time. This could be handled by concatenating feature vectors for different time points or classifying time series.

Non stationary: EEG signals often vary over time, and it might even be the change that is interesting.

Small training sets: It is usually very time consuming to get large training sets. This is usually not feasible since the test-person could get fatigued. Hence the classifier method needs to work well even with a low number of training data.

3. Sources

There are many methods for reading signals that measure the brain activity. EEG is most commonly used and also the choice for this work. Hence it will be given a more thorough description than the other methods. The sources can be split into non invasive methods where readings are done from outside the skull and invasive where the readings are done from inside the skull. Invasive methods give better readings but should of course be avoided whenever possible due to the danger of them. Hence almost all future BCI systems should be using non invasive methods.

[Hwang et al., 2013] found that in the years 2007-2011 32% of sources were invasive, and 60% used EEG. The remaining 8 % included fMRI and MEG introduced later.

EEG

Electroencephalography (EEG) uses electrodes placed on the skull to measure electric activity in the brain. The electrodes are placed according to the international 10-20 system, see Figure 2.2, so that different experiments more easily can be compared. The analog signals are then amplified and sampled by an A/D converter for further handling.

A conductive gel is usually applied to the electrodes to improve reading quality. This can take over an hour, even for skilled users. For an every day BCI system this is not feasible. Dry electrodes are however becoming available. A camera is often used to record eye movement and helps in artifact removal. Such a camera is typically not present on an everyday BCI systems, but the artifact removal could be done without the help of the camera.

The strength of EEG lies in its high temporal resolution and its relative low cost compared to other methods.

What do we measure? EEG measures electrical activity in the brain. One of the ways for the neurons in the brain to communicate is through electrical discharge. The current released from just one or a couple of neurons is too low to be measurable using non-invasive measurements. A large group of neurons must fire in synchronization for its activity to be detected by an electrode. Hence, the strength

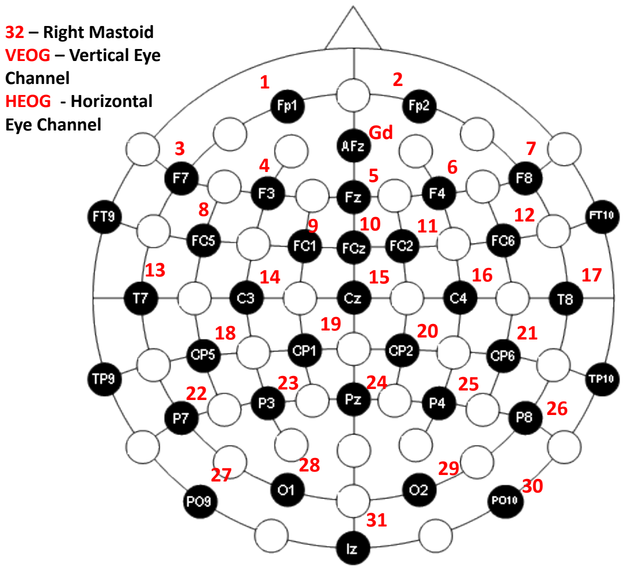


Figure 2.2 Electrode placement for the 10-20 systems as seen from above the skull. The triangle in the upper part of the figure represents the location of the nose. The Electrode are spread evenly over the scalp to get measurements from the entire brain. There are also sensors measuring eye movement. [Image source: Inês Bramão, Department of Psychology, Lund University.]

of the signal depends on the number of active neurons and their synchronization strength [Widmaier et al., 2014].

The EEG signals have for long been categorized into different frequency bands corresponding to different activities. For example, the alpha rythm of 8-13 Hz is present when the subject is relaxed while the beta rythm (>14 Hz) is dominant when the subject is alert. Higher frequency is generally an indication of increased alertness [Widmaier et al., 2014]. Different parts of the brains are also known to be active during different tasks.

Limitations The main limitation of non invasive EEG's is its poor spatial resolution and its poor ability to measure activity deep inside the brain [Srinivasan, 1999]. It is also biased to certain neurons types [Murakami and Okada, 2006] and the signals are also distorted by the skull.

Other methods

Below follows a short introduction to other methods to measure brain activity.

MEG Magnetoencephalography (MEG) is a technique similar to EEG, but it reads the magnetic fields induced by the current in the brain.

MEG typically has hundreds of channels. Furthermore its magnetic readings are less distorted than the electric potential of EEG by the skull.

MEG requires shielding from magnetic fields, including the earth's. This means that it can only be used in controlled environments such as in clinical applications.

fMRI Functional magnetic resonance imaging measures brain activity by detecting changes in blood flow. It uses the principle that active areas in the brain have increased blood flow. The blood flow is detected using a strong magnetic field to align the oxygen nuclei and another field to to locate the nuclei. The method has very high spatial resolution, pinpointing the location in 3d-space, but has low time resolution.

Invasive methods Invasive methods involve taking measurements under the skull. The benefit is that the signals do not get distorted by the skull. It is however a very dangerous procedure and should only be used when necessary.

4. Preprocessing

The EEG signal comes with a lot of noise. Contracting muscles or even moving the eyes will result in big changes in the signal. Because of this the signals need to be cleaned up from such artifacts. In clinical applications this is usually done by a combination of independent component analysis, which tries to separate the signal into independent components [Lee, 1998], and visual inspection. For a BCI system there is no time for manual intervention and the clean up needs to be automated.

Automatic artifact removal is necessary for a BCI system but will not be covered in this work.

5. Features

There are many different possible features to use for classifying EEG signals. A brief overview of methods used in the field will be given, but for details please see the references given below. As this project only considered one type of features there will be no discussion of the different features strength and weaknesses due to time constraints.

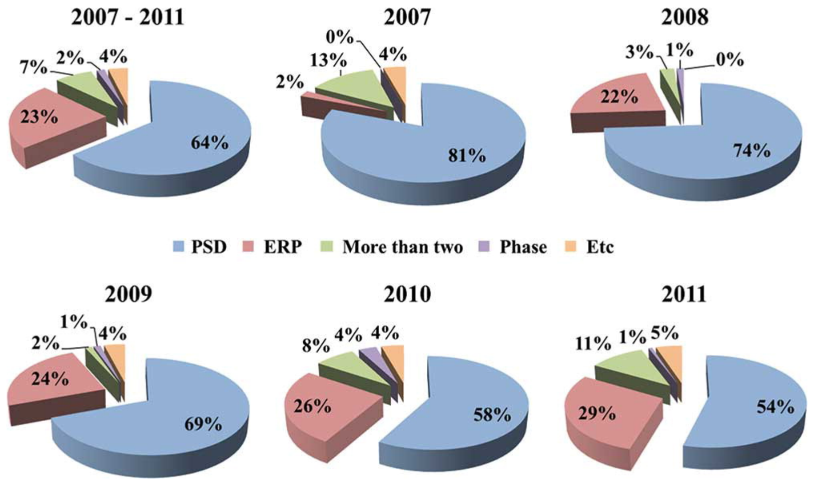


Figure 2.3 Overview of features used in BCI research. [Hwang et al., 2013]

Frequency

The Fourier transform of a signal $x(t)$ defined for all $t \in \mathbb{R}$ is given by

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}\, \mathrm{d}t. \tag{2.1}$$

The spectrogram of $x(t)$ is given by $S(f) = |X(f)|^2$ and serves as an estimation of the magnitude of different frequencies. This method has two obvious problems. Firstly, it requires data on an infinite time domain, and secondly, the frequency is the same for all time points. These two problems are attempted to be solved by the so called short time Fourier transform which uses a window w to locally calculate an approximation of the Fourier transform.

$$X(\tau, f) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i2\pi ft}\, \mathrm{d}t \tag{2.2}$$

The window function is usually symmetric and of unit $\mathcal{L}^2$ norm. The spectrogram now has a time component and is calculated as $S(t, f) = |X(t, f)|^2$. For further treatment see [Sandsten, 2016].

For a window function with support in a finite interval, infinite future samples are no longer needed, but for calculations for time t there still must be measurements up to time $t + \Delta$ where Δ depends on the support of the window function. Lower frequencies require larger delays.

Since the data will be used in a computer there is no use for continuous frequency measurements. The short time Fourier transform has a discrete counterpart

$$X(m,f) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-i2\pi n} \tag{2.3}$$

There are many ways to calculate the frequencies of the signals. Some of those methods will be briefly introduced here for completeness, but this work does not attempt to find the best method or finding the best configuration for the method of choice.

Multitapers Multitapers use several independent windows to estimate from the same data segment. The final estimation is then the average of all windows which reduces the variance of the estimation, see [Percival and Walden, 1993] for further treatment.

Continuous Wavelets One limitation of the short time Fourier transform is that it uses the same window function for all frequencies. Using a big window function will lead to bad time resolution and a small window will lead to bad frequency resolution. The continuous wavelet transform of $x(t)$ is calculated using a wavelet function $\psi_{s,t_0}(t)$

$$W(s,t_0) = \int_{-\infty}^{\infty} x(t)\psi^*_{s,t_0}(t)\,\mathrm{d}t, \tag{2.4}$$

where x^* denotes the complex conjugate of x, t_0 is the center of the windows and s is the timescale. For some windows $s = \frac{1}{f}$ but this is generally not true. The wavelet is calculated from the mother wavelet

$$\psi_{s,t_0}(t) = \frac{1}{\sqrt{s}}\psi_0\left(\frac{t-t_0}{s}\right). \tag{2.5}$$

Notice how the wavelet gets smaller for larger frequencies. See [Hramov et al., 2015] for further discussion.

Autoregressive coefficents

An autoregressive (AR) model is used to describe a random process where the next value depends linearly on some of the previous values, and a white noise e. For a scalar AR model of order p this can be described as

$$x[t] = \sum_{i=1}^{p} \alpha_i x[t-i] + e[t]. \tag{2.6}$$

For further discussion and how to calculate the coefficents α_i, see [Jakobsson, 2013]. For EEG this needs to be extended to be time varying and multivariate. For an adaptive autoregressive process, $\alpha_i(t)$ is a function of time. This was used in [Schlögl et al., 1997] where it was argued that the random nature of the process well described the random nature of the EEG data.

The AR coefficents can then either directly be fed into a classifier or used for frequency estimation [Krusienski et al., 2006]

$$\hat{S}(e^{i2\pi f}) = \frac{\sigma_e^2}{|1 - \sum_{i=1}^{p} \alpha_i e^{-j2\pi f}|^2}. \tag{2.7}$$

where σ_e is the variance of the white noise e.

Hjorth parameters

Hjorth parameters were introduced by Bo Hjorth [Hjorth, 1970]. They consist of three quantities which can be defined in the frequency domain but which also can be calculated in the time domain.

Activity: The variance, which is the same as the power, of the signal $\mathrm{var}(y(t))$.

Mobility: Describes the mean frequency and is calculated as

$$\mathrm{mobility}(y(t)) = \sqrt{\frac{\mathrm{var}\left(\frac{\mathrm{d}y}{\mathrm{d}t} y(t)\right)}{\mathrm{var}(y(t))}} \tag{2.8}$$

Complexity: Describes the change in frequency and is calculated as

$$\mathrm{complexity}(y(t)) = \frac{\mathrm{mobility}\left(\frac{\mathrm{d}y}{\mathrm{d}t} y(t)\right)}{\mathrm{mobility}(y(t))} \tag{2.9}$$

Phase synchronization

Most frequency methods only look at the amplitude, ignoring the phase. AR coefficients also ignore the phase of the signals. The phase information could be very relevant since two parts of the brain having the same phase indicate that they are cooperating [Brunner et al., 2006]. There are multiple measures of synchronization but they will not be presented here.

Phase has not been popular in BCI research. This might partly be due to BCI being a very m alsoulti-disciplinary research field and phase features require more mathematical involvement.

An example of when it is used is [Daly et al., 2012] where functional connectivity is used. Functional connectivity is defined as communication between different regions and is identified via phase synchronization.

Event-Related Potential

Event-related potential (ERP) is the change in potential that happens due to stimuli. The most commonly used ERP is P300. P300 reacts to a visual stimuli and after about 300 ms the ERP can be observed. A common setup for a P300 BCI system is a matrix of example letters where the user focuses on the one he or she wants to write. Flashing the row or column of the desired letter elicits a P300 while the others do not. The P300 ERP can be detected by spectral methods [Fazel-Rezai et al., 2012].

ERP is a common method in BCI research. However, it will not be discussed further since it is far from the rest of the work.

6. Different types of BCI systems

BCI systems can be split into different categories. What follows is a summary from [Nicolas-Alonso and Gomez-Gil, 2012]. The first distinction is between systems that rely on external stimuli and those that do not.

Exogenous: Exogenous BCI system uses neural activity that originates from external stimuli. This method have the advantage of requiring only one channel, very little training and having a bit rate of up to 60 bit/min. It does however require permanent attention to the stimuli, limiting its use in industrial applications. A P300-based BCI system is an example of an exogenous system.

Endogenous: Endogenous BCI does not rely on any stimuli. It instead relies on the user self regulating the brain rhythm. This technique has the advantage of being operated when the user wants to, and without having to focus on the stimuli. It does however require a lot of training (weeks or months) where the trainie has to learn to produce specific patterns. This is done via neuro feedback, where the user is presented how close to a pattern he is. Multi-channel EEG recordings are also required for good performance. The bit rate is typically 20-30 bits/min. A system based on imagined movement is one example of an endogenous system.

BCI systems can further be split into *synchronous*, which only analysis the signals during pre defined time bins, and *asynchronous*, which always looks at the signals seeing if there is a command present. Synchronous is much easier to implement, but asynchronous offers much more seamless interaction.

Endogenous asynchronous is the most interesting system, but also the hardest!

7. State of the art

This work did not have time for a complete overview of the start of the art results. Instead the current capabilities of the BCI technology will be highlighted by presenting some results in finger tapping classification.

Three properties of a BCI system are of interest: its speed, its accuracy, and the number of possible outputs. The bit rate, or information content the classifier is able to achieve depends on these three properties. A high bit rate is wanted, but if the accuracy of the system is too low, it will be unsatisfactory to use. No bit rate calculation was done in this work.

Blankertz et al. [2006] achieved a mean accuracy of 89.5 % on imagined movement for detection of continuous tapping over trials of 2 seconds. 128 EEG channels were used which is a lot of channels. It did however require very little training with the help of presenting the user with feedback. The highest bit rate was at 35 bits/min with an accuracy of 98 % while one subject had 15 bit per min and another could not achieve any BCI control, illustrating the variability in current BCI performance.

Ang et al. [2008] achieved an accuracy of 76.7 % on imagined taps on 2 second intervals. This study used 118 electrodes.

The cited work above managed to achieve good accuracy, but at the cost of only being able to detect continuous tapping, and only in segments of 2 seconds. This would lead to very delayed control. Daley et al. [2011] tried to identify single imagined taps for different time intervals. For intervals of 1 second an accuracy slightly above 80 % was achieved. 0.5 second gave an accuracy around 63% and was the smallest window that was significant above chance on a $\alpha = 0.01$ level. This work only used 19 channels.

It can be noted that high accuracy can only be achieved by low time resolution, limiting the information content that can be extracted from the sensor.

3

Classification

1. Introduction

The goal of the classification in this work is to be able to distinguish between the three different memory categories by using the EEG data. To do so, the computer must be able to recognize the different patterns that arise in the measurements of the electrical potential of the brain. This is a standard problem in the field of machine learning.

The problem could be viewed as a decision problem. The machine is presented with some data and is then expected to make a correct decision based on that data. Below follows some examples of general decision problems to illustrate the concept of decision problems.

- Given a black jack hand, should the player hit or stand?
- Given measurements about wind and temperature, what will the temperature be in two days?
- Given a picture of a face, who is the person?

For a black jack hand, the decision could be made by understanding the mathematics of blackjack. For the weather prediction, the prediction could be made explicitly by models of the weather which have been learned from previous experience. For face recognition, it is common that the system has a set of training pictures which it uses to make a prediction when presented with a new picture. For all these examples it is clear that the decisions require some prior information about the problem.

In this work, the data was EEG signals and the goal was to make some decision based upon them. In general, that decisions could be where to move the cursor on a screen or how to move a prosthetic limb. Specifically in the memory experiment, the decision that had to be made was which of the three categories the user has tried to remember.

Old samples with known memory category will be used to classify new ones. This is called supervised classification and is the only machine learning technique that will be used in this work. Supervised means that the labels of the training data are known. Classification means that the output of the learning system is discrete, for example "hit" or "stand". For regression based learning the output is instead continuous, for temperature prediction the temperature could take on any value. For unsupervised learning the machine learning is presented with data samples with unknown label. Part of the learning task is to cluster which training data that belong together, for example which pictures that depict the same person.

Outline of the Classification Problem

The previous discussion is now formalized for the case of supervised classification. The task is to use a set of old training samples $X = \{\vec{x}^{\,i}\}$ to classify new samples $\vec{x}_{new}$. Each sample contains a set of features $\vec{x}^{\,i} = x_1^i, \ldots, x_n^i$. Choosing a good feature set is an important part of getting good classifier performance and will be discussed further in the next chapter.

Classification is made by designing a *decision rule* that assigns each feature vector $\vec{x}$ to a class y_i based on the samples in the training set. The decision rule can then be used to classify both old and new samples. Classifying the training set gives an accuracy called the *training accuracy*.

The goal of the decision rule is to find the structure of the classes. A good training accuracy is important as it can not be expected to do better on new data than on older data. But it is the accuracy on a previously unseen set, called the testing test, that is the true measure of the classifier's performance. That accuracy is often called *testing accuracy*. Using a model that is too complex will often lead to good training accuracy, but bad testing accuracy as the model finds the structure in the training set instead of the class structure.

Sometimes there are parameters in the classifier that have to be chosen by the user. These are sometimes chosen in a data driven fashion by splitting the data into three sets: Training, testing and evaluation. The classifier is trained on the training set for different parameter configurations and evaluated on the testing sets. This could be done by searching on a grid in the parameter space or by using some more advanced search technique. The classifiers are evaluated on the testing set and the one that performs best is then tested on the previously untouched evaluation set to yield its generalization performance. Note that if the accuracy of the testing set was to be used the classifier would be evaluated on data it has already seen!

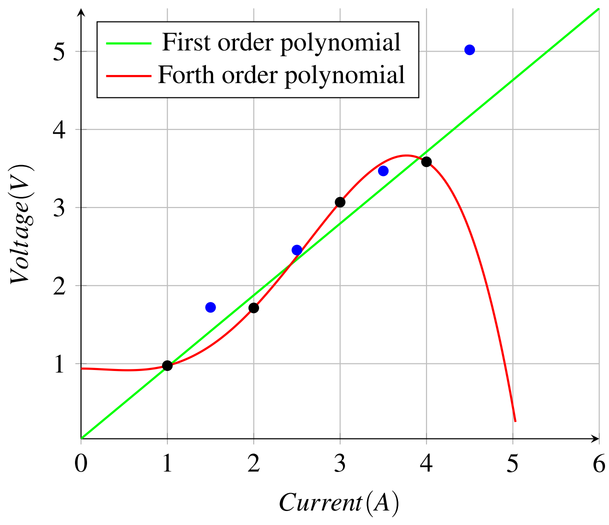


Figure 3.1 Example of using a too complex model. Black dots denote training samples and blue denote testing samples. Green line is first order polynomial and red line is fourth order polynomial. The fourth order polynomial has perfect training accuracy but that does not mean that it is better at predicting future data.

Example: Polynomial interpolation To illustrate the concepts above a similar problem will be presented that should be known to most of the readers. Imagine that given n measurements (x_i, y_i) one wants to predict y_i given x_i. The variable x could for example be the current in a circuit and y the voltage. It is known that as long as the resistance is constant there will be a linear relationship between the two quantities. Due to measurement noise and other disturbances the relationship will not be perfectly linear, but instead on the form $y = ax + e$ where e might for example have normal probability distribution. If one tries to fit a first degree polynomial to the data it will most likely be impossible to get a perfect fit and there will be some error on the set used to calculate the line. If instead a polynomial of degree $n-1$ or higher is used, the curve will perfectly fit the samples. The generalization performance will however most likely be worse. The result of using a first and fourth order polynomial on four data points can be seen in Figure 3.1. □

2. Example of some classifiers

Next some classifiers will be presented. The purpose is twofold, to further illustrate the classifier problem and also to present some of the popular classifiers used in EEG classification. Since none of these classifiers will be used in the experiment of this project the section can be skipped at the reader's discretion.

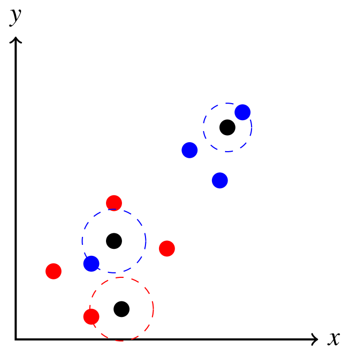


Figure 3.2 Example of Nearest neighbor classifier. Blue and red are the two different classes. Black indicate new samples that should be classified and the circle around them show the closest neighbor. Note how old samples are used to classify new.

Nearest Neighbor Classifiers

The nearest neighbor classifier has a very simple decision rule. For each new sample it assigns the label of its closest neighbor in the training set. See Figure 3.2 for a visualization. The classifier is generalized to k-Nearest neighbor classifier where each new sample is assigned to the class that have the closest k'th neighbors. This classifier is less sensitive to samples that are different from the rest of its class, called outliers [Duda et al., 2000].

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a two-step classifier that starts with a feature transformation that transforms the data into a lower dimensional space where the classes should be as separated as possible. Here only the feature reduction will be covered. The reason it is covered is because its frequent use in brain computer interface research and that it illustrates well how a classifier tries to seperate the data. Only the two class LDA will be covered. For the general case see for example [Duda et al., 2000].

The goal of LDA is to find the projection onto a line that is best for discriminating the data. This can mathematically be described as $y = \vec{w} \cdot \vec{x}$ where $\vec{x}$ is the original feature representation and y is the new scalar representation of the sample. The goal is to find the weight vector $\vec{w}$ with unity magnitude that best separates the two classes, see Figure 3.3 for an illustration. There are many ways of measuring the separation of the classes. LDA uses the sample mean defined as

$$\vec{m}_i = \frac{1}{n_i} \sum_{\vec{x} \in \mathscr{D}_i} \vec{x} \tag{3.1}$$

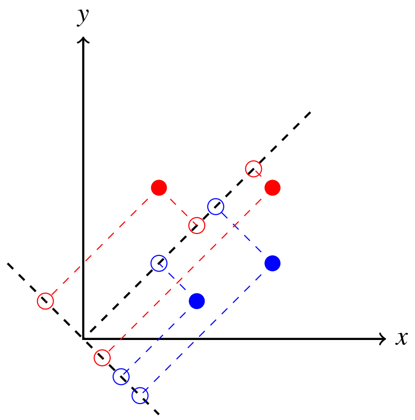


Figure 3.3 An illustration of projection of points onto two different lines. The solid circles are the samples and the color indicates their class. The hollow circles indicates their projections. It can be seen that for one of the projection the two classes are separated while it is not on the other.

where n_i is the number of samples in class i and $\mathcal{D}_i$ is the set of samples in class i. For the projected points the sample means are

$$\hat{m}_i = \frac{1}{n_i} \sum_{y \in \mathcal{Y}_i} y = \frac{1}{n_i} \sum_{\vec{x} \in \mathcal{D}_i} \vec{w} \cdot \vec{x} = \vec{w} \cdot \vec{m}_i \tag{3.2}$$

due to the linearity of the dot product. The difference in the projected means is then

$$|\hat{m}_1 - \hat{m}_2| = |\vec{w} \cdot (\vec{m}_1 - \vec{m}_2)|. \tag{3.3}$$

This difference can be made arbitrarily large by increasing $\vec{w}$ but that will just be a scaling. This is why $\vec{w}$ was forced to be of unit magnitude. To get good discrimination the difference in mean should be large compared to the variance of each class. To that end, define the within-class scatter

$$\hat{s}_i^2 = \sum_{y \in \mathcal{Y}_i} (y - \hat{m}_i)^2. \tag{3.4}$$

The total within-class scatter is then simply $\hat{s}_1^2 + \hat{s}_2^2$. A reasonable goal is to find the $\vec{w}$ that maximizes

$$J(\vec{w}) = \frac{|\hat{m}_1 - \hat{m}_2|^2}{\hat{s}_1^2 + \hat{s}_2^2}. \tag{3.5}$$

There are now two tasks left: Finding the optimum $\vec{w}$ and finding the decision point on the projection.

It would be good if $J(\cdot)$ depended explicitly on $\vec{w}$. To find such a description define the scatter matrices

$$\boldsymbol{S}_i = \sum_{x \in \mathscr{D}_i} (\vec{x} - \vec{m}_i)(\vec{x} - \vec{m}_i)^T, \quad \boldsymbol{S}_w = \boldsymbol{S}_1 + \boldsymbol{S}_2. \tag{3.6}$$

$\hat{s}_i^2$ can then be rewritten as

$$\begin{aligned} \hat{s}_i^2 &= \sum_{y \in \mathscr{Y}_i} (y - \hat{m}_i)^2 = \sum_{\vec{x} \in \mathscr{D}_i} (\vec{w}^t \vec{x} - \vec{w}^t \vec{m}_i)^2 \\ &= \sum_{\vec{x} \in \mathscr{D}_i} \vec{w}^t (\vec{x} - \vec{m}_i)(\vec{x} - \vec{m}_i)^t \vec{w} = \vec{w}^t \boldsymbol{S}_i \vec{w}. \end{aligned} \tag{3.7}$$

The sum of the scatters can then be expressed as

$$\hat{s}_1^2 + \hat{s}_2^2 = \vec{w}^t \boldsymbol{S}_w \vec{w}. \tag{3.8}$$

The difference in means can be written as

$$\begin{aligned} (\hat{m}_1 - \hat{m}_2)^2 =& (\vec{w}^t \vec{m}_1 - \vec{w}^t \vec{m}_2)^2 \\ =& \vec{w}^t (\vec{m}_1 - \vec{m}_2)(\vec{m}_1 - \vec{m}_2)^t \vec{w} \\ =& \vec{w}^t \boldsymbol{S}_b \vec{w}, \end{aligned} \tag{3.9}$$

where $\boldsymbol{S}_b = (\vec{m}_1 - \vec{m}_2)(\vec{m}_1 - \vec{m}_2)^t$. Thus the expression for $J(\cdot)$ becomes

$$J(\vec{w}) = \frac{\vec{w}^t \boldsymbol{S}_b \vec{w}}{\vec{w}^t \boldsymbol{S}_w \vec{w}}. \tag{3.10}$$

To maximize $J(\cdot)$ $\vec{w}$ must satisfy

$$\boldsymbol{S}_b \vec{w} = \lambda \boldsymbol{S}_w \vec{w}, \tag{3.11}$$

which is a generalized eigenvalue problem [Duda et al., 2000].

For the n category case the data is typically instead projected onto the $n-1$ orthogonal directions that best discriminate the data, see [Duda et al., 2000].

Hidden Markov Models

In image classification there is no temporal component that could be used, but for an image stream or for EEG readings it could be useful to use that there is information about which order the measurements come in. Hidden Markov Models have proven to be useful in such problems [Duda et al., 2000]. Hidden Markov models will not be used in the experiments of this work, but it constitutes an alternative method to classifying EEG data which the author think could be interesting.

What follows is a brief introduction of Markov Models and Hidden Markov Models. It might be insufficient for readers that have never dealt with Markov models before and if so there are plenty of books on the subject, see for example [Brémaud, 2013].

Markov Models A (discrete) Markov Model is a random process that contains N states denoted $\boldsymbol{S} = \{S_1, ..., S_N\}$. The process is realized by one state being visited at each time point $\boldsymbol{q} = \{q(1), q(2), q(3), q(4), q(5), q(6)\}$ where $q(i)$ is the state at time i. A possible realization for a three state Markov model could be $\boldsymbol{q} = \{S_1, S_1, S_3, S_2, S_1, S_2, S_2\}$.

The transitions between the states follow the *Markov assumption* saying that the state at time $t+1$ is only dependent on the state of time t. The probability of a transition from one state to another is then fully described by the set of transitional probabilities

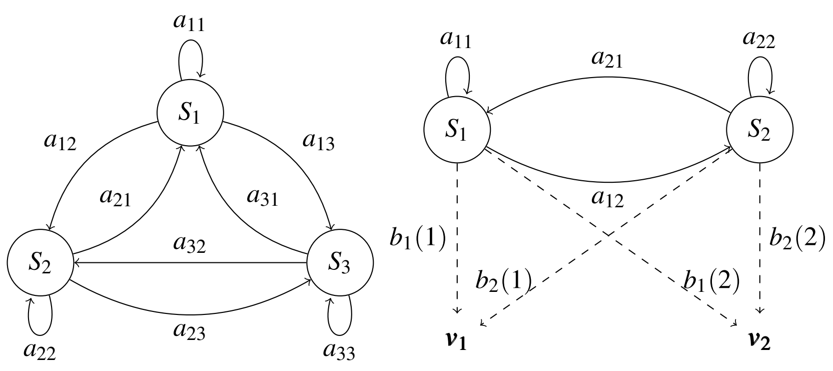
$$a_{ij}(t) = P(q_{t+1} = S_j | q_t = S_i),$$

which gives the probability that the state changes from state i to state j at time t. If all the transition probabilities are time independent, the Markov model is said to be homogeneous. This assumption is very common. An illustration of a Markov chain can be seen in Figure 3.4(a).

The initial state of the Markov model might also be random. The probability of starting in state i is denoted $\pi_i = P(q_1 = S_i)$. For simulation of the model, these probabilities need to be estimated or decided otherwise.

Hidden Markov Models For the hidden Markov model the states are not directly observable but they give rise to some observations $\boldsymbol{O} = \{O_1, O_2, ..., O_n\}$. Each state has its own random distribution for the observations that could be either discrete or continuous. An illustration can be seen in Figure 3.4(b).

For discrete observations the set $\boldsymbol{V} = \{v_i\}$ describes all possible observation values. The distribution for each state's observations is then described by $\boldsymbol{B} = \{b_j(k)\}$,



(a) Markov Model (b) Hidden Markov Model

Figure 3.4 **(a)** Example of Markov model, **(b)** Example of hidden Markov model. The states S_i are not directly observable, but we can see the observations v_i.

where $b_j(k) = P(v_k \text{ at time } t | q_t = S_j)$.

To summarize, the following is needed to design a hidden Markov model with discrete observation space.

1. The number of states N.
2. The possible observations $V = \{v_i\}$.
3. The transition probabilities $A = \{a_{ij}\}$.
4. The distribution of the observations $B = \{b_j(k)\}$.
5. The initial distribution $\pi_i = P(q_1 = S_i)$.

There are three problems that need to be solved to use HMM for classification. Those will be presented, but the solutions see [Duda et al., 2000], [Juang and Rabiner, 1991].

The evaluation problem: Given a model $\lambda = (A, B, \pi)$ and a set of observations $\boldsymbol{O}$, compute $P(\boldsymbol{O}|\lambda)$ in an efficient way. This is the probability that the model generated the observations.

The decoding problem: Given a sequence of observations $\boldsymbol{O}$ and a model λ determine the most probable set of states $\boldsymbol{q}$.

The learning problem: Given a set of observation sequences $\{\boldsymbol{O}_i\}$, determine the set of parameters $\lambda = (A, B, \pi)$ to maximize $P(\boldsymbol{O}|\lambda)$.

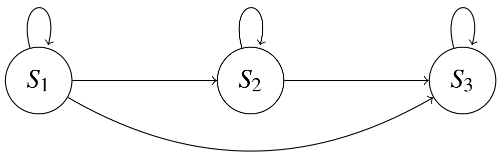


Figure 3.5 Example of a left right Markov model. Once a node has been left, it can not be revisited.

Example: Speech Recognition We now try to illustrate how we can build a classifier using the solution to these problem by presenting an example from [Juang and Rabiner, 1991].

Consider the classification of individual words. Assume that each word can be represented as a time sequence of M unique spectral vectors. This is done by mapping each time segment to the closest spectral vector. So for each word there is a sequence of observations $\boldsymbol{O}$. By using the solution to the learning problem a hidden Markov model can be built for each word.

By using the decoding problem it is possible to examine how good our model seems to be and make adjustments to the number of states N.

Finally, the solution to the evaluation problem can be used to classify a new word by calculating which model that was most likely to have generated the test word's spectral vector. □

Different HMM For speech recognition it makes sense to allow the process to return to a node that has already been visited. But for some process it might be beneficial to force the process to move forward by only allowing state transitions in one direction. This can be implemented with a so called left right model, see Figure 3.5 . This has been used in BCI research [Obermaier et al., 2001].

3. Algorithm independent theory

Introduction

In this section properties that are general for all machine learning algorithms are discussed. The two interesting theoretical questions "Is there an inherently stronger classifier" and "Is there a best feature representation?" are answered. Next, generalization performance is studied using the concepts of overfitting and underfitting.

Superior Classifier

No Free Lunch theorem As stated in the introduction the goal is to get good generalization performance. An interesting question is if there is any classifier that can be considered generally superior, or if there even is one that is superior to chance. The answer to this question is given by the No Free Lunch Theorem and it states that in the absences of assumptions all classifiers are expeced to have the same performance. For a formal statement of it see [Duda et al., 2000].

Ugly Duckling theorem So there is no reason to lean towards any classifier before the problem is taken into account. What about the choice of features? The answer is given by the Ugly Duckling theorem, which states “There is no problem-independent or "best" set of features or feature attributes” [Duda et al., 2000].

Note that the implication of this theorem is only that we can not choose a feature set before taking the problem into account.

Classifier Complexity

In the introduction it was said that having a too complex classifier could lead to bad testing accuracy even though the training accuracy was good. This is often called *overfitting* and can be simplified as having two different sources [Hawkins, 2004]:

- Using a model that is more advanced than it need to be. For example, if the data can be separated linearly it might do more harm than good to use a quadratic decision surface.
- Using irrelevant or too many features.

The first part can be counteracted when designing the classifier and the second part is counteracted during feature selection. It is, however, important that the feature selection is adapted to the classifier that is later going to be used.

Due to the No Free Lunch theorem no classifier, including a simpler one, can be expected to be better before taking the problem into account. Focusing too much on avoiding overfitting could lead to *underfitting*, which means both bad training and testing accuracy. This can be explained by

- Using a model that is too simple
- Using too few features.

Clearly, avoiding overfitting might lead to underfitting and vice versa. This will be formalized with the help of the bias-variance tradeoff in the following paragraphs.

Bias and variance for regression Even though the rest of the report does not deal with regression it will be used here since it is a clear illustration of the bias and variance for a classifier.

The goal of regression is to estimate the function $F(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ with the help of training data $\mathscr{D} = \{(\boldsymbol{x}_i, y)\}$. This gives the regression function $g(\boldsymbol{x}; \mathscr{D}) : \mathbb{R}^n \to \mathbb{R}$ which is the estimation of $F(\boldsymbol{x})$. The effectiveness of the regressor could be measured by its mean squared error averaged over all possible training data sets $\mathscr{D}$.

$$E_{\mathscr{D}}[(g(\boldsymbol{x};\mathscr{D}) - F(\boldsymbol{x}))^2] = \underbrace{(E_{\mathscr{D}}[g(\boldsymbol{x};\mathscr{D}) - F(\boldsymbol{x})])^2}_{\text{bias}^2} + \underbrace{E_{\mathscr{D}}\left[(g(\boldsymbol{x};\mathscr{D}) - E_{\mathscr{D}}[g(\boldsymbol{x};\mathscr{D})])^2\right]}_{\text{variance}} \tag{3.12}$$

It is noticed that the expected error has two terms, the bias and the variance. The bias is how far off it is expected to be on average. It should be intuitively clear that higher bias leads to lower accuracy without looking at the formula above. Variance measures how much the accuracy of the guess varies. Clearly a large variation in accuracy means that the total accuracy suffers.

Now all that has to be done is to lower the bias and variance. The problem is that once a model has been chosen it is hard to decrease them both at the same time. Making the model more complex will lead to lower bias but higher variance. While making it less complex will lead to lower variance, but higher bias. In the polynomial fitting example earlier the first degree polynomial has some bias and low variance while the fourth degree polynomial has zero bias, since over any training set the error will be zero, but high variance.

With this in mind the design of a classifier could be seen as a two step process. First a classifier model has to be chosen. It is important to chose one that fits the problem well to reduce the bias and variance. One can also try to get as many training samples as possible to further reduce bias and variance. The second step is finding the optimal complexity for the model which gives the best combination of bias and variance, i.e., the best compromise between over- and underfitting.

If one, for example, chooses a hidden Markov model with sufficiently many states, all training samples can be classified correctly but test performance will most likely be bad. If one instead chooses too few states the model will perform badly both in training and in testing.

See Figure 3.6 for an illustration of the concept.

For classification the error is instead a product of the bias and the variance [Duda et al., 2000], but the same principles still apply.

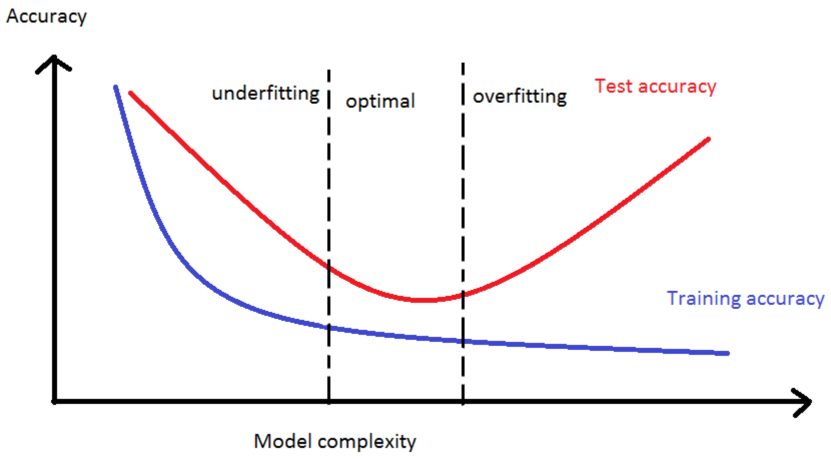


Figure 3.6 Illustration of the bias variance trade-off. A too simple model leads to bad training and test accuracy. Using a too complex model will give good training accuracy, but bad test accuracy as the model finds all the patterns in the training set instead of the general patterns. The goal of the designer should be to find the complexity that maximizes test accuracy.

4. Support Vector Machines

A support vector machine (SVM) uses a linear decision boundary to classify new samples. The training samples are for now assumed to be linearly separable and the case for which they are not will be handled later. There are typically many lines that separate the data, see Figure 3.7. The idea of SVM is to choose the decision boundary that maximizes the margin of the training data in the sense that the minimum margin is maximized. The minimum margin is the distance to the decision boundary of the samples closest to it. The reasoning being that this will hopefully maximize the chance for new data to be correctly classified.

A new sample $\vec{x}_i$ can be classified according to

$$\text{if} \quad \vec{w}\cdot\vec{x}_i + b \geq 0 \quad \text{then positive, otherwise negative.} \tag{3.13}$$

So the orientation of line $\vec{w}$ and the bias b must be found. For the training samples it is required that

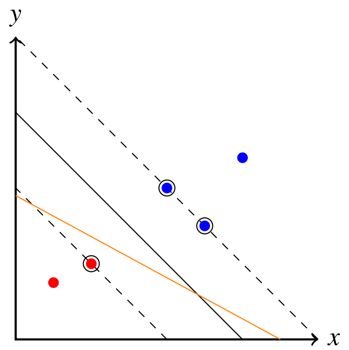


Figure 3.7 An illustration of a support vector machine classifier. The solid black line is the classification boundary that maximizes the margin of the training samples. The circled points on the dashed line are the support vectors. They have the smallest margin to the decision boundary. The orange solid line also separates the data, but its margin is worse.

$$\begin{cases} \vec{w}\cdot\vec{x}_+ + b \geq \;\; 1 \\ \vec{w}\cdot\vec{x}_- + b \leq -1, \end{cases} \tag{3.14}$$

where $\vec{x}_+$ are positive samples and $\vec{x}_-$ negative samples. Equality will correspond to the points with the smallest margin. There will be at least one vector of each class that gives equality, otherwise the margin could be increased further. Those vectors are called support vectors. The goal is to find the $\vec{w}$ and b that maximize the margin. Introduce y_i such that y_i is $+1$ for positive samples and -1 for negative samples. One can then rewrite (3.14) as

$$y_i(\vec{x}_i\cdot\vec{w} + b) - 1 \geq 0 \tag{3.15}$$

with equality for all support vectors. Taking any two support vectors of different classes the margin can be expressed as

$$\text{margin} = (\vec{x}_+ - \vec{x}_-)\cdot\frac{\vec{w}}{||\vec{w}||} \tag{3.16}$$

Using (3.15) gives $\vec{x}_+\cdot\vec{w} = 1 - b$ and $\vec{x}_+\cdot\vec{w} = -1 - b$. The margin can now be written as

$$\underset{\vec{w}}{\arg\max}\,\text{margin} = \underset{\vec{w}}{\arg\max}\,\frac{2}{||\vec{w}||} = \underset{\vec{w}}{\arg\min}\,||\vec{w}|| = \underset{\vec{w}}{\arg\min}\,\frac{1}{2}||\vec{w}||^2. \tag{3.17}$$

So $\frac{1}{2}||\vec{w}||^2$ should be minimized under the constraints in (3.15). This gives the Lagrangian which is to be maximized

$$L = \frac{1}{2}||w||^2 - \sum_i \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1], \tag{3.18}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. This will fall out as a quadratic optimization [Cristianini and Shawe-Taylor, 2000] for which there are many solution methods. There is one more nice property of the SVM left to discover. Differentiating with respect to $\vec{w}$ and b gives

$$\begin{cases} \frac{\delta L}{\delta \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0 \\ \frac{\delta L}{\delta b} = -\sum_i \alpha_i y_i = 0 \end{cases} \Rightarrow \begin{cases} \vec{w} = \sum_i \alpha_i y_i \vec{x}_i \\ \sum_i \alpha_i y_i = 0 \end{cases} \tag{3.19}$$

Since α_i is non zero only for support vectors [Cortes and Vapnik, 1995] the decision boundary is only dependent on the support vectors which make geometrical sense. Substituting the expression for $\vec{w}$ in (3.19) into (3.18) gives

$$L = \frac{1}{2}(\sum_i \alpha_i y_i \vec{x}_i) \cdot (\sum_j \alpha_j y_j \vec{x}_j) - \sum_i \alpha_i y_i \vec{x}_i (\sum_j \alpha_j y_j \vec{x}_j) - b \sum_i \alpha_i y_i + \sum_i \alpha_i \tag{3.20}$$

$$= \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{3.21}$$

and substituting $\vec{w}$ from (3.19) into (3.13) gives for classification of $\vec{x}_{\text{new}}$

$$\vec{w} \cdot \vec{x}_{new} + b = \sum_i \alpha_i y_i \vec{x}_{\text{new}} \cdot \vec{x}_i. \tag{3.22}$$

Note that we only have to calculate the scalar product between the training samples, both for training of the classifier (3.21) and classification of new samples (3.22). If we were to map our data to another domain using a function $\Phi(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^N$ we would need to calculate $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$. This could be beneficial since data not separable in one domain could be in another.

Example [Cortes and Vapnik, 1995]: Imagine we want to have a decision boundary corresponding to a second degree polynomial. We would then from our original features $x_1, ..., x_n$ create the following features $z_1, ..., z_N$

$$\begin{array}{rlcr} z_1 = & x_1,\ \ldots,\ z_n = x_n & : & \text{n dimensions} \\ z_{n+1} = & x_1^2,\ \ldots,\ z_{2n} = x_n^2 & : & \text{n dimensions} \\ z_{2n+1} = & x_1 x_2,\ \ldots,\ z_N = x_n x_{n-1} & : & \frac{n(n-1)}{2}\ \text{dimensions} \end{array}$$

We can see that for a second order polynomial the dimensionality of the new space is much larger than the original one. Calculating $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j) = \vec{z}_i \cdot \vec{z}_j$ could potentially take very long time. This can be avoided by the kernel trick. □

The idea is to replace $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$ with a kernel function $K(\vec{x}_i, \vec{x}_j)$ that never has to map the data to the higher dimensional space. There is some Hilbert space theory for when this is possible. Details will not be covered, but one theorem used later will be stated.

Theorem: Mercer Condition [Cortes and Vapnik, 1995]. A necessary and sufficient condition for that $K(\boldsymbol{u}, \boldsymbol{v})$ defines a dot-product in a feature space is that

$$\int\int K(\boldsymbol{u}, \boldsymbol{v}) g(\boldsymbol{u}) g(\boldsymbol{v})\ \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{v} \geq 0 \tag{3.23}$$

for all g such that

$$\int g^2(\boldsymbol{u})\ \mathrm{d}\boldsymbol{u} < \infty. \tag{3.24}$$

□

For a polynomial of degree d it is known that $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$ can be used [Cortes and Vapnik, 1995]. This is much more efficient than mapping the vectors first and then calculating the scalar product.

Radial basis function (RBF) uses the following kernel

$$K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\gamma |\boldsymbol{u} - \boldsymbol{v}|^2\right), \tag{3.25}$$

which gives desicions on the form [Cortes and Vapnik, 1995]

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i \exp\{\gamma |x - x_i|^2\}\right) \tag{3.26}$$

If γ is chosen big enough 100% training accuracy can be achieved as only local information is then used. Smaller γ will mean that more points are used and the

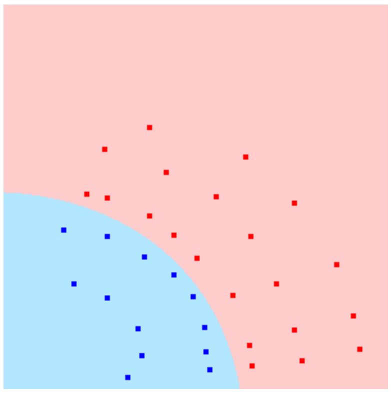


Figure 3.8 An illustration of a decision boundary for a radial basis function kernel. A linear decision boundary is found in the kernel space which then maps to a non linear boundary in the original space.

decision boundary will be less complex. An illustration of a decision boundary from an RBF kernel can be seen in Figure 3.8.

The benefits of SVM is that it guarantees finding a global minimum and that it can efficiently map the data to a higher dimension without paying with much higher computation time. So far it was assumed that the data is linearly separable, at least in some domain. Being restricted to finding a domain where the data is linearly separable would make the method useless for most cases. Luckily this can be counteracted.

Non separable case

For the non separable case we require that

$$\begin{cases} y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \tag{3.27}$$

This allows for some of the samples to be on the other side of the margin or even on the other side of the decision boundary. ξ_i is the distance from the margin. We then minimize

$$\min \frac{1}{2}||\vec{w}||^2 + C\sum_i \xi_i^k. \tag{3.28}$$

We notice that choosing a large regularization parameter C will lead to fewer training errors but possibly lead to overfitting since training errors are heaviliy penalized. The standard choices of k is 1 or 2, leading to the 1-norm and 2-norm soft margin problem. For the solution to those see [Cristianini and Shawe-Taylor, 2000]. It is clear that the 1-norm is less sensitive to outliers.

Even for the separable case it could be useful to allow for some samples to have some error ξ_i to increase the margin. This could lead to lower training accuracy, but also to higher validation accuracy.

Generalization Performance

The regularization parameter C gives the user good control with regards to overfitting. For radial basis function the user have further control with the parameter γ.

SVM have good generalization properties and are insensitive to overtraining [Lotte et al., 2007]. The complexity of a SVM is based on the decision boundary and not the number of features [Joachims, 1998]. Hence for a linear kernel many features can be used while still keeping a low complexity kernel. Using a non linear kernel makes the model more complex and makes overfitting an issue that must be considered.

SVM multiclass classification

The observant reader might have noticed that SVMs can only classify between two classes. There are two standard methods to extended the SVM framework to multiclass problem.

one against one In one vs one classification, each class is paired with all the other classes. The final result is the class that wins the most duels. Ties can be broken as long as the same features are used for all classifiers.

one against all In one vs rest each class is instead fed into a classifier problem were the competing class consists of all other classes. Hopefully only one class wins against the rest, but if multiple do, the resulting tie can be broken.

See [Hsu and Lin, 2002] for further treatment.

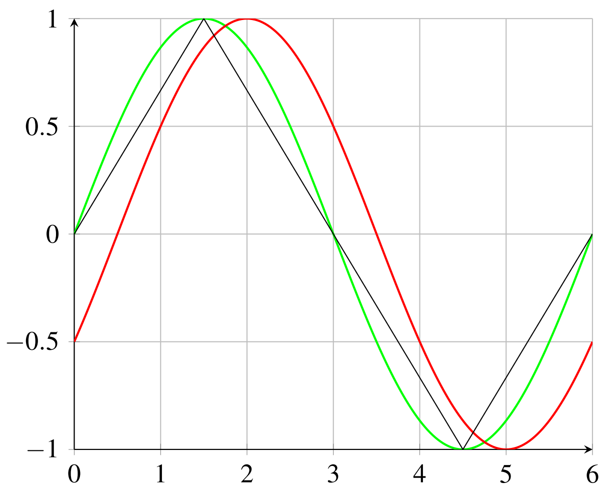


Figure 3.9 An example of how the Euclidean norm might be insufficient to compare time series. The black seesaw signal is closer to the green sinusoid than the red delayed sinusoid is. If one wants to decide if the given signal is a sinusoid or a seesaw signal the time shift is irrelevant.

5. Time Series Classification using SVM

It is possible to classify time series with SVMs using the theory covered so far by using each data point from the time series as a feature. This could, however, be unsatisfying since the time dynamics are not taken into account. If all time points were shuffled randomly (same for all samples) the classification would work out exactly the same. Clearly some information is lost in the shuffling, but it is important to remember that adding explicit time series is not guaranteeing improved performance.

This work will extend SVM to allow for classification of time series via dynamic time warping (DTW) which will be covered next.

Dynamic Time Warping

There are many ways of measuring the similarity of two time series $\boldsymbol{x} = \{x_i\}$ and $\boldsymbol{y} = \{y_i\}$. One might be tempted to use the $\mathcal{L}^2$ norm $|\boldsymbol{x} - \boldsymbol{y}|$. This simple approach has some problems. Two identical signals with just one sample delay could be measured as very different. An illustration of this can be seen in Figure 3.9.

The idea of dynamic time warping is to counteract time shifts and change in time scale by finding the alignment of the signals that has the lowest distance. This is done by defining a warping between the first $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ and the second signal $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$. The warping is defined by the warping sequence $\boldsymbol{p} =$

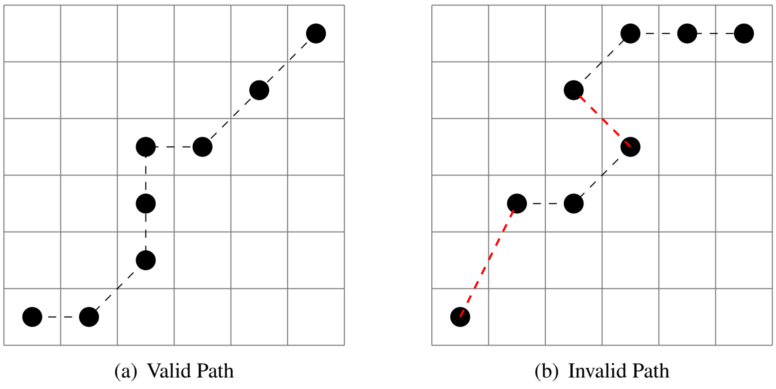


Figure 3.10 (a) shows an illustration of a valid warping. (b) shows an invalid warping where the invalid paths are marked red. The first one takes a too long step and the second one does not satisfy the monotonicity condition.

$(p_1, p_2, ..., p_L)$ where $p_l = (n_l, m_l)$ means that x_{n_l} and y_{m_l} should be compared. The warping path must satisfy the following ["Dynamic Time Warping" 2007]

1. The boundary conditions states that $p_1 = (1,1)$ and $p_n = (N,N)$.
2. Monotonicity condition $n_1 \leq n_2 \leq ... \leq n_n$ and $m_1 \leq m_2 \leq ... \leq m_n$.
3. Step size $p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\}$.

An illustration of a valid path and an invalid path can be seen in Figure 3.10. Note that a warping path might contain more comparisons than the $\mathcal{L}^2$ norm. The cost for a path is then calculated as

$$c_p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}). \tag{3.29}$$

The distance between two sequences is then chosen as one of the potentially multiple minimum paths. Now, only the question of finding the optimum path p is left. Calculating all possible paths would be very slow. This problem is solved by dynamic programming, which allows the calculation to be done in $\mathcal{O}(N^2)$ time complexity.

Multivariate DTW To reach good classifier performance it would be beneficial to use multiple time series. The two simplest ways to handle this would be to either warp them independently or force them to all have the same warp.

Normalization Normalizing each time point independently of the other time points would defeat some of the purpose of the time warping. One possible normalization is to normalize the time series, but then scaling information would be lost.

Using DTW In Support Vector Machines

With the theory covered so far it would be possible to design a k-nearest neighbor classifier using dynamic time warping as the distance measure. There is still some work to be done to use it in a support vector machine.

One might be tempted to define a kernel as $k(\boldsymbol{x},\boldsymbol{y}) = -D_{DTW}(\boldsymbol{x},\boldsymbol{y})$ but this is not a valid Kernel [Gudmundsson et al., 2008]. Neither is the gaussian kernel attempt $k(\boldsymbol{x},\boldsymbol{y}) = \exp\left(\frac{-D_{DTW}(\boldsymbol{x},\boldsymbol{y})}{\sigma^2}\right)$ [Lei and Sun, 2007]. The problem is that they are not positive semidefinite which means they do not satisfy the Mercer condition, stated in (3.23). This means that the optimization problem is no longer convex. While it is still possible to find a separating hyperplane we can not be sure that it is the optimal one. Cuturi [Cuturi, 2011] solved this by introducing the Fast Global alignment kernel.

This work will use the pairwise proximity function SVM formulated by [Gudmundsson et al., 2008]. This method relies on a *proximity function* instead of a kernel function. Let $\mathcal{X}$ be the feature space, then $P : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ maps each pair of feature vectors to their distance.

Given the training set $\{\boldsymbol{x}_i\}$ define

$$\boldsymbol{\phi}(x) = \Big(P(\boldsymbol{x},\boldsymbol{x_1}), P(\boldsymbol{x},\boldsymbol{x_2}), ..., P(\boldsymbol{x},\boldsymbol{x_n})\Big). \tag{3.30}$$

This function maps the features to a new feature space where each feature is the distance to one of the training vectors. P will be the DTW distance and the SVM can be trained using these features.

6. BCI classification

There are two ways to handle the time property of BCI signals. When classifying a time interval a *static* classifier just concatenates the features of different time points into a feature vector for the whole time segment. Using this technique the information at each time point is used. However, the information that time point t_1 comes before t_2 is completely lost. This can be realized by concatenating the time vectors in a different order. The classifier would work exactly the same!

A *dynamic* classifier instead classifies a sequence of feature vectors, like hidden Markov models classifiers. This is a common practice in speech recognition, but not very used in BCI research.

In [Hwang et al., 2013] it was found that the dominant classifiers in BCI research during the period 2007-2011 was Linear Discriminant Analysis (36%) and support vector machines with 17%. The rest were below 5 %.

A classifer for a BCI problem often needs to handle noisy high dimensional data and few training sets.

4

Feature Reduction

1. Introduction

Motivation

In some classification problems there is a lot of measurements that are candidate features for classification. Reducing the amount of features fed into the classifier is called feature reduction. A perfect classifier would have no need for feature reduction since it would be able to ignore irrelevant features. In practice no classifier is perfect but there is still no guarantee that we can improve its accuracy with feature selection. For EEG signals there are a lot of candidate features so it is likely that a good feature selection can improve the accuracy by removing irrelevant features and reducing overfitting.

The original feature input might also be so large that it has considerable impact on the running speed, limiting real time performance. This further motivates the use of feature reduction.

Overview

Feature reduction can be split into two categories, feature selection where only some of the original features are kept and feature extraction where the original feature space is transformed into one with lower dimension. Methods can further be split into supervised, where the labels of the training samples are known, and unsupervised where they are not known.

Feature selection is usually employed when there are many features to start with. This means that the time complexity of the algorithm is very important. An exhaustive search is generally not possible and one has to settle for a near optimal solution. This is usually implemented by searching the feature space by a heuristic or non deterministic approach and finding a useful criterion to evaluate the current solution candidate. Heuristic approaches tend to get stuck in local extrema, while non deterministic approaches need to be run multiple times as the results can differ between runs due to its random nature.

The feature selection is done offline so while it is important to keep the speed reasonably fast, it does not impact the run speed.

The two standard evaluation categories are wrapper methods and filter methods. Wrapper methods is the intuitive choice since they use the classifier accuracy as a metric. However, when an advanced classifier is used, the time it takes to train the classifier for each feature set to be tested might be impractical. Filter methods evaluate the feature set with some other measure. It can for example use correlation or uncertainty measures from information theory [Huang, 2015].

One example of an heuristic seach apporach is Forward Selection. It starts with an empty feature set and adds the feature that improves the evaluation function the most. This contintues until a certain number of features have been added or there are no features that improves the evaluation function [Rejer and Lorenz, 2013]. Backwards selection instead starts with all features and removes the feature that is best to remove one at a time. Forward and backward selection can be combined so that features are first added and then removed.

2. ANOVA

Analysis of variance (ANOVA) is often used when evaluating results of experiments where different parameters are tested [Penny et al., 2011]. This work will use one-way ANOVA, which tests the hypothesis that the means of different classes are the same. This hypothesis is discarded on a confidence level α when there is a $1-\alpha$ chance that the means are all not the same. So the chance that the hypothesis was wrongfully discarded is α. It can also be used as a feature selection method by only choosing features where the means are not all the same.

The data is assumed to be normally distributed for ANOVA. This is generally not true for the EEG data. This does not mean that the tests are useless. A high confidence level will still be a good indication that the means are not all the same.

ANOVA could be seen as a filter method, where the feature set is decided by each feature being evaluated individually using the ANOVA test described above.

When there are only two classes present, a T-test is sufficient to decide if the means are the same. One-way ANOVA is a generalization of this, so the theory for a T-test will be presented first.

T-test

For a T-test, the data is assumed to be drawn from two different distributions $X_i \in N(\mu_i, \sigma_i)$. There are two competing hypotheses

$$\begin{cases} H_0: & \mu_1 = \mu_2 \\ H_1: & \mu_1 \neq \mu_2 \end{cases} \tag{4.1}$$

Using the measurements x_1^i and x_2^i the test quantity T can be calculated

$$T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/N_1 + s_2^2/N_2}} \tag{4.2}$$

where $\bar{x}_i$ is the sample mean, s_i^2 sample variance and N_i the number of samples of each class. The zero hypothesis H_0 is discarded on significance level α if $|T| > t_{1-\alpha/2}$. Or put in other words, H_1 is said to be true with probability $1-\alpha$.

One-Way ANOVA

One-way ANOVA tests the hypothesis that all means are the same versus the hypothesis that they are not all the same. So to be clear, it is enough that two means are different.

Since only the three class case will be used in this work the presentation of ANOVA will only use three classes. This is easily generalized to n classes.

Let the means be described by $\beta_1,\ \beta_2,\ \beta_3$ and the number of samples for each class by $N_1,\ N_2,\ N_3$. The samples are then assumed to be given by

$$y_{ij} = \beta_i + e_{ij} \quad (i = 1,\ \ldots, 3;\ \ j = 1,\ \ldots, N_i) \tag{4.3}$$

Where $\{e_{ij}\}$ is independent and $N(0, \sigma^2)$ distributed. Then two quantities are calculated

$$SS_{between} = \sum_{i=1}^{3} N_i (\bar{y}_i - \bar{y})^2 \tag{4.4}$$

$$SS_{within} = \sum_{i=1}^{3} \sum_{j=1}^{N_i} (y_{ij} - \bar{y}_i)^2 \tag{4.5}$$

where $\bar{y}$ is the sample mean for all samples and $\bar{y}_i$ is the sample mean for all the samples in class i. $SS_{between}$ is a measure of the spread between the means of the classes. SS_{within} is a measure of the spread within the classes.

The test quantity for I classes is $\mathscr{F} = MS_{between}/MS_{within}$, where

$$MS_{between} = SS_{between}/(I-1) \qquad MS_{within} = SS_{within}/(n-I) \tag{4.6}$$

where n is the total number of samples. The hypothesis that all the means are equal are rejected at significance level α, by a comparison with the F distribution, if and only if $\mathscr{F} > F_{\alpha;I-1,n-I}$. For a rigorous treatment see for example [Scheffé, 1959].

3. Correlation-Based Feature Selection

In the general sense two things are said to be correlated if knowing something about one tells you something about the other. Clearly one wants features that are correlated with the classes so that the value of the features can be used to make prediction about the classes.

Correlation-based feature selection is a method for feature selection developed by Hall [Hall, 1998]. It is based on the following principle

"Good feature subsets contain features highly correlated with the class, yet uncorrelated with each other."

The idea is that having features that are correlated adds very little information, while increasing the risk of overfitting.

To penalize correlated features a feature subset $\mathcal{S}$ is evaluated by calculating

$$M_{\mathcal{S}} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}. \tag{4.7}$$

Where k is the number of features in the subset, $\overline{r_{cf}}$ is the average class-feature correlation and $\overline{r_{ff}}$ is the average feature-feature correlation. A high feature-class correlation and a low feature-feature correlation is wanted. So a larger value of $M_{\mathcal{S}}$ is better.

One might then ask what the point is of using correlation for feature selection. Or rather in what scenarios it outperforms ANOVA. For features that separate the classes into two groups, ANOVA will do a good job since their means will be different. There could however be features that separate the samples into multiple groups

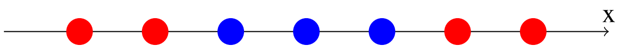


Figure 4.1 An example where ANOVA would not recognize that the feature could be discriminative. Red circles are one class and blue another. The classes have the same mean, but their x value is still correlated with the class. Note that a linear decision boundary would not do well in classifying the samples.

which are well separated, but the classes still have the same mean. In this case the correlation measure could be better. For an example of a situation where the means are the same but the classes still are well separated, see Figure 4.1.

Correlation Estimation

There are three methods proposed in [Hall, 1998] for correlation estimation: relief, minimum description length and symmetrical uncertainty. Symmetrical uncertainty was used in this work due to its ease of implementation. The others will not be explored due to time constraints. All methods require the data to be discretized. Method for this will be covered later.

Symmetrical uncertainty uses the entropy measure which is a measure of the uncertainty of the system, hence the latter half of the name. The entropy of X is given by

$$H(x) = -\sum_{x \in X} p(x) \log p(x),$$

where base 2 is used for the logarithm henceforth. The minus sign makes H a positive quantity since the logarithm of a probability is always negative. If $p(x) = 0$ then $p(x)\log(x)$ is defined as zero, which is consistent with the limit. Entropy can be seen as an information measure. If $p(x) = 1$ for $x = \xi$ then x only takes on that value and no additional information is gained when one is presented with x, as it was already known that $x = \xi$. Conversely if $p(x) = 1/n$ for all x the entropy, or information gained when presented with x is maximum. To see that the uncertainty and the information measure are consistent with each other, note that the uncertainty of a variable is the highest when the information gained when presented with it is the highest.

Next, we define the entropy of X given Y. This is the information gained when presented with X if Y is already known. If X and Y are independent this will be the same as the entropy of X as no new information is supplied by Y. If, on the other hand, the information about Y changes the probability of X the information could be changed. This is mathematically defined as

$$H(X|Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log p(x|y)$$

There are two cases. One where Y is another feature in which case the conditional probability can be written as

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

Clearly the distribution of a good feature will change when the class is known.

The other case is when Y is one of the classes. Then the conditional probabilities will easily be calculated as the probabilities for the features of the samples within that class.

If the entropy for X decreases when Y is supplied then additional information was gained. Define this gain, also called mutual information, as

$$\begin{aligned} \text{gain} &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(Y) + H(X) - H(X,Y), \end{aligned} \tag{4.9}$$

where it is noticed that the last form can be used for calculation. Normalization can be done so that the gains are comparable for different features via

$$\text{symmetrical uncertainty coefficient} = 2 \cdot \left[\frac{\text{gain}}{H(X) + H(Y)} \right].$$

The factor two makes the range to be $[0,1]$.

Discretization

Two simple methods for discretization is equal interval width and equal frequency intervals. Equal interval width splits the data into n bins where each bin covers an interval of size l so that all features are covered. Equal frequency interval splits the data into n bins where each bin contains the same amount of data points.

There are more advanced methods. One based on minimum description will be briefly introduced below. For a full treatment see the original article [Fayyad, 1993].

The idea works iteratively by finding a cut point on a segment. The cut should be such that it minimizes the class entropy of the two new partitions. The entropy of a segment S with k classes is given by

$$H(S) = -\sum_{i=1}^{k} P(C_i, S) \log(P(C_i, S)). \tag{4.10}$$

where $P(C_i, S)$ is the proportion of class i on S. The cut point T is chosen such that it minimizes

$$H(T;S) = \frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2) \tag{4.11}$$

The computation speed is improved by noting that cut points can only be between two classes. They further derive an expression for when keeping a cut based on the minimum description length principle. Basically a cut is kept if the information paid by doing the cut is repaid by better separation of the data.

4. Genetic Algorithm

The Genetic algorithm (GA) is, as the name suggests, inspired by the evolution in nature that leads to fitter and fitter individuals. First the basic algorithm is introduced [Whitley, 1994]. Then some possible variations are presented.

Motivation

The previous two methods only looked at each feature's goodness independently. The main motivation for using a genetic algorithm is its ability to evaluate the feature set as a unit. This is not useful for linearly separable features, but it could be efficient in finding features that only hold discriminative power when combined with another feature. See Figure 4.2 for an example where the x or the y direction have no discriminative power on their own but can classify the data perfectly together.

The Standard Genetic Algorithm

Introduction In nature the genome of populations is changing because fitter individuals have a higher chance of survival and subsequent reproduction compared to those less fit. This makes the populations strive to be fitter. The genetic algorithm tries to achieve the same thing. Now the fitness of the population is explicitly stated as the fitness function $f(\bar{X})$ and the goal is to find the best $\bar{X} = (X_1, ..., X_n)$. For an optimization problem $f(\bar{X})$ would be the function to optimize and $\bar{X}$ the variable for the function.

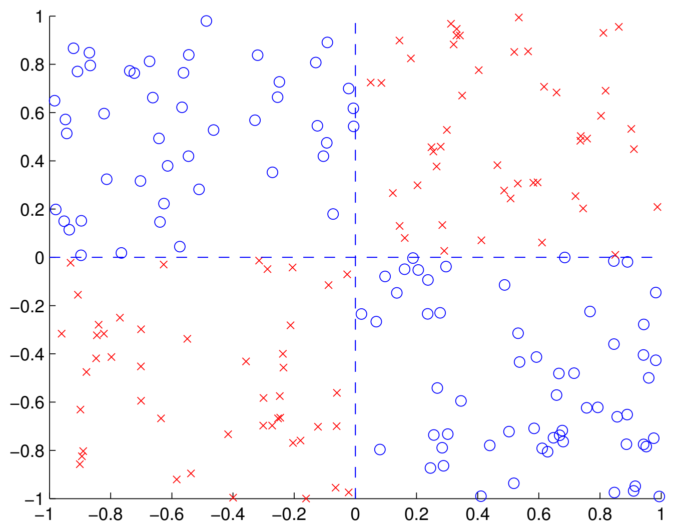


Figure 4.2 An example where the two features have no discriminative power on thier own, but do together. The blue circles and red crosses are the two different classes and the features are the x and y coordinates. Note that no linear decision boundary can do better than chance.

In this work GA will be used for feature selection where the number of features to be chosen are predetermined. The goal is to choose the n best features from the feature candidates numbered $1,...,N$. The variable X_i can take on values in the range $[1,N]$ to represent the feature chosen, but duplicates are not allowed.

If the size of the feature subsets is allowed to change then the encoding need to be binary and $\bar{X} = (X_1,...,X_N)$ and $X_i = 1$ encodes that feature i is present and $X_i = 0$ that it is not.

The fitness function will measure the goodness of the feature subset. When classifier performance is used the method is a wrapper method. It is also possible to use some other measure, making it a filter method.

The algorithm The algorithm is initialized with a genome of individuals. Each individual consists of a vector that encodes that individual's vector $\bar{X}$, This vector is often called a chromosome.

The algorithm then iterates by evolving the genome. The first step to this is calculating the evaluation function which is an absolute measure of how good the current chromosome is, for example the value of $f(\bar{X})$ or the accuracy of a classifier. Next the fitness function is calculated which is the chromosomes relative strength compared to the rest of the genome. This can for example be calculated as $f_i/\bar{f}$ where f_i is the evaluation function of the chromosome and $\bar{f}$ is the mean value of all the evaluation function.

The chromosomes next go through the *selection process*, which yields an intermediate generation where a chromosome with a high fitness score might be present multiple time and one with a lower might not be present at all.

Next is the *recombination* step, where there is a chance that each chromosome is combined with another chromosome generating two new offsprings. This is called a crossover and might, for example, be implemented by choosing a random crossover point and splitting the chromosomes at that point and combining one half from each.

As a final step there is a small chance that an offspring goes through a *mutation* where one of its binary values will flip. This is the end of the iteration, and is followed by a new calculation of the fitness value. A graphical overview of the process can be seen in Figure 4.3.

There are many possible stopping criteria. Examples include terminating after a specific number of iterations, when the change in fitness is below a threshold or when the maximum fitness is above a threshold.

Overfitting in GA When using the wrapper GA there will be a set of samples we can use for training and evaluation. There is a risk that the resulting classifier gets very well-tuned for the data that is presented to it. This is not overfitting in the strict sense that was talked about in classification but the term is extended to cover this case as well. The classical overfitting is still a potential problem if the size of the feature subset is not prespecified. This can be counteracted by punishing genes with many features [Duda et al., 2000].

The tuning problem can be counteracted by not presenting the same data to the fitness function for each generation so that the classifier learns the trends and not the testing set. Liu and Khoshgoftaar [Liu and Khoshgoftaar, 2004] showed that random sampling technique (RST) reduces overfitting. The usage of RST means that only a subset of the training set is used for each generation. Thus presenting different patterns every time.

Why does it work?

There are many different variations and parameter sets for GA and there are no strong theoretical statements for why it works. Instead, we have to believe in the previous results shown, for example in the BCI field [Rejer and Lorenz, 2013], and its good properties.

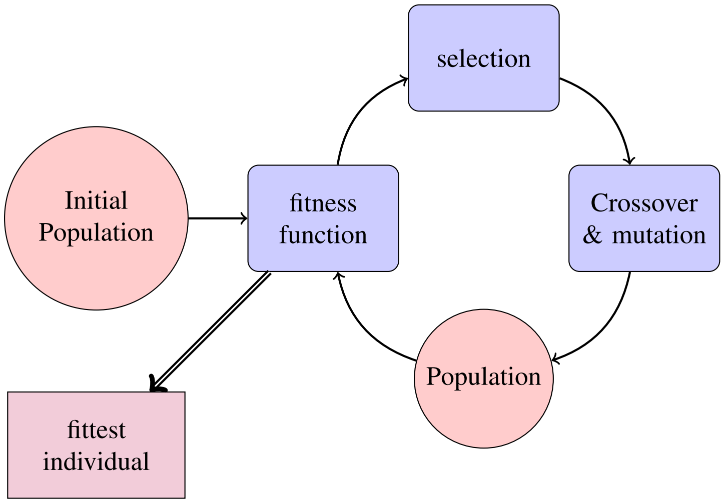


Figure 4.3 A graphical illustration of the genetic algorithm. First an initial population is chosen. Then the fitness value is calculated for each individual. This fitness value gives the probability for each individual to survive the selection process. Those that survive the selection process goes through crossover where 2 genes are combined together and mutation where there is a probability that one feature might swap to another. The fittest individual is extracted when the algorithm terminates.

Extensions

Diversity The diversity of a population in GA is a measure of how spread out the genome is. Failing to maintain a diverse solution might mean that the feature space is not adequately searched, while too high diversity could lead to very slow convergence [D. Gupta, 2012].

One method of maintaining diversity of the population is to use a ranked space where each individual is ranked both on fitness and diversity. The fitness function is then calculated based on this rank. An overview of other methods can be found in [D. Gupta, 2012].

Elitism In nature the parents can not live on for ever but in GA they can. Allowing the parents to fight with their cross-over children for survival into the next generation means that a gene is never replaced with one that has worse fitness during recombination. This might rapidly increase convergence, but at the cost of lower diversity.

Parameter Control For the classic genetic algorithm the crossover rate and mutation rate are set to be constant. The genetic algorithm is a very dynamic process so it is reasonable to use dynamic parameters. It might, for example, be better to have higher mutations rate in the beginning to help explore the search space and lower mutation rate later to fine tune the solutions.

Parameter control can be classified as deterministic in which it changes deterministically for each generation, as adaptive where it changes depending on the properties of the entire population or and self adaptive where the parameters themselves are encoded in the genome and undergoes crossover and mutation. For a thorough representation see [Eiben et al., 1999].

5

Experimental Setup

1. The Study

The data for this project was supplied by the Department of Psychology at Lund University. The experiment is motivated by their research in memory representation. A visualization of the setup can be seen in Figure 5.1.

The study was conducted in a Faraday cage at the department of Psychology at Lund University. The subjects were Swedish native speakers without any known neurological or psychiatric disorders. Both males and females were participating.

The stimuli material of the study was 64 different pictures from the three categories "familiar faces", "landmarks" and "objects", yielding 192 pictures in total. The study was split up into three parts. The first part was the familiarity task where the participant is shown a picture on a computer screen for a couple of seconds. The pictures were taken from one of the three categories. The participants were then asked how well it represents the concept of the picture. This could in theory be used to remove pictures that were not recognized by participants, but was not used in practice.

The second part was the study phase for which 24 word-picture pairs were presented and the participants was asked to try to remember the pair and also rate the association between the pair.

Finally for the test phase the participant was shown a word and needed to retrieve the category of the picture it was associated with. If they were correct they were then also shown the original and a mirrored picture and had to choose which one they thought they saw earlier. This was to force the participants to visualize the picture, otherwise the retrieval is only conceptual. The goal of this was to increase the classifier accuracy.

The study took approximately two and a half hour with breaks between each part.

The goal of the study at the psychology department was to study if any of the pat-

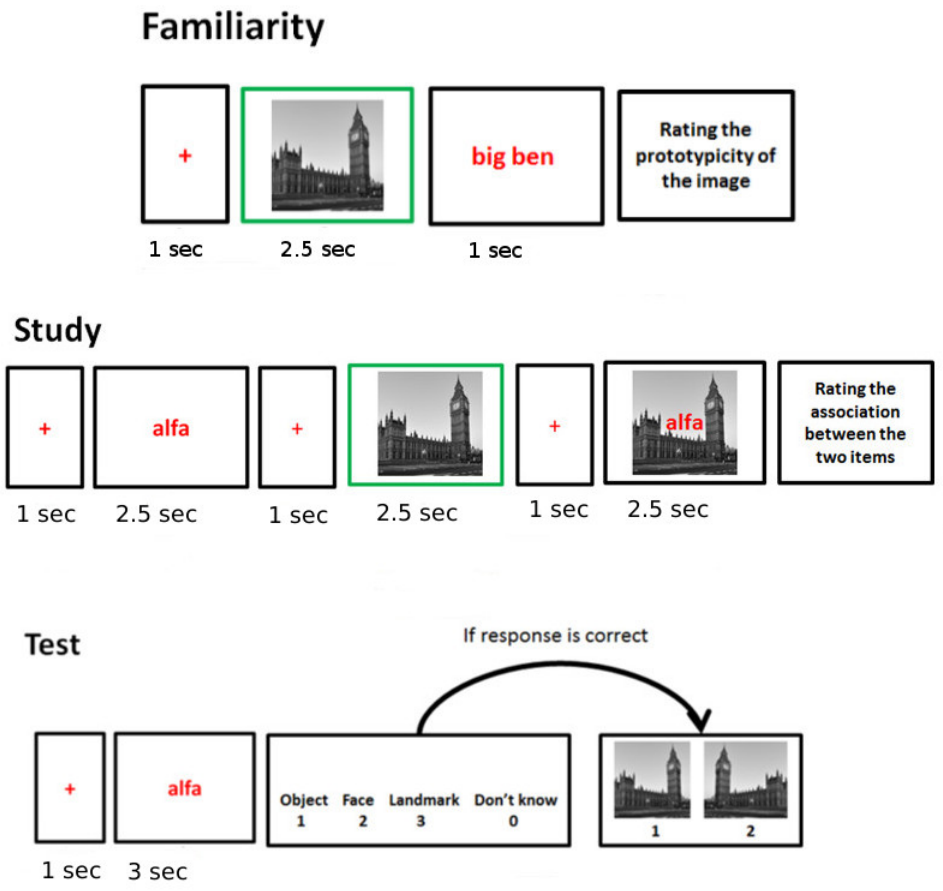


Figure 5.1 Illustration of the experiment from which the data of this work originates. The study consisted of three phases. During the familiarity phase the participants were showed pictures from three categories:faces, objects, and landmarks. During the study phase they were asked to memorize word/picture pairs. For the test part the participants were tasked with remembering the picture that corresponded to the displayed word. Only the study phased was used in this work. [Image source Inês Bramão Department of Psychology Lund University.]

terns during the encoding of the study phase were replayed during the recollection of the test phase.

The data

EEG was measured throughout the entire experiments with a frequency of 2048 Hz. This high frequency was used since it was necessary to detect triggers that were sent each time the picture changed. The data was later down-sampled to 512 Hz. The average of the left and the right mastoid is used as a reference for the data. That means that all other measurements are relative to the reference.

For each part one EEG segment was saved. For the familiarity part data was saved for the display of the picture. For the study part the data was also saved for the display of the picture. Finally, for the test part data was saved for the display of the word.

Data was saved from 1 second before the stimuli and up to 2.5 second after the stimuli. The reason for saving data before the stimuli and for so long after was to allow for frequency calculations. One such segment is called a trial.

Artifacts coming from sources such as eye and muscle movement were removed using independent component analysis and the expertise at the psychology department. The removal of eye artifacts was also helped by measurements from VEOG and HEOG channels that measured vertical and horizontal eye movements. Trials that could not be cleaned up to a satisfactory degree were removed.

2. Classification Setup

Features

Power spectral density will be used as features throughout the experiments. The data to be classified will be split into time points. For each time point the magnitude of a set of frequencies will be calculated. The phase of the signal will not be used. This will be done for each channel yielding a three dimensional space to choose features from, see Figure 5.2 for a visualization.

For a static classifier, features can be chosen arbitrarily over this three dimensional grid. One might for example choose channel four, frequency magnitude at five Hz and time point four as a feature. If one instead wants to use a dynamic classifier the features must instead be time series. An example of a feature is then channel four, frequency magnitude at five Hz and measurements from time point two to time point eight.

Time bins

The goal of the study at the psychology department was to explore if part of the process of remembering (encoding) a memory was replayed during the retrieval of that memory. Those processes clearly will not be the same all the time. If one was

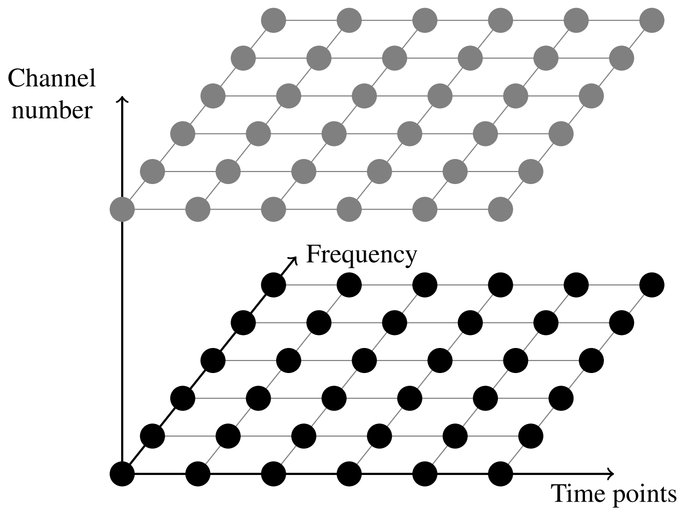


Figure 5.2 Illustration of the possible features to choose from. For each channel the magnitude of some frequencies are calculated at different time points.

to build a classifier for the entire signal it would be impossible to tell at what time eventual similarities happened.

To solve this, different classifiers are trained for different time intervals, called time bins. A time bin starts at t_0 and ends at time t_1 and uses only data from that time period for training. The training data is split into time bins that cover the whole trial (could be overlapping). See Figure 5.3 for an illustration. All those time bins are then used to classify the test phase. A classifier trained for a time bin that starts at t_0 and ends at t_1 for the study data needs not to classify the same interval for the test data. The interval it is run on needs, however, to have the same interval size.

The time bin technique is useful also in other BCI applications. Waiting for the whole signal before doing any classification would however lead to systems with large delays. Daly et al [Daly et al., 2011] investigated the effect on accuracy for different window sizes when identifying single taps. Smaller windows increase the

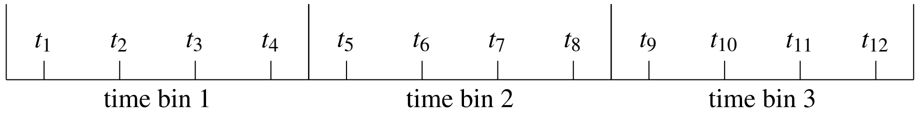


Figure 5.3 Illustration of the time bin concept. The samples are split into several independent time bins containing some of the samples each.

theoretical bit rate but at the cost of decreased accuracy.

The cited work also used sliding time windows. With this technique the output of the classifier can switch with every time point instead of the length of the time window, allowing the classification output to change more often without loss of accuracy. Sliding time bins require to use some information from prior time bins to be efficiently implemented, but in theory each time bin can be handled independently of other time bins.

3. Previous Work

Jafarpour et al. [Jafarpour et al., 2014] studied at what point in recollection the encoding of the memory was replayed. The study consisted of 3 phases. During the first phase the participants were shown pictures of faces or scenes, which they should not have seen before. If they had, the picture was removed from the experiment. One classifier for faces and one for scenes were then trained from the data that was kept.

For the second phase the participants were tasked with memorizing word-picture pairs. For the last phase the participants were shown a word and asked to try to remember the face/scene it was associated with. The classifier built was then used during the third phase with the idea being that a high classification score during a time slot meant that the original memory was replayed.

The machine learning used MEG readings from 274 channels. The data was then transformed to the time frequency domain using 5 cycles Morlet wavelets (continuous wavelet transform). The frequency scale went from 8-45 Hz with a resolution of 1 Hz. Analysis was performed in time bins of 66 ms and there were 21 time points in each time bin. Data was normalized by z-scoring (data normalized to zero mean and unite variance) at each time point, frequency and channel across trials. A support vector machine was the choice of classifier.

First they looked for category specific patterns during encoding. This was done by using 10 fold cross validation. This means that for each time bin, 10 different classifiers were trained. Each classifier uses a different 10 % of the data for validation. The accuracy was then the mean of the accuracy for the 10 classifiers. For each validation pass a feature selection was run by only keeping those features that were significantly different between the two categories in a T-test with 0.05 level. The performance was then calculated as the average of all the 10 classifiers. The classifiers that were significantly different from zero were then trained on all the data in their time bin.

These classifiers were then used to check if the pattern present during their time bin was later replayed during encoding which there were no indications of in the classifier performance.

They then checked if category specific patterns emerged during recollection. This was done by running the classifier trained on data from the encoding time bin centered around 180 ms on the recollection trials where the participants correctly recognized the word cue. This gave significant classifier performance for the time bins centered at 446 and 513 ms. Sliding time bins were not used. This could possibly give better performance as there is no guarantee that the replayed pattern is aligned with one of the time bins.

Finally, only those recollection trials where the participants also chose the correct picture were tested. Now the classifier was only significant for 513 ms.

6 Method

1. Software

The matlab toolbox Fieldtrip toolbox [Oostenveld et al., 2011] was used in this work. It is developed by Donders Institute for Brain, Cognition and Behaviour in Nijmegen, the Netherlands, and have multiple methods for analysis of EEG and MEG data. Methods for filtering the signals and to do time frequency calculations using continuous wavelets were used.

For SVM classification LIBSVM [Chang and Lin, 2011] was used. It is written in C++ and contains a Matlab interface. It supports one-vs-one multiclass SVM classification.

2. Static Classifier

Introduction

The goal of this part was to explore alternatives to the ANOVA + linear SVM method that was used by Jafarpour [Jafarpour et al., 2014] and by the psychology department. It was decided to keep SVM as the classification method so the focus could be kept on the feature selection. Other than ANOVA, two feature selection methods were tested: correlation based feature selection and genetic algorithm.

Experiment

As the goal was to test which method performed best, it was important to have a well defined goodness measure. Time bins is a good tool for memory research, but it is hard to find a satisfying goodness measure for it. One could take the mean accuracy over all time bins but it is not apparent that all time bins should be equally important. It is yet unknown in what time segments good classifier performance is expected from physiological reasons.

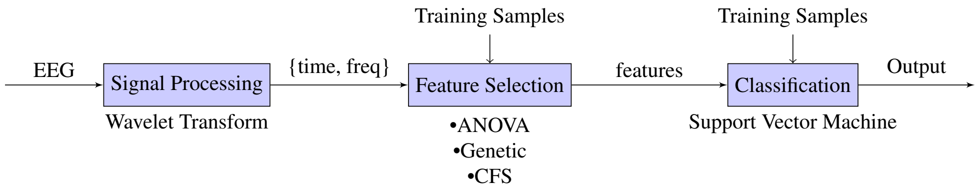


Figure 6.1 Illustration of the classification pipeline used in this project. The EEG signals are first transformed using wavelet transform. Then features are chosen from the possible combination of time and frequency measurements. Those features are fed into a classifier that outputs the prediction of the category for the input.

It was decided that it was better to evaluate classifier performance by classifying the whole time segment at the same time. This could be seen as having one large time bin. Different methods might be better at classifying different length of signals (more efficient at using less data) or more efficient at different parts of the data. However, this approach should serve well as a first evaluation method. If multiple methods are close then further experiments should be conducted.

The classification pipeline of this work have three steps. The first is signal processing, where the original EEG signals are transformed into the frequency domain. The second step is feature selection, where the best features were chosen. Those features are fed into the third step, which is classification. An illustration of the this can be seen in Figure 6.1.

Base feature set For all experiments in this work the base feature set was the frequency magnitude given by continuous wavelet transform.

Comparison with other work Using all the data at the same time instead of working with only a subset of it will mean that the accuracy will be at least as good as the accuracy for the best time bin, and most likely better. This means that it is hard to compare the results with other studies. But since there are no other published results on this study that would be impossible anyway. Instead the comparison will be between the old method, ANOVA, and the new tested in this work.

Rejected methods

Two simple methods were tested initially and rejected due to bad performance. Here follows a short description of them

Principal Component Analysis Principal component analysis (PCA) finds the directions were the data has the highest variance, see [Duda et al., 2000]. PCA was used to find the directions where the spectral features had the largest variance. These features were then fed into a linear support vector machine. The initial results were not good enough to continue perusing this method.

This was not unexpected as there likely is large in-class variance in the data.

Linear Discriminant Analysis Linear discriminant analysis was used in a smiliar way as PCA, i.e. projecting data into some new directions.. The method only gives two directions for the three class problem. This was not sufficient to achieve good classifier performance and was not further investigated.

ANOVA

Method The ANOVA method evaluates each feature independently and performs a one way ANOVA test (Chapter 4.2). There are then two alternatives, either the n most significant features are kept, or all features significant on an α level.

Next the features are normalized using z-scoring, so that all have zero mean and unit variance. This is important so that the SVM does not give higher weight to features with larger values.

For classification a linear SVM was used.

Motivation The usage of a linear kernel is motivated both theoretically and by experiments.

The features chosen by ANOVA are different in means between at least two of the three classes. So in that feature's direction a linear separator should do a good job. In the other directions there is no information. If a non linear kernel was to be used the amount of features would have to be reduced significantly to avoid overfitting. There is no theoretical motivation to do so since there is no indication that the features would be good in a non linear kernel.

By experiments it was found that the previous arguments indeed were correct.

Correlations-Based Feature Selection

The correlation based feature selection (Chapter 4.3) was initially motivated by keeping the number of features low, to reduce overfitting. This is not really needed for linear SVMs, since they handle correlated features well. Furthermore the calculation for inter-feature correlation gets really slow when the time and frequency resolution is high. Due to this, the denominator containing the inter-feature correlation term $\bar{r}_{ff}$ of equation (4.7) was ignored when a linear kernel was used.

The data was z-scored and then discretized. Features were then chosen via correlation-based feature selection and fed into a SVM.

Genetic Algorithm

See Chapter 4.4 for an introduction to genetic algorithms. The data was first z-scored. Then feature selection was done using a genetic algorithm. The fitness function was the classifier performance for the target SVM. To avoid overfitting, the features in the test and training set was randomized.

3. Dynamic Classifier

The idea to use a dynamic classifier came along late into the project. However, it was still decided to introduce the concept and make a quick experiment.

Features

Feature definition In this work a feature was a time series containing the magnitude of a frequency over a channel. It is possible to let each time series have its own start and finish time. But to keep it simple for this initial experiment each time series was forced to start at time zero and end at time T. Time T was chosen by inspection in Figure 6.2 that shows where features that passed the ANOVA test on an $\alpha = 0.1$ level.

So the feature space consisted of all combinations of frequencies and channels. One example would be the time series for the frequency 5 Hz at channel ten.

Feature Selection For feature selection ANOVA tests were calculated for all time points, channels and frequencies. Then a time series was kept if q percent of the samples had significant level above α.

Experiment

For classification, pairwise distance function (introduced in Chapter 3.5) was used. Dynamic time warping was used as the distance metric with the distance as cost functions. All features were forced to have the same warping. The total distance was calculated using the $\mathscr{L}^1$ norm over all channels.

The pairwise distance features were then fed into a linear SVM.

The result was compared with the ANOVA method choosing features from the same feature set and another pair wise distance classifer that used the $\mathscr{L}^2$ norm instead of dynamic time warping for each channel, but the $\mathscr{L}^1$ over the channels.

The goal was to see if this dynamic classifier could outperform the static and if the dynamic time warping was useful.

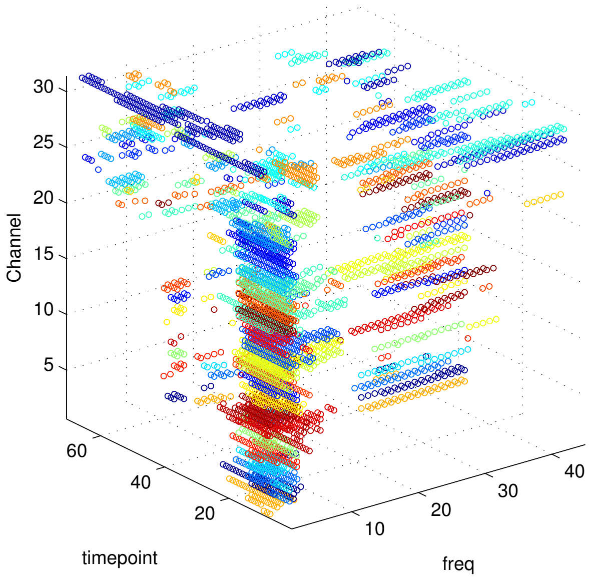


Figure 6.2 Features that passed the ANOVA test on an $\alpha = 0.1$ level. It can be seen that there are more selected features for lower frequencies and earlier time points. Colors correspond to channels.

7

Results

For all cases, 40 samples from each category was used for training. The remaining trials was then used for testing. For subject 9 that was 24 faces, 22 objects and 22 landmarks. One subject was chosen to reduce work.This particular subject was chosen since it gave the highest classification accuracy.

Frequency calculation was done using the function `ft_freqanalysis` in Field Trip toolbox using (Morlet) wavelet and `cfg.width = 5`. The rest of the settings were the defaults.

1. Static Classifier

Frequency is used from 5 to 49 Hz and time from 0 to 1 second. The full time segment was used as the classifier should be able to handle if the features are worse after a certain time point. The choice of 49 Hz was used since there is a lot of disturbances of 50 Hz due to the power line. The lower limit was chosen as 5 Hz, since low frequencies have bad time resolution and would lead to high delays if used in a real-time BCI systems.

ANOVA

The ANOVA method was initially tested for other kernels than the linear. Those results were however around chance and are not presented. The results for a linear kernel are given in Table 7.1.

It was found that the best performance was achieved for a significance level of $\alpha = 0.01$.

Correlation-based feature selection

Correlation-based feature selection was tested for two cases. For the linear kernel only class-feature correlation was used. For the non-linear kernels the feature-feature correlation was also used. Uniform discretization of the features was used

Significance α	time resolution	number of features	C	accuracy %
0.05	0.1	1.8k	1	55.9
0.05	0.1/7	11k	1	67.7
0.01	0.1/7	5k	1	64.7
0.001	0.1/7	2.5k	1	61.8

Table 7.1 Results for ANOVA method for different configurations. Linear kernel was used. The significance describes the significance for the one-way ANOVA test used during feature selection. *C* is the regularization parameter for the SVM. Frequency resolution of 1 Hz was used for all configuration.

for all experiments. The time resolution of 0.1 s and frequency resolution of 5 Hz was used as that gave the best result.

Linear Kernel The SVM regularization parameter C , see (3.28), was set to 1, see Table 7.2.

# features	accuracy %
75	< 33
100	47.1
125	39.7
200	36.8

Table 7.2 Results for correlation based feature selection and linear SVM.

Radial basis function kernel The results were very bad for most parameter configuration, the result presented in Table 7.3 is one which is acceptable. γ is the parameter in the radial basis function, see (3.25).

# features	C	γ	accuracy %
100	1	0.01	41.2

Table 7.3 Results for correlation based feature selection and radial basis function SVM.

None of the kernels could outperform with the ANOVA method.

Genetic Classifier

All genetic algorithm experiments were run with the following base configuration:

Crossover proability	0.8
elitism	turned off
mutation rate	0.05
genome size	100

The fitness function was the classifier performance for the target classifier with randomized training and test set for each generation. 70 percent of the available samples were used to train the classifier for the fitness function and the remaining 30 percent were used for calculating the accuracy.

The best gene was always saved.

All accuracies are the mean of three runs, given in the last column

Linear Kernel For the linear kernel time resolution 0.1/7 s and frequency resolution 1 Hz was used. The regularization parameter C was 1 and the algorithm went through 50 generations before termination, see Table 7.4.

number of features	mean accuracy %	accuracies %
500	60.3	58.8, 60.3 61.8
1000	59.8	55.9, 61.8, 61.8
2000	57.4	55.9, 55.9, 60.3

Table 7.4 Results for genetic algorithm selection and linear SVM.

Polynomial Kernel For the polynomial kernel the time resolution was 0.1 s and frequency resolution was 5 Hz. No configuration of parameters gave results where the accuracy for all the classes was above chance.

Radial basis function For the radial basis kernel the time resolution was after some experimination chosen to be 0.05 s and frequency resolution was 4 Hz. The algorithm went through 100 generations before termination, see Table 7.5.

number of features	C	γ	mean accuracy %	accuracies %
10	1	0.01	51.5	51.5, 50.0, 52.9
10	1	0.1	48.5	47.0, 45.6, 52.9
100	0.01	0.01	51.5	51.5, 51.5, 51.5

Table 7.5 Results for genetic algorithm feature selection and radial basis function SVM.

Summary

Here follows the best results for the different classifiers

feature selection	classifier	accuracy %
ANOVA	Linear SVM	67.7
Genetic algorithm	Linear SVM	60.3
Genetic algorithm	Radial basis SVM	51.5
CFS	Linear SVM	47.1
CFS	Radial basis SVM	41.2

None of the proposed method seems to be able to beat the ANOVA method.

2. Dynamic Classifier

For the dynamic classifier, time from 0 to 0.25 s with a time resolution of 1/3 s was used. The frequency from 5 to 15 Hz was used with a resolution of 1 Hz. This was chosen as it seemed like by visual inspection that configuration contained many good time series. The time resolution was decided after some experiments.

The dynamic classifier kept all time series where all samples $(q = 100\%)$ were significant on $p = 0.2$. Three different classifiers were trained. One based on the ANOVA method with significance level $p = 0.01$. Two dynamic, one where the distance function was the dynamic time warping, and one where the distance was the $\mathcal{L}^1$ norm. The results can be seen in Table 7.6.

Method	Total accuracy %	Face acc %	Landmark acc %	Object acc %
ANOVA	**45.6**	79.2	36.4	18.2
DTW dynamic	**41.2**	62.6	31.8	27.3
$\mathcal{L}^1$ dynamic	**38.2**	58.3	31.8	22.7

Table 7.6 The results of the dynamic time warping experiment.

Here the accuracy for the categories are displayed as there is a large variation between them and some of them were below chance. The reason for this large variation might be that features that were important to distinguish between landmarks and objects were discarded by the visual inspection that chose the features to use for this experiment.

The dynamic classifier could not beat the ANOVA method, but it was not the most well developed classifier. The dynamic time warping proved to slightly increase the classifier results compared to the standard $\mathcal{L}^1$ norm.

8

Discussion and Conclusions

1. Conclusions

The BCI technique is promising and is able to achieve high accuracy. However, so far no system manage to be both fast and accurate without having to focus on a stimuli. In addition, the gear is very immobile and sensitive. For industrial use, the speed of the systems must be increased, while atleast maintaining accuracy. For industrial use, the sensors needs to achieve a higher combination of portability and quality.

The results of the classifier on the data used could not be improved by the two feature selection method tested.

The dynamic time warping experiment showed that there is some information to be gained by using the time dynamics, however the magnitude of that information was very low for this particular dynamic classifier.

2. Discussion

For the experimental part of this work two things can be noted. Firstly, the genetic algorithm had its best accuracy for fewer features than the ANOVA method for linear kernel. That might be an indication that there is some overfitting problem in the configuration and implementation of the genetic algorithm. Even with that solved, it seems far fetched that it could beat the ANOVA method.

Secondly, the results for the non linear kernels were worse than the linear ones. Furthermore they had to use small γ so they were not that complex. It seems like it is best to use a linear kernel for the data set and the features used.

3. Future Work

There are some techniques that could be explored in future work to improve the classifier. Outlier rejection or sample weighting, where the error penalty depends on each samples could help deal with the possible noisy samples. Feature weighting could also be explored. At the moment all features are assumed to be equally important as they are normalized the same way. Features could be normalized differently depending on how good they seem to be. This would allow for different features to have different impact.

It is the authors opinion that the dynamic classifier should be explored further. The first thing to implement would be to allow each time series to start and end at different times. Normalization of both time series and pair wise distance could be explored. Another dynamic classifier could also be used, for example a hidden Markov model. Pair wise distance function SVM was used for its simplicity. Even if it is not superior on this data set it is expected to perform better in non-clinical applications were the signals are not as well aligned.

Features and classifier should be chosen with the problem in mind. But this work used simple features and a classifier that is known to do well with high dimensional data. Overall the approach have been of a somewhat black box nature. We have some inputs coming from something and we want to classify them.

This was fine for this project as it tried to improve upon old results and used the same setup as previous work did. But it is my opinion that combining knowledge of different field to produce more meaningful features is likely to improve the result. A more advance feature could, for example, take into account interactions between different parts of the brain explicitly. This will require cooperation between researches from different fields.

I also think inspiration could be gained from the signal processing field were dynamic classifiers have been studied for a long time. I envision that most BCI classifiers will be dynamic just as the process is.

Bibliography

Ang, K. K., C. Guan, K. S. G. Chua, B. T. Ang, C. W. K. Kuah, C. Wang, K. S. Phua, Z. Y. Chin, and H. Zhang (2008). “A clinical evaluation of non-invasive motor imagery-based brain-computer interface in stroke”. In: *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, pp. 4178–4181.

Berg, A. T., S. F. Berkovic, M. J. Brodie, J. Buchhalter, J. H. Cross, W. Van Emde Boas, J. Engel, J. French, T. A. Glauser, G. W. Mathern, S. L. Moshé, D. Nordli, P. Plouin, and I. E. Scheffer (2010). “Revised terminology and concepts for organization of seizures and epilepsies: report of the ilae commission on classification and terminology, 2005–2009”. *Epilepsia* **51**:4, pp. 676–685. ISSN: 1528-1167. DOI: `10.1111/j.1528-1167.2010.02522.x`. URL: `http://dx.doi.org/10.1111/j.1528-1167.2010.02522.x`.

Blankertz, B., G. Dornhege, M. Krauledat, K.-R. Muller, V. Kunzmann, F. Losch, and G. Curio (2006). “The Berlin Brain-Computer Interface: EEG-based communication without subject training”. *IEEE transactions on neural systems and rehabilitation engineering* **14**:2, pp. 147–152.

Brémaud, P. (2013). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Vol. 31. Springer Science & Business Media, New York.

Brunner, C., R. Scherer, B. Graimann, G. Supp, and G. Pfurtscheller (2006). “Online control of a brain-computer interface using phase synchronization”. *IEEE Transactions on Biomedical Engineering* **53**:12, pp. 2501–2506.

Chae, Y., J. Jeong, and S. Jo (2012). “Toward brain-actuated humanoid robots: asynchronous direct control using an eeg-based bci”. *IEEE Transactions on Robotics* **28**:5, pp. 1131–1144.

Chang, C.-C. and C.-J. Lin (2011). “Libsvm: a library for support vector machines”. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2**:3. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, accessed 2016-07-15, p. 27.

Chong, M. S. T. (2013). *Parameter and state estimation of nonlinear systems with applications in neuroscience*. PhD thesis. University of Melbourne, Department of Electrical and Electronic Engineering.

Cortes, C. and V. Vapnik (1995). "Support-vector networks". *Machine Learning* **20**:3, pp. 273–297. DOI: `10.1023/A:1022627411411`. URL: `http://dx.doi.org/10.1023/A:1022627411411`.

Cristianini, N. and J. Shawe-Taylor (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, Cambridge.

Cuturi, M. (2011). "Fast global alignment kernels". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 929–936.

D. Gupta, S. G. (2012). "An overview of methods maintaining diversity in genetic algorithms". *International Journal of Emergin Technology and Avanced Engineering* **2**:5, pp. 56–60.

Daly, I., S. J. Nasuto, and K. Warwick (2011). "Single tap identification for fast BCI control". *Cognitive Neurodynamics* **5**:1, pp. 21–30. ISSN: 1871-4099.

Daly, I., S. J. Nasuto, and K. Warwick (2012). "Brain computer interface control via functional connectivity dynamics". *Pattern recognition* **45**:6, pp. 2123–2136.

Duda, R. O., P. E. Hart, and D. G. Stork (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience, New York. ISBN: 0471056693.

"Dynamic Time Warping" (2007). In: *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 69–84. ISBN: 978-3-540-74048-3. DOI: `10.1007/978-3-540-74048-3_4`. URL: `http://dx.doi.org/10.1007/978-3-540-74048-3_4`.

Eiben, A. E., R. Hinterding, and Z. Michalewicz (1999). "Parameter control in evolutionary algorithms". *IEEE Transactions on Evolutionary Computation* **3**:2, pp. 124–141. ISSN: 1089-778X. DOI: `10.1109/4235.771166`.

Fayyad U & Irani, K (1993). "Multi-interval discretization of contrinuous-values attributes for classification learning". In: *Proceedings of the Thirteenth International Joint Conference on Artifical Intelligence*, pp. 1022–1027.

Fazel-Rezai, R., B. Z. Allison, C. Guger, E. W. Sellers, S. C. Kleih, and A. Kübler (2012). "P300 brain computer interface: current challenges and emerging trends". *Frontiers in Neuroengineering* **5**:14. ISSN: 1662-6443. DOI: `10.3389/fneng.2012.00014`. URL: `http://www.frontiersin.org/neuroengineering/10.3389/fneng.2012.00014/abstract`.

Feyereisen, T. *The pilot brain*. accessed online `https://aerospace.honeywell.com/en/blogs/2016/april/the-pilot-brain` 2016-06-30.

Gudmundsson, S., T. P. Runarsson, and S. Sigurdsson (2008). “Support vector machines and dynamic time warping for time series”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2772–2776. DOI: `10.1109/IJCNN.2008.4634188`.

Hall, M. A. (1998). *Correlation-based feature selection for machine learning*. PhD thesis. University of Waikato, Department of Computer Science.

Hawkins, D. M. (2004). “The problem of overfitting”. *Journal of chemical information and computer sciences* **44**:1, pp. 1–12.

Hjorth, B. (1970). “EEG analysis based on time domain properties”. *Electroencephalography and Clinical Neurophysiology* **29**:3, pp. 306 –310. ISSN: 0013-4694. DOI: `http://dx.doi.org/10.1016/0013-4694(70)90143-4`. URL: `http://www.sciencedirect.com/science/article/pii/0013469470901434`.

Hramov, A. E., A. A. Koronovskii, V. A. Makarov, A. N. Pavlov, and E. Sitnikova (2015). *Wavelets in Neuroscience*. Springer-Verlag, Berlin Heidelbergs.

Hsu, C.-W. and C.-J. Lin (2002). “A comparison of methods for multiclass support vector machines”. *IEEE Transactions on Neural Networks* **13**:2, pp. 415–425.

Huang, S. H. (2015). “Supervised feature selection: a tutorial”. *Artifical Intelligence Research* **4**:2, pp. 22–32.

Hwang, H.-J., S. Kim, S. Choi, and C.-H. Im (2013). “EEG-based brain-computer interfaces: a thorough literature survey”. *International Journal of Human-Computer Interaction* **29**:12, pp. 814–826. DOI: `10.1080/10447318.2013.780869`. eprint: `http://dx.doi.org/10.1080/10447318.2013.780869`. URL: `http://dx.doi.org/10.1080/10447318.2013.780869`.

Jafarpour, A., L. Fuentemilla, A. J. Horner, W. Penny, and E. Duzel (2014). “Replay of very early encoding representations during recollection”. *The Journal of Neuroscience* **34**:1, pp. 242–248.

Jakobsson, A. (2013). *An Introduction to Time Series Modeling*. 2nd edition. Studentlitteratur, Lund.

Joachims, T. (1998). “Text categorization with support vector machines: learning with many relevant features”. In: Nédellec, C. et al. (Eds.). *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 137–142. ISBN: 978-3-540-69781-7. DOI: `10.1007/BFb0026683`. URL: `http://dx.doi.org/10.1007/BFb0026683`.

Juang, B. H. and L. R. Rabiner (1991). “Hidden Markov Models for Speech Recognition”. *Technometrics* **33**:3, pp. 251–272. DOI: `10.1080/00401706.1991.10484833`. eprint: `http://www.tandfonline.com/doi/pdf/10.1080/00401706.1991.10484833`. URL: `ttp://www.tandfonline.com/doi/abs/10.1080/00401706.1991.10484833`.

Krusienski, D. J., D. J. McFarland, and J. R. Wolpaw (2006). “An evaluation of autoregressive spectral estimation model order for brain-computer interface applications”. In: *Engineering in Medicine and Biology Society, 2006. EMBS’06. 28th Annual International Conference of the IEEE*. IEEE, pp. 1323–1326.

LaFleur, K., K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He (2013). “Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface”. *Journal of Neural Engineering* **10**:4, p. 046003. URL: `http://stacks.iop.org/1741-2552/10/i=4/a=046003`.

Lee, T.-W. (1998). “Independent component analysis”. In: *Independent Component Analysis: Theory and Applications*. Springer US, Boston, MA, pp. 27–66. ISBN: 978-1-4757-2851-4. DOI: `10.1007/978-1-4757-2851-4_2`. URL: `http://dx.doi.org/10.1007/978-1-4757-2851-4_2`.

Lei, H. and B. Sun (2007). “A study on the dynamic time warping in kernel machines”. In: *Signal-Image Technologies and Internet-Based System, 2007. SITIS ’07. Third International IEEE Conference on*, pp. 839–845. DOI: `10.1109/SITIS.2007.112`.

Liu, Y. and T. Khoshgoftaar (2004). “Reducing overfitting in genetic programming models for software quality classification”. In: *High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on*, pp. 56–65. DOI: `10.1109/HASE.2004.1281730`.

Lotte, F, M Congedo, A Lécuyer, F Lamarche, and B Arnaldi (2007). “A review of classification algorithms for EEG-based brain–computer interfaces”. *Journal of Neural Engineering* **4**:2, R1. URL: `http://stacks.iop.org/1741-2552/4/i=2/a=R01`.

Murakami, S. and Y. Okada (2006). “Contributions of principal neocortical neurons to magnetoencephalography and electroencephalography signals”. *The Journal of Physiology* **575**:3, pp. 925–936. ISSN: 1469-7793. DOI: `10.1113/jphysiol.2006.105379`. URL: `http://dx.doi.org/10.1113/jphysiol.2006.105379`.

Nicolas-Alonso, L. F. and J. Gomez-Gil (2012). “Brain computer interfaces, a review”. *Sensors* **12**:2, pp. 1211–1279.

Obermaier, B, C Guger, C Neuper, and G Pfurtscheller (2001). “Hidden Markov models for online classification of single trial EEG data”. *Pattern Recognition Letters* **22**:12. Selected Papers from the 11th Portuguese Conference on Pattern Recognition - {RECPAD2000}, pp. 1299 –1309. ISSN: 0167-8655. DOI: `http://dx.doi.org/10.1016/S0167-8655(01)00075-7`. URL: `http://www.sciencedirect.com/science/article/pii/S0167865501000757`.

Oostenveld, R., P. Fries, E. Maris, and J.-M. Schoffelen (2011). “Fieldtrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data”. *Intell. Neuroscience* **2011**, 1:1–1:9. ISSN: 1687-5265. DOI: `10.1155/2011/156869`. URL: `http://dx.doi.org/10.1155/2011/156869`.

Penny, W. D., K. J. Friston, J. T. Ashburner, S. J. Kiebel, and T. E. Nichols (2011). *Statistical parametric mapping: the analysis of functional brain images*. Academic press, Great Britain.

Percival, D. B. and A. T. Walden (1993). *Spectral analysis for physical applications*. Cambridge University Press.

Rejer, I. and K. Lorenz (2013). “Genetic algorithm and forward selection for feature selection in eeg feature space”. *Journal of Theoretical and Applied Computer Science* **7**:2, pp. 72–82.

Sandsten, M. (2016). *Time-frequency analysis of time-varying signals and non-stationary processes*. Lund University, Centre for Mathematical Sciences, available online at `http://www.maths.lth.se/matstat/kurser/masm26/2016/TIMEFREQkompendie.pdf`.

Schaap, P. (2016). *Solving paralysis using brain computer interfaces*. `http://europe.newsweek.com/brain-computer-interface-technology-469920?rm=eu` accessed 2016-09-25.

Scheffé, H. (1959). *The Analysis of Variance*. John Wiley & Sons, Inc., New York.

Schlögl, A., K Lugger, and G Pfurtscheller (1997). “Adaptive autoregressive parameters for a brain-computer-interface experiment”. In: *Engineering in Medicine and Biology Society*. Vol. 4, pp. 1533–1535.

Srinivasan, R. (1999). “Methods to improve the spatial resolution of eeg”. *International Journal of Bioelectromagnetism* **1**:1, pp. 102–111.

Thorpe, J., P. C. van Oorschot, and A. Somayaji (2005). “Pass-thoughts: authenticating with our minds”. In: *Proceedings of the 2005 Workshop on New Security Paradigms*. NSPW ’05. ACM, Lake Arrowhead, California, pp. 45–56. ISBN: 1-59593-317-4. DOI: `10.1145/1146269.1146282`. URL: `http://doi.acm.org/10.1145/1146269.1146282`.

Whitley, D. (1994). “A genetic algorithm tutorial”. *Statistics and Computing* **4**:2, pp. 65–85. ISSN: 1573-1375. DOI: `10.1007/BF00175354`. URL: `http://dx.doi.org/10.1007/BF00175354`.

Widmaier, E. P., H. Raff, and K. T. Strang (2014). *Vander’s Human Physiology: The mechanisms of Body Function*. 13th ed. Mc Graw-Hill, New York.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER´S THESIS *Date of issue* October 2016 *Document Number* ISRN LUTFD2/TFRT--6019--SE
Author(s) Martin Heyden	*Supervisor* Johan Eker, Ericsson Mikael Johansson, Department of Psychology, Lund University, Sweden Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden (examiner) *Sponsoring organization*

Title and subtitle

Classification of EEG data using machine learning techniques

Abstract

Automatic interpretation of reading from the brain could allow for many interesting applications including movement of prosthetic limbs and more seamless manmachine interaction.

This work studied classification of EEG signals used in a study of memory. The goal was to evaluate the performance of the state of the art algorithms. A secondary goal was to try to improve upon the result of a method that was used in a study similar to the one used in this work.

For the experiment, the signals were transformed into the frequency domain and their magnitudes were used as features. A subset of these features was then selected and fed into a support vector machine classifier. The first part of this work tried to improve the selection of features that was used to discriminate between different memory categories. The second part investigated the uses of time series as features instead of time points.

Two feature selection methods, genetic algorithm and correlation-based, were implemented and tested. Both of them performed worse than the baseline ANOVA method.

The time series classifier also performed worse than the standard classifier. However, experiments showed that there was information to gain by using the time series, motivating more advanced methods to be explored.

Both the results achieved by this thesis and in other work are above chance. However, high accuracies can only be achieved at the cost of long delays and few output alternatives. This limits the information that can be extracted from the EEG sensor and its usability.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-78	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/